

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Cumulative dissertation with the aim of achieving the doctoral degree at the
Faculty of Mathematics, Informatics, and Natural Sciences
Department of Informatics
University of Hamburg

Sustainably Establishing Technical Debt Management in Practice

Submitted by

Marion Wiese

marion.wiese@uni-hamburg.de
Matr.-Nr. 7365660

Date of disputation: 28.05.2026

First Evaluator: Prof. Dr. Eva Bittner
Second Evaluator: Prof. Dr. Carolyn Seaman
Third Evaluator: Dr. Sebastian Werner

Submission Date: 19.03.2026

"Feminism isn't about making women stronger. Women are already strong. It's about changing the way the world perceives that strength." – *G.D. Anderson*

Abstract

Context: Software systems are subject to continuous change due to evolving requirements and contextual shifts, particularly in agile environments, where changing requirements are explicitly welcome. Implementing a maintainable software system often requires more effort than delivering a quick solution. As a consequence, suboptimal technical solutions are introduced that are beneficial in the short term but hinder long-term development and maintenance. The Technical Debt (TD) metaphor was introduced to communicate such trade-offs, comparing difficult-to-maintain implementations to financial debt. Following this metaphor, interest in the form of additional effort to make changes must be paid if the debt is not repaid, i. e., if the suboptimal solution is not improved. Over time, research has identified various TD types, causes and consequences of TD, and TD Management (TDM) activities (i. e., identification, documentation, measurement, prioritization, repayment, monitoring, and prevention).

Despite this extensive body of research, dedicated TDM processes remain uncommon in practice. Research on holistic approaches covering multiple TD types and activities is rare. Socio-technical aspects such as communication and awareness are often neglected, and research findings are insufficiently transferred to industrial contexts.

Objectives: The research goal of this thesis is to understand the reasons behind the poor adoption of TDM processes in practice and to identify ways to make research findings on TDM more applicable. Existing approaches often focus on single TD activities or specific TD types, predominantly code debt, and rarely address the socio-technical challenges of communication, cultural change, or awareness of TD in decision-making situations. Furthermore, concrete, practical guidance on establishing and integrating a TDM process into agile work environments is limited. To this end, this thesis focuses on establishing a holistic and feasible TDM process, concentrating on its socio-technical aspects, i. e., TD communication and TD awareness, as well as TD prevention. In the end, the thesis shall provide *best practices* as a low-level starting point for a general TDM *best practice* library.

Research Design and Methods: The research of this thesis is structured into three steps. Step I analyzes the research area and identifies research gaps through literature reviews, case studies, interviews with IT managers and software architects, surveys, and exploratory analyses. These studies focus on communication structures, changes in awareness due to a TDM process, prevention strategies, and existing TDM practices.

Based on these insights, Step II develops artifacts to address identified research gaps, including a game to foster discussions about TD and a five-step workshop concept to establish a TDM process. The idea behind the workshops is to present the practitioners with a few TD activities per workshop, introduce approaches identified by research for these activities, and let the practitioners decide which approaches to adopt. Additionally, practitioners were encouraged to develop new, useful approaches for their specific situations. During this step, the TD-SAGAT (TD-Situation Awareness Global Assessment Technique) method was developed. TD-SAGAT is an adaptation of a method from psychology for the TD context, designed to measure changes in TD awareness during TDM establishment.

Step III evaluates the five-step workshop concept through action research studies in three companies. The measures determined during the workshops in the *action-planning* phase are applied between workshops in the *action-taking* phase. The *evaluation* and *learning* phases during the workshops allow for changes to the process, following the agile "inspect&adapt" approach. Data collection comprises surveys, meeting observations, TD-SAGAT, and backlog analyses. In this study, the adopted approaches and the observed challenges are further outcomes of the research, i. e., part of the collected data. Comparing results across cases enables the identification of *best practices* and recurring challenges.

Results & Contributions: The results show that socio-technical aspects are critical to successfully establishing a TDM process, and a self-empowered approach is superior to predefined processes for agile working teams.

Communication about TD improves when causes and consequences are determined, discussed, and visualized, for example, using the V4CTD model (Visibility, Cycles & Chains of Causes & Consequences of TD). Communication structures need to be ensured and should account for psychological effects, e. g., factors that hinder or facilitate communication. The TD Game allows for discussion of this critical topic in an emulated, and thus psychologically safe, environment.

The TD-SAGAT method defines the term "TD awareness" and enables its measurement during daily work. The action research studies indicate that establishing a TDM process sustainably increases both subjective and observable indicators of TD awareness when attributes are added to the backlog issue types that remind practitioners to discuss certain aspects of a decision.

TD prevention is closely linked to TD communication and TD awareness. Debiasing architectural decision-making by explicitly considering alternatives, drawbacks, and risks of potential solutions contributes to reducing unintentional TD incurrence. These factors are among the attributes that should be added as reminders to all backlog issues. Recording intentionally incurred TD items and repaying them as part of a project might prevent TD by creating a feedback loop.

The five-step workshop concept provides a feasible, transferable approach to establishing a holistic TDM process. Across three companies, teams integrated all TD activities into their existing agile workflows. The cross-case comparison reveals common elements that can serve as *best practices* and are outlined in the TDM Guide, a whitepaper for practitioners.

Limitations: The findings are limited to teams that are dedicated to changing their processes, willing to invest time in this, and supported by their management. The workshop concept is specifically designed for agile teams who are authorized to adapt their processes. A key constraint seems to be the guidance from an expert familiar with the TDM topic during the workshops.

Conclusions and Future Work: The findings indicate that TDM can be sustainably established when socio-technical aspects are explicitly addressed and when the process is tailored to the team's context. The workshop-based establishment method proved feasible across multiple companies and led to improved communication, increased awareness, and more systematic handling of TD.

Future work includes further evaluating the TDM guide, particularly the need for external guidance, streamlining the five-step workshop concept, scaling TDM to the company level, and developing online course formats to support broader adoption.

Zusammenfassung

Kontext: Softwaresysteme unterliegen aufgrund sich ändernder Anforderungen und Veränderungen des Umfeldes einem ständigen Wandel, insbesondere in agilen Umgebungen, in denen sich ändernde Anforderungen ausdrücklich begrüßt werden. Die Implementierung eines wartbaren Softwaresystems erfordert oft mehr Aufwand als die Bereitstellung einer schnellen Lösung. Infolgedessen werden suboptimale technische Lösungen eingeführt, die zwar kurzfristig von Vorteil sind, aber die langfristige Entwicklung und Wartung behindern. Die Metapher der technischen Schulden (Technical Debt, TD) wurde eingeführt, um solche Kompromisse zu veranschaulichen, indem schwer zu wartende Implementierungen mit finanziellen Schulden verglichen werden. Dieser Metapher folgend müssen Zinsen in Form von zusätzlichem Aufwand für Änderungen gezahlt werden, wenn die Schulden nicht zurückgezahlt werden, d. h. wenn die suboptimale Lösung nicht verbessert wird. Im Laufe der Zeit hat die Forschung verschiedene Arten von TD, Ursachen und Folgen von TD sowie Aktivitäten des TD-Managements (TDM) identifiziert (d. h. TD-Identifizierung, Dokumentation, Messung, Priorisierung, Rückzahlung, Überwachung und Prävention).

Trotz dieser umfangreichen Forschungsarbeit sind dedizierte TDM-Prozesse in der Praxis nach wie vor selten. Forschung zu ganzheitlichen Ansätzen, die mehrere TD-Typen und -Aktivitäten abdecken, ist selten. Soziotechnische Aspekte wie Kommunikation zu TD und Bewusstsein für TD werden oft vernachlässigt, und Forschungsergebnisse werden nur unzureichend auf industrielle Kontexte übertragen.

Ziele: Das Forschungsziel dieser Arbeit besteht darin, die Gründe für die geringe Akzeptanz von TDM-Prozessen in der Praxis zu verstehen und Wege zu identifizieren, wie Forschungsergebnisse zu TDM besser anwendbar gemacht werden können. Bestehende Ansätze konzentrieren sich oft auf einzelne TD-Aktivitäten oder bestimmte TD-Typen, vorwiegend auf Code-Schulden, und befassen sich selten mit den soziotechnischen Herausforderungen der Kommunikation, des kulturellen Wandels oder des Bewusstseins für TD in Entscheidungssituationen. Darüber hinaus gibt es nur wenige konkrete, praktische Anleitungen zur Etablierung und Integration eines TDM-Prozesses in agile Arbeitsumgebungen. Zu diesem Zweck konzentriert sich diese Arbeit auf die Etablierung eines ganzheitlichen und umsetzbaren TDM-Prozesses, wobei der Schwerpunkt auf dessen soziotechnischen Aspekten liegt, d. h. auf TD-Kommunikation und TD-Bewusstsein sowie auf der TD-Prävention. Letztendlich soll die Arbeit *Best Practices* als grundlegenden Ausgangspunkt für eine allgemeine TDM-*Best-Practice*-Bibliothek liefern.

Forschungsdesign und -methoden: Die Forschung dieser Arbeit gliedert sich in drei Schritte. Schritt I analysiert den Forschungsbereich und identifiziert Forschungslücken durch Literaturrecherchen, Fallstudien, Interviews mit IT-Managern und Softwarearchitekten, Umfragen und explorative Analysen. Diese Untersuchungen konzentrieren sich auf Kommunikationsstrukturen, Bewusstseinsveränderungen durch einen TDM-Prozess, Präventionsstrategien und bestehende TDM-Praktiken.

Auf der Grundlage dieser Erkenntnisse werden in Schritt II Artefakte entwickelt, um die identifizierten Forschungslücken zu schließen, darunter ein Spiel zur Förderung von Diskussionen über TD und ein fünfstufiges Workshop-Konzept zur Etablierung eines TDM-Prozesses. Die Idee hinter den Workshops ist, den Praktikern pro Workshop einige TD-Aktivitäten vorzustellen, für diese Aktivitäten Ansätze aus der Forschung zu präsentieren und die Praktiker entscheiden zu lassen, welche Ansätze sie übernehmen möchten. Darüber hinaus wurden die Praktiker dazu ermutigt, neue, nützliche Ansätze für ihre spezifischen Situationen zu entwickeln. In diesem Schritt wurde die TD-SAGAT-Methode (TD-Situation Awareness Global Assessment Technique) entwickelt. TD-SAGAT ist eine Anpassung einer Methode aus der Psychologie an den TD-Kontext, die dazu dient, Veränderungen im TD-Bewusstsein während der Etablierung von TDM zu messen.

In Schritt III wird das fünfstufige Workshop-Konzept anhand von Aktionsforschungsstudien in drei Unternehmen evaluiert. Die in den Workshops der *Aktionsplanungsphase* festgelegten Maßnahmen werden zwischen den Workshops in der *Aktionsphase* umgesetzt. Die

Evaluierungs- und *Lernphase* während der Workshops ermöglichen Anpassungen des Prozesses gemäß dem agilen Ansatz „Inspect & Adapt“. Die Datenerhebung umfasst Umfragen, Beobachtungen von Besprechungen, TD-SAGAT und Backlog-Analysen. In dieser Studie sind die angewendeten Ansätze und die beobachteten Herausforderungen weitere Ergebnisse der Forschung, d. h. Teil der Datensammlung. Der Vergleich der Ergebnisse über die verschiedenen Fälle hinweg ermöglicht die Identifizierung von *Best Practices* und wiederkehrenden Herausforderungen.

Ergebnisse & Beiträge: Die Ergebnisse zeigen, dass soziotechnische Aspekte für die erfolgreiche Etablierung eines TDM-Prozesses entscheidend sind und dass ein selbstbestimmter Ansatz für agile Arbeitsteams vorgefertigten Prozessen überlegen ist.

Die Kommunikation über TD verbessert sich, wenn Ursachen und Folgen ermittelt, diskutiert und visualisiert werden, beispielsweise mithilfe des V4CTD-Modells (Visibility, Cycles & Chains of Causes & Consequences of TD). Kommunikationsstrukturen müssen sichergestellt werden und sollten psychologische Effekte berücksichtigen, z. B. Faktoren, die die Kommunikation behindern oder erleichtern. Das TD-Spiel ermöglicht die Diskussion dieses kritischen Themas in einer emulierten und damit psychologisch sicheren Umgebung.

Die TD-SAGAT-Methode definiert den Begriff “TD-Bewusstsein” und ermöglicht dessen Messung im Arbeitsalltag. Die Aktionsforschungsstudien zeigen, dass die Etablierung eines TDM-Prozesses sowohl subjektive als auch beobachtbare Indikatoren für das TD-Bewusstsein nachhaltig erhöht, wenn den Backlog-Issue-Typen Attribute hinzugefügt werden, die die Praktiker daran erinnern, bestimmte Aspekte einer Entscheidung zu diskutieren.

Die Vermeidung von TD ist eng mit TD-Kommunikation und TD-Bewusstsein verbunden. Die kognitive Entzerrung der architektonischen Entscheidungsfindung durch die explizite Berücksichtigung von Alternativen, Nachteilen und Risiken potenzieller Lösungen trägt dazu bei, das unbeabsichtigte Eingehen von TD zu reduzieren. Diese Faktoren gehören zu den Attributen, die als Erinnerung zu allen Backlog-Elementen hinzugefügt werden sollten. Die Erfassung bewusst verursachter TD-Elemente und deren Rückzahlung als Teil eines Projekts können durch die Schaffung einer Rückkopplungsschleife TD verhindern.

Das Fünf-Schritte-Workshop-Konzept bietet einen praktikablen, übertragbaren Ansatz zur Etablierung eines ganzheitlichen TDM-Prozesses. In drei Unternehmen integrierten Teams alle TD-Aktivitäten in ihre bestehenden agilen Arbeitsabläufe. Der fallübergreifende Vergleich zeigt gemeinsame Elemente auf, die als *Best Practices* dienen können und im TDM-Leitfaden, einem Whitepaper für Praktiker, beschrieben sind.

Einschränkungen: Die Ergebnisse beschränken sich auf Teams, die sich dafür einsetzen, ihre Prozesse zu verändern, die bereit sind, Zeit dafür zu investieren und die von ihrem Management unterstützt werden. Das Workshop-Konzept ist speziell für agile Teams konzipiert, die befugt sind, ihre Prozesse anzupassen. Eine wesentliche Einschränkung scheint darin zu bestehen, dass während der Workshops die Unterstützung durch einen mit TDM vertrauten Experten erforderlich ist.

Schlussfolgerungen und zukünftige Arbeiten: Die Ergebnisse deuten darauf hin, dass TDM nachhaltig etabliert werden kann, wenn soziotechnische Aspekte explizit berücksichtigt werden und der Prozess auf den Kontext des Teams zugeschnitten ist. Die workshopbasierte Etablierungsmethode erwies sich in mehreren Unternehmen als durchführbar und führte zu einer verbesserten Kommunikation, einem gesteigerten Bewusstsein und einem systematischeren Umgang mit TD.

Zukünftige Arbeiten umfassen die weitere Evaluation des TDM-Leitfadens, insbesondere hinsichtlich des Bedarfs an externer Anleitung, die Optimierung des fünfstufigen Workshop-Konzepts, die Skalierung von TDM auf Unternehmensebene sowie die Entwicklung von Online-Kursformaten zur Unterstützung einer breiteren Einführung.

Dedicated to
my husband Jens and my son Nikas
for their endless support.

Acknowledgements

I wish to express my deepest gratitude to all those who contributed to the successful completion of this dissertation.

Heartfelt thanks to my supervisors Prof. Dr. Eva Bittner, Prof. Dr. Matthias Riebisch, and Prof. Dr. André van Hoorn for their invaluable guidance, constructive feedback, and unwavering support throughout this journey.

I am grateful to my colleagues Dr. Sebastian Werner, Dr. Oliver Kopp, Kamila Serwa, Dr. Paula Rachow, Leif Bonorden, Sebastian Gerdes, and Dr. Sandra Schröder for their tireless feedback and collaboration, as well as to Prof. Dr. Carolyn Seaman, Prof. Dr. Klaus Schmid, and Prof. Dr. Uwe Zdun for their inspiring ideas and wise counsel.

My sincere appreciation goes to my collaborators, Dr. Klara Borowa, Dr. Ariane Marion-Jetten, Prof. Dr. Rodrigo Rebouças de Almeida, and Prof. Dr. Mohamed Soliman.

A special thank you also to Anastasia Besier and Kay Gonschior for their support in the Software Campus project, as well as to all Bachelor and Master students whose theses contributed to this dissertation.

Gratitude to Gruner + Jahr Deutschland GmbH for allowing the evaluation of their approach, to TRUMPF SE & Co. KG and DATEV eG. and their employees for enabling the evaluation of my approaches, and to all Software Campus staff for fostering collaboration between research and practice. A special note of thanks to all interview participants who generously shared insights into their challenges.

Finally, I thank my family, Jens and Nikas, for their patience and support, especially during the times when I was away from home.

This work honors the memory of Prof. Dr. Mathias Riebisch and Prof. Dr. André van Hoorn, whose profound influence endures.

Contents

Abstract i

Zusammenfassung iii

Acknowledgements vii

1 Introduction 1
- 1.1 Motivation . . . 1
- 1.2 Research Goal and Research Questions . . . 3
- 1.3 Thesis Outline . . . 5
  - 1.3.1 Part I . . . 5
  - 1.3.2 Part II . . . 5

2 Thesis Foundations 7
- 2.1 Software Quality and Maintainability . . . 7
- 2.2 Technical Debt Metaphor & Definition . . . 7
- 2.3 Technical Debt Conceptual Model . . . 9
- 2.4 Technical Debt Types . . . 11
- 2.5 Technical Debt Causes & Consequences . . . 11
- 2.6 Technical Debt Management . . . 15
  - 2.6.1 Technical Debt Activities . . . 16
  - 2.6.2 Technical Debt Management Maturity Models . . . 21

3 Related Work and Research Gaps 23
- 3.1 Technical Debt Communication . . . 23
  - 3.1.1 Related Work . . . 23
  - 3.1.2 Research Gaps . . . 24
- 3.2 Technical Debt Awareness . . . 25
  - 3.2.1 Related Work . . . 25
  - 3.2.2 Research Gaps . . . 26
- 3.3 Technical Debt Prevention . . . 26
  - 3.3.1 Related Work . . . 26
  - 3.3.2 Research Gaps . . . 27
- 3.4 Technical Debt Management . . . 28
  - 3.4.1 Related Work . . . 28
  - 3.4.2 Research Gaps . . . 29

4 Research Design 31
- 4.1 Research Paradigm . . . 31
- 4.2 Research Strategy . . . 33
  - 4.2.1 Step I – Analysis of Research Area and Gaps . . . 33
  - 4.2.2 Step II – Artifact and Study Preparation . . . 35
  - 4.2.3 Step III – Artifact Evaluation . . . 40
- 4.3 Research Methods . . . 42
  - 4.3.1 Data Collection . . . 43
  - 4.3.2 Data Analysis . . . 44

5 Publications **47**
5.1 Publications . . . 47
5.2 Contributing Publications . . . 48
5.2.1 Publications as Main Author . . . 48
5.2.2 Publications as Co-Author . . . 55

6 Contributions **57**
6.1 Research Contributions . . . 57
6.1.1 Technical Debt Metaphor and Communication Issues . . . 57
6.1.2 V4CTD Model . . . 58
6.1.3 Model for Managing Software Architecture Mistakes . . . 59
6.1.4 Effects of a Technical Debt Management Process . . . 60
6.1.5 Measuring Technical Debt Awareness with TD-SAGAT . . . 61
6.1.6 Awareness Effects of the Technical Debt Management Establishment . . . 61
6.1.7 Uncovered new Research Areas . . . 62
6.1.8 Proposals for Improving Research Applicability in Practice . . . 65
6.1.9 Improving the Collaboration with Tool Vendors . . . 65
6.2 Practical Contributions . . . 66
6.2.1 TAP Framework – a new Technical Debt Repayment Strategy . . . 66
6.2.2 3-Minute Pitch based on the V4CTD Model . . . 67
6.2.3 Communication about Software Architecture Mistakes . . . 68
6.2.4 TechDebt Game . . . 69
6.2.5 Workshop Concept . . . 71
6.2.6 Technical Debt Management Guide . . . 73

7 Discussion of Contributions to Research Questions and Gaps **81**
7.1 RQ1 – How can we optimize the communication about TD? . . . 81
7.2 RQ2 – How can practitioners increase long-term awareness of TD? . . . 85
7.3 RQ3 – How can practitioners prevent TD incurrence? . . . 86
7.4 RQ4 – How can practitioners establish a TDM process? . . . 87

8 Threats to Validity **91**
8.1 Construct Validity . . . 92
8.2 Internal Validity . . . 92
8.3 External Validity . . . 92
8.4 Conclusion Validity . . . 93
8.5 Summary of Limitations . . . 93

9 Conclusion and Future Work **95**
9.1 Conclusion . . . 95
9.2 Future Work . . . 95
9.2.1 Evaluating the TDM Guide . . . 95
9.2.2 Streamlining and Reevaluating the Workshop Concept . . . 96
9.2.3 Online Course for TD Management . . . 96
9.2.4 Scaling TD Management . . . 97

10 Preventing Technical Debt by Technical Debt Aware Project Management **99**

11 Preventing Technical Debt with the TAP framework for Technical Debt Aware Management **111**

12 IT managers' perspective on Technical Debt Management **129**

13 Learning From Each Other: How Are Architectural Mistakes Communicated in Industry? **153**

14 The TechDebt Game - Enabling Discussions about Technical Debt **173**

15 Establishing Technical Debt Management – A Five-Step Workshop Approach and an Action Research Study **185**

16 A Practical Guide for Establishing a Technical Debt Management Process 217

17 Exploring Web Search Engines to Find Architectural Knowledge 231

18 Debiasing Architectural Decision-Making: An Experiment With Students and Practitioners 243

Bibliography 255

A Appendix A – TDM Guide 271

B Additional Material of Contributing Publications 317

Declaration of Artificial Intelligence Use 325

Eidesstattliche Versicherung 327

Identität eingereichter Versionen 329

List of Figures

2.1 The Four Colors of the Backlog according to Kruchten et al. [KNO19] 8
2.2 Conceptual Model of TD [Avg+16a] 9
2.3 Conceptual Model of TD [RMS18] 10
2.4 Causes and Consequences of TD 11
2.5 Causes of TD according to M. Fowler [Fow09] 12
2.6 Causes of TD [Rio+19], p. 6 13
2.7 Consequences (Effects) of TD [Rio+19], p. 7 14
2.8 V4CTD model of Visibility, Cycles & Chains of Causes & Consequences of TD [WB23] 15
2.9 Matrix for Identifying TD Items [WSB26], Chapter A 18

4.1 Research Path Schema by Stol et al. [SF15] 32
4.2 Research Paths Pursued in this Thesis 33
4.3 5-Step Workshop Concept 36
4.4 Alignment of the Workshop Concept and the Research Questions 37
4.5 Action Cycles by Staron [Sta20] (Colors added to indicate the alignment Figure 4.6) 40
4.6 Study Design presenting Action Cycles, Introduced TD Activities (Diagnosing Phase), and Measures (Action-Taking Phase) [WSB26] 42

6.1 Model of Managing Software Architecture Mistakes (SAM) [WBH23] 59
6.2 Matrix for Identifying TD Items across Stakeholders [WSB26], Chapter A . . . 63
6.3 Project Plan with TD Repayment Phase and Feedback loop [Wie+22] 67
6.4 Example of a 3-Minute Slide [WB23] (thin arrows = chains of causes/consequences; thick arrows = cause-effect relation) 68
6.5 Overview of the Game Board [Wie+25] 70
6.6 Publications Contributing to the Workshop Concept (green – contributions and inspirations; orange – results) 71
6.7 Flowchart for Identifying TD Items [WSB26], Chapter A 75
6.8 Example Visualization of TD issues for Planning Purposes [WSB26] Chapter A 79

7.1 Publications per Year with their Contribution to the Research Questions Grey – Step I, Red – Step II, Blue – Step III of the Thesis 81

The figures of the publications included in this dissertation are not listed.

List of Tables

1.1 Part I - Overview . . . 5
1.2 Part II - Publications . . . 5

2.1 Common TD types according to [LAL15]; [RMS18]; [EKD21]; [Ahm+21] . . . 11
2.2 Categories of TD causes [WB23] . . . 16
2.3 Categories of TD consequences [WB23] . . . 16
2.4 TD activities according to [LAL15]; [RMS18] . . . 16
2.5 TD metrics, i. e., cost factors according to [LAL15]; [MB15] . . . 19
2.6 TDM maturity Model according to Yli-Huumo et al.[YHMS16] . . . 21
2.7 TDM maturity model according to Matrini et al. [MBB18] . . . 21
2.8 TDM maturity model Kruchten et al. [KNO19] . . . 21

4.1 Workshop topics and their assignment to the TD activities . . . 38
4.2 Presented approaches with references . . . 38
4.3 Participating companies . . . 41
4.4 Participating teams . . . 41
4.5 Data collection per team (time and interventions) . . . 42
4.6 Data collection per team (meetings and TD-SAGAT) . . . 42

5.1 Publications published during the thesis (Cont. – Publication contributes to the thesis) . . . 47
5.2 Portrait of conference publication No. 1 *(TAP Framework)* . . . 48
5.3 Portrait of journal publication No. 2 *(TAP Framework – extended)* . . . 49
5.4 Portrait of journal publication No. 3 *(IT Managers' Perspective)* . . . 50
5.5 Portrait of conference publication No. 4 *(Learning from each other)* . . . 51
5.6 Portrait of conference publication No. 5 *(TD Game)* . . . 52
5.7 Portrait of journal publication No. 6 *(Establishing TDM)* (Part I) . . . 53
5.8 Portrait of conference publication No. 7 *(Practical Guide)* . . . 54
5.9 Portrait of conference publication No. 8 *(AK in Web Search Engines)* . . . 55
5.10 Portrait of conference publication No. 9 *(Debiasing)* . . . 56

6.1 Overview of research contributions . . . 57
6.2 Overview of practical contributions . . . 66
6.3 *Best practice*-attributes for a TD issue (Chapter A, [WSB26]) . . . 77
6.4 Example of determining priority based on return on investment (ROI). . . . 78

7.1 Contributions per RQ with relations to publications and research gaps (Type: R – Research, P – Practical; Research Gaps: reference Chapter 3) . . . 82

9.1 Shortened workshop agenda without action research elements . . . 96

10.1 Publication No. 1 *(TAP Framework)* . . . 99

11.1 Publication No. 2 *(TAP Framework – extended)* . . . 111

12.1 Publication No. 3 *(IT Managers' Perspective)* . . . 129

13.1 Publication No. 4 *(Learning from each other)* . . . 153

14.1 Publication No. 5 *(TD Game)* . . . 173

15.1 Publication No. 6 *(Establishing TDM)* . . . 185

16.1 Publication No. 7 *(Practical Guide)* 217

17.1 Publication No. 8 *(AK in Web Search Engines)* 231

18.1 Publication No. 9 *(Debiasing)* 243

A.1 TDM Guide from the additional material of publication No. 7 *(Practical Guide)* 271

B.1 Additional Material to Publication No. 1 *(TAP Framework)* 317
B.2 Additional Material to Publication No. 2 *(TAP Framework – extended)* 318
B.3 Additional Material to Publication No. 3 *(IT Managers' Perspective)* 319
B.4 Additional Material to Publication No. 4 *(Learning from each other)* 319
B.5 Additional Material to Publication No. 5 *(TD Game)* 320
B.6 Additional Material to Publication No. 6 *(Establishing TDM)* 321
B.7 Additional Material to Publication No. 7 *(Practical Guide)* 322
B.8 Additional Material to Publication No. 8 *(AK in Web Search Engines)* 323
B.9 Additional Material to Publication No. 9 *(Debiasing)* 323

The tables of the publications included in this dissertation are not listed.

List of Abbreviations

ADD	**A**ttribute **D**riven **D**esign
AK	**A**rchitecture **K**nowledge
API	**A**pplication **P**rogramming **I**nterface
ATD	**A**rchitecture **T**echnical **D**ebt
CI/CD	**C**ontinuous **I**ntegration / **C**ontinuous **D**eployment
CoP	**C**ommunity **o**f **P**ractice
COTS	**C**ommercial-**O**ff-**T**he-**S**helf
DoR	**D**efinition **o**f **R**eady
GUI	**G**raphical **U**ser **I**nterface
IDE	**I**ntegrated **D**evelopment **E**nvironment
KPI	**K**ey **P**erformance **I**ndicator
LLM	**L**arge **L**anguage **M**odel
ML	**M**aschine **L**earning
ROI	**R**eturn **O**f **Investment**
RQ	**R**esearch **Q**uestion
SA	**S**ituation **A**wareness
SAGAT	**S**ituation **A**wareness **G**lobal **A**ssessment **T**echnique
SAM	**S**oftware **A**rchitecture **M**istakes
SE	**S**oftware **E**ngineering
SW	**S**oft**w**are
SWA	**S**oft**w**are **A**rchitecture
TAP	**T**echnical debt **A**ware **P**roject management
TD	**T**echnical **D**ebt
TDM	**T**echnical **D**ebt **M**anagement
TD-SAGAT	**T**echnical **D**ebt-**S**ituation **A**wareness **G**lobal **A**ssessment **T**echnique

The abbreviations of the publications included in this dissertation are not listed.

Chapter 1

Introduction

This first Section briefly motivates this thesis in Section 1.1, names its research goal and main research questions in Section 1.2, and provides an overview of this thesis for orientation in Section 1.3.

1.1 Motivation

Software systems are subject to frequent changes due to new or evolving requirements or context changes. Accordingly, a central goal in the development of software systems is to implement them in a way that facilitates long-term modifiability and maintainability, which is defined in the ISO/IEC standard 25010-2023: Maintainability is the *"capability of a product to be modified by the intended maintainers with effectiveness and efficiency."* [ISO23]. Relevant aspects of Maintainability comprise modularity, modifiability, reusability, analyzability, and testability [ISO23].

This need for change is also reflected in the evolution of IT development processes over the last 10 to 15 years. Since the Agile Manifesto was published, more and more companies have adopted agile development methods for building their software systems, including *"welcome[ing] changing requirements"* [Bec+01]; [Din+12]. This approach is supported by the development of technologies, such as deployment pipelines for continuous integration and continuous development (CI/CD) [PWL18]; [SE20].

However, building a system with maintainability in mind often takes longer than swiftly changing it [GSS16]; [Rio+20]; [AECV25]. Optimizing a system for maintainability typically provides no visible benefit to non-technical stakeholders and is difficult to understand [KNO12]; [Avg+16a]. Accordingly, those optimizations are often given lower priority by business managers [GSS16]; [RK19]. To avoid this, the maintainability-related issues need to be communicated in a way that is comprehensible to non-technical stakeholders to justify the invested effort [Avg+16b]; [Avg+25].

To overcome these **communication** issues, Ward Cunningham introduced the term technical debt (TD) as a metaphor to financial debt to explain that code rewrites may be needed when software is implemented under time pressure [Cun92]. In this metaphor, a difficult-to-maintain implementation or design is interpreted as debt. The resulting maintainability problems are interpreted as interest rates, refactoring as repayment, and refactoring cost as principal. The definition of TD has evolved since then. Currently, a single instance of TD, namely a TD item, can be described as a construct that is beneficial in the short term, but hinders further development and maintenance of the software in the long term [Avg+16a]; [KNO19]. These constructs can occur in various elements of the system or software engineering process, e. g., code, architecture, tests, infrastructure, or requirements [LAL15]; [EKD21]. Multiple causes and consequences of TD have been identified and categorized [Rio+20]; [Ram+21]. A conceptual model of the causes, consequences, and symptoms of TD has been developed and has been gradually refined [Avg+16a]; [JT22]. Additionally, researchers differentiate between intentional TD, resulting from a conscious decision, and unintentional incurred TD, resulting from unconsciously made mistakes or faulty decisions [McC08]; [Fow09].

A lack of **TD awareness** is a frequently cited cause of inadvertently, i. e., unintentionally incurring TD [MB15]; [Ton+17]; [Rio+20]; [Ram+21]. Some studies have described it as a necessity for managing TD [Bes+17]; [Hol+18]; [KNO19], and some report that establishing

a TD management process led to increased TD awareness, but they fall short of proving this claim, e. g., [Eli+15]; [Ock+19]; [RCM22]. Only a few studies have measured TD awareness, typically using one-time surveys that ask participants whether they are aware of TD, e. g., [DOVH19]; [Ber+21].

TD prevention is widely regarded as the preferred approach to addressing TD among practitioners [Tso+20]; [Fre+20b]; [Apa+20]. In the long term, implementing the optimal solution from the outset is cheaper than incurring TD, especially when accounting for potential interest payments [YHMS16]; [Rio+20]. In 2015, Li et al. found in a systematic mapping study that TD prevention was understudied [LAL15]. Subsequent research on how to prevent TD beyond raising TD awareness has been conducted. Still, TD prevention is presented primarily as a problem [Fre+20a]; [Tso+20]; [Apa+20]; [VHB21]. Papers on prevention strategies focus on inadvertent code debt and on tools that help identify these debts, e. g., code smells, before their incurrence [YHMS16]; [Apa+20]; [Tso+20]. Fewer papers focus on preventing TD at the design and architectural levels, which are usually incurred during decision-making [Bor+21b]; [DMR22]. In terms of avoiding debt-incurring decisions, two main aspects must be considered:

1. Stakeholders have to understand the causes of TD [Rio+20]; [Fre+21].
2. Stakeholders have to be aware of TD and its consequences to make conscious and informed decisions [Mor+12]; [Mor+21]; [Pér+21].

TD management (TDM) comprises two major decisions: whether to incur a TD item or not (i. e., TD prevention), and whether to repay a TD item or not [Bus11]. Research on TDM is extensive [LAL15]; [Guo+16]; [YH+16]; [RK19] and typically organized around so-called TD activities, such as TD identification, TD documentation, TD measurement, TD prioritization, TD repayment, TD monitoring, and TD prevention [LAL15]. While TD has been identified as a serious problem in practice, often caused by tight deadlines due to the fast-changing requirements [Avg+16b]; [Rio+20]; [Avg+16a], few IT departments consciously handle TD through a structured TDM process. The survey results by Martini et al. show that only 7.2% of respondents reported having a systematic approach to TD tracking [MBB18]. Two tertiary studies on TDM found that holistic approaches for TDM, i. e., approaches including various TD activities and various TD types, are still not researched [RMS18]; [JT22]. In their vision paper on TDM, Avgeriou et al. specified that TD research currently focuses on single TD types, mostly code debt, but should include various TD types [Avg+23]. Furthermore, in the Dagstuhl Manifesto on "Reframing Technical Debt", Avgeriou et al. find that the transfer of TD research results to practice remains inadequate, and it was defined as a goal on the research roadmap (Phase 3) [Avg+25]. Among others, the manifesto comprises the goals of creating a *"shared 'standard' best-practice processes and guidelines for TDM"*, to *"manage TD in alignment with its context"*, and to formulate *"a minimum viable TDM process"* [Avg+25].

Decision-making, a central element of TDM, is a cognitive process performed by humans. Consequently, TDM involves socio-technical aspects and cannot be addressed solely by technical interventions. These socio-technical aspects further comprise TD communication, TD awareness, and changes in stakeholders' mindsets regarding TD [Avg+25]. In terms of **socio-technical aspects**, the Dagstuhl manifesto names three main challenges [Avg+25]:

- *"Bridge Socio-Technical Gaps with Quantitative and Qualitative Insights"*, i. e., overcoming the issues of transferring research results across contexts and making research results better generalizable.
- *"Integrating Socio-Technical Features into Existing Toolsets"*, i. e., adding TD-specific features in tool sets used by practitioners, e. g., issue-tracking systems to capture TD-related key performance indicators (KPIs).
- *"Promote Management Support and Cultural Shifts for Proactive Technical Debt Management"*, i. e., promoting a change in upper management's perspective on TD and on an organizational level to increase the support for TDM.

1.2 Research Goal and Research Questions

In the previous section, we showed multiple remaining challenges in TD research. Notably, the main question remains: Why are TDM approaches not widely adopted in practice? Research results either appear to be hidden from practitioners or are infeasible in practice as they are rarely adopted [MBB18]; [Avg+25]. Research that focuses on TDM in practice, e. g., prior case studies and action research on this topic, appears to lack transferability to other contexts [RMS18]; [JT22]. A lack of focus on the socio-technical aspects of managing TD appears imminent, and further research on TDM should address these aspects and demonstrate transferability to other contexts as presented in the Dagstuhl Manifesto [Avg+25].

This thesis separates the Management of TD into two main objectives that we plan to investigate, i. e., preventing TD and managing TD if it is incurred:

- Before TD is incurred, we want to ensure that only necessary TD is intentionally incurred, and we would like to prevent unnecessary and unintentional TD incurrence (TD Prevention).
- After TD has been incurred (intentional or unintentional), we need to evaluate these TD items, prioritize them, and repay them if efficient (TD Management).

As shown in the previous section, awareness of TD's consequences and the issues it poses for a system's future development is often lacking [MB15]; [Ton+17]; [Rio+20]; [Ram+21]. In particular, business stakeholders have difficulty understanding the problems arising from TD [GSS16]; [RK19]. However, technical stakeholders also often do not fully understand TD concepts and struggle to continuously apply measures to address TD in their daily work [MBB18]; [RMS18]; [JT22]. This opens two more issues related to TDM, which we explore as part of this thesis:

- We need to optimize communication about TD to convey its concepts, particularly its consequences (TD communication).
- We not only need to increase awareness of the TD topic, but also to ensure that this awareness is maintained during stakeholders' daily work (TD awareness).

RESEARCH GOAL:
The aim of this work is to make research findings on TDM applicable in practice.
Consequently, the guiding question behind this thesis is:
Why is TDM not common in practice, and how can we change this?
The ultimate goal is to provide best practices of TDM and a method to establish TDM in practice.

This thesis aims to leverage practitioners' knowledge to tailor existing research results to their needs. First, we examine the essential requirements for TDM (Step I). Based on this, we develop a method for establishing TDM (Step II) and evaluate it in practice across three separate companies and studies, ensuring that the effort required to establish and manage TD is acceptable to practitioners. The comparison of the resulting TDM processes, then, leads to the definition of best practices (Step III).

During these steps, this research focuses on four previously introduced (socio-technical) dimensions of TDM: (1) Communicating about TD, (2) increasing and maintaining awareness of TD, (3) preventing TD, and (4) managing TD after its incurrence. Consequently, the research questions (RQs) are defined per dimension to guide this research:

RQ1: How can practitioners optimize the communication about TD?

To initiate work on the TD topic and establish a TDM process, it is essential to improve TD communication. In particular, improved negotiations with business stakeholders may secure the time and support needed to implement TDM approaches. Furthermore, heightened awareness may reduce the amount of unintentionally incurred TD items. This thesis aims to identify communication challenges and develop and test ideas to address them.

To this end, we conducted a case study analyzing communication structures in a team that employed a TDM process and comparing them with those of a team that did not use any TDM approach. Additionally, we interviewed IT managers to gather their insights, and

conducted an interview study with software architects on handling mistakes (i. e., faulty decisions) in their domain and how to communicate them. We have developed a game to enable players to gain insights into TD, experience its effects on daily work, and discuss strategies in an emulated, and thus psychologically safe, environment.

RQ 2: How can practitioners increase long-term awareness of TD?

The communication challenges are closely coupled with TD awareness. If stakeholders are unaware that TD exists or do not understand its causes and consequences, they are less willing to invest in TDM. It is a communication issue to raise awareness, but we also need to identify ways to maintain awareness in daily business. To evaluate TD awareness, we specifically need to identify methods for measuring it.

This thesis defines TD awareness and identifies methods for measuring it during TDM establishment. The initial study analyzed TD awareness only subjectively, and long after introducing a TDM process via a survey. To optimize these subjective results, we collaborated with a researcher in psychology to identify an appropriate method that could be adapted to the TD awareness topic. We defined a series of queries from which TD awareness can be derived and have used them in surveys, during interruptions to the practitioners' work, and during meeting observations.

RQ 3: How can practitioners prevent TD incurrence?

TD prevention is likewise closely coupled to TD communication and awareness. TD can be prevented by making intentional, conscious, and rational decisions about its incurrence and by recognizing when such a decision is made (i. e., being aware of TD in those situations), rather than unconsciously deciding for a suboptimal solution option. Specifically, maintaining awareness of TD in fast-paced daily business is challenging in practice. The aim of this thesis is to identify the relationships among these three aspects (communication, awareness, and prevention) and, consequently, to develop methods to prevent TD in practitioners' daily work.

In a first study, we identified the relation of communication, awareness, and prevention of TD. In subsequent studies, we introduced TD prevention methods, such as debiasing decision-making by consciously considering implementation alternatives, their drawbacks, and risks. We applied and evaluated this method in the final action research studies.

RQ 4: How can practitioners establish a TDM process?

The previous RQs collectively contribute to developing a concept for establishing TDM in practice, focusing on the socio-technical challenges, such as communicating TD-related issues, integrating TD aspects into already used issue-tracking tools, and enabling the transferability of results. Based on this initial research and literature reviews, we developed a successive, iterative workshop concept to establish TDM in a practical environment, i. e., within an IT team. During five workshops, participants are presented with previously researched approaches and encouraged to select those that seem feasible to them. This aligns with the free choice of processes by a team in agile development environments and further aligns with the ideas of the agile change management method [Fü21] from the economics domain. Moreover, the iterative nature of action research appears well-suited to this research.

The goal of this establishment method and the corresponding study is twofold: First, we aim to evaluate the success of this workshop concept and its replicability in three companies. Other companies can then adopt the workshop concept itself. Second, after replicating this study three times, this thesis analyzes commonly adopted approaches as a starting point for defining best practices for TDM and identifying common challenges that remain to be addressed and researched. These results will be summarized in a white paper on managing TD, which other practitioners can use without having to complete the five workshops.

1.3 Thesis Outline

This thesis consists of two parts: the first provides an overview of the thesis, including a brief introduction to each contributing study and all contributions, and their alignment with the research gaps. In the second part, all studies are presented in their original form.

1.3.1 Part I

After the introduction in this Chapter 1, the foundations of TD and TDM are presented in Chapter 2. Chapter 3 introduces related work and details research gaps for each research question. Chapter 4 explains the overall research design and provides foundational details on the methodologies used. Chapter 5 lists all papers and publications that contributed to this thesis and provides a short portrait of these. Chapter 6 provides an overview of the contributions of this thesis, including practical and theoretical contributions. The overall contributions to the RQs and research gaps are discussed in Chapter 7. In Chapter 8, the relevant threats to validity, including their mitigation strategies or resulting limitations, are discussed, following the guidelines by Wohlin [Woh+12] and Staron's book on action research [Sta20]. Chapter 9 summarizes the work and provides an outlook on future work, including already ongoing studies. Table 1.1 provides an overview of these chapters.

Chapter	Name
1	Introduction
2	Thesis Foundations
3	Related Work and Research Gaps
4	Research Design
5	Publications
6	Contributions
7	Discussion of Contributions to Research Questions and Gaps
8	Threats to Validity
9	Conclusion & Future Work

TABLE 1.1: Part I - Overview

1.3.2 Part II

The second part supplements the published papers and journal articles in their original format as listed in Table 1.2. The table introduces publication numbers that will be referenced throughout PART I of this thesis to align the presented information with these publications.

Chapter	Publication No.	Publication
10	1 *(TAP Framework)*	*Preventing Technical Debt by Technical Debt Aware Project Management* (2021)
11	2 *(TAP Framework – extended)*	*Preventing Technical Debt with the TAP framework for Technical Debt Aware Management* (2022)
12	3 *(IT Managers' Perspective)*	*IT managers' perspective on Technical Debt Management* (2023)
13	4 *(Learning from each other)*	*Learning From Each Other: How Are Architectural Mistakes Communicated in Industry?* (2023)
14	5 *(TD Game)*	*The TechDebt Game - Enabling Discussions about Technical Debt* (2025)
15	6 *(Establishing TDM)*	*Establishing Technical Debt Management – A Five-Step Workshop Approach and an Action Research Study* (2026)
16	7 *(Practical Guide)*	*A Practical Guide for Establishing a Technical Debt Management Process* (2026)
17	8 *(AK in Web Search Engines)*	*Exploring Web Search Engines to Find Architectural Knowledge* (2021)
18	9 *(Debiasing)*	*Debiasing Architectural Decision-Making: An Experiment With Students and Practitioners* (2025)
Chapter A		*A Practical Guide to Establishing Technical Debt Management* (Whitepaper) (2026)

TABLE 1.2: Part II - Publications

Chapter 2

Thesis Foundations

2.1 Software Quality and Maintainability

The ISO 25010:2023 [ISO23] is the international standard for assessing the quality of *information and communication technology products* and *software products*. The standard specifies quality attributes that should be evaluated to ensure a system's value extends beyond mere feature development. Balancing quality attributes in software development is crucial due to conflicting interests and trade-offs. Practitioners must evaluate stakeholders' needs and preferences regarding these attributes, as well as the risks associated with each decision. The standard assists in identifying relevant quality characteristics by listing nine quality attributes and their associated characteristics. The nine quality attributes are: Functional Suitability, Performance Efficiency, Compatibility, Interaction Capability (named "Usability" in the standard's previous version), Reliability, Security, Maintainability, Flexibility, and Safety.

TD, the core topic of this thesis, is typically a problem of undervalued Maintainability, though it may also influence other quality attributes. Maintainability is defined as the *"capability of a product to be modified by the intended maintainers with effectiveness and efficiency"* [ISO23]. Maintainability is crucial to remaining competitive in the market, especially when a system is long-lasting and constantly evolving. Maintainability is composed of the following sub-characteristics:

- *Analyzability* is the capability to understand a system or a program to efficiently identify causes of an error, change the system according to new requirements, and determine potential side effects of a change.
- *Modularity* is the way a system is separated into different components. Each module should be designed to focus on a single concern to avoid cascading effects. This means that when a change to a system is necessary, subsequent changes in other modules should be kept to a minimum.
- *Modifiability* refers to the ability to change a system efficiently and without introducing errors or defects.
- *Testability* means a system should be created in a way that is easy to test. This means, in particular, that components must be testable in isolation, without relying on adjacent components.
- *Reusability* means that components should be built in a way that makes it easy to use them in various contexts, so changes to one component remain isolated within it. In contrast, if a function is implemented twice (e. g., a similar function but in different components), changes to this function must be applied to both copies, making the change process time-consuming and error-prone.

2.2 Technical Debt Metaphor & Definition

The rate of change of software systems due to new or evolving requirements or environmental changes has increased substantially over the last few decades. In particular, the adoption of agile development methods has accelerated this rate over the last two decades [Bec+01]; [Din+12]. Hence, the goal of implementing software systems in ways that enable long-term change, placing greater emphasis on system maintainability, has become increasingly important. Elements that limit maintainability should be prioritized and addressed. However,

these modifications usually provide no visible benefit to non-technical stakeholders and are difficult to understand. Thus, business leaders often prioritize developing new features over them. Ward Cunningham introduced the TD metaphor to make this problem more understandable to business managers [Cun92].

> *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.* [Cun92]

The term "technical debt" is therefore a (technical) metaphor for financial debt, and its terminology is based on terms familiar from the banking sector, such as interest or repayment [McC08]; [Guo+11]; [LAL15].

- *Debt* is a sub-optimal implementation or design of a system. While the system still performs as expected, TD will limit its changeability, i. e., maintainability.
- *Repayment* is the action of removing TD from a system.
- *Principal* is the cost, i. e., time and budget, necessary to repay TD.
- *Interest* is the additional cost that arises when TD is not repaid. For example, if implementing a new feature takes longer due to unnecessarily complex code structures, the additional time spent understanding the complexity (analyzability) and testing these structures (testability) is treated as overhead. Interest in particular is often overlooked in discussions about TD and can be difficult to grasp.

A single instance of TD is called a TD item to distinguish a specific problem from a diffuse "TD in the system" [Guo+11]; [Avg+16a].

After some discussion about the exact definition, a group of researchers developed the following definition during a Dagstuhl seminar:

> *In software-intensive systems, technical debt is a collection of design or implementation constructs that are expedient in the short term, but set up a technical context that can make future changes more costly or impossible. Technical debt presents an actual or contingent liability whose impact is limited to internal system qualities, primarily maintainability and evolvability.* [Avg+16a]

The part entitled "future changes" is particularly relevant, as it logically explains two aspects of TD: First, this shows that TD becomes an issue only if there are plans to modify the system or the specific component impacted by the debt. If the code is set to remain unchanged, as in a legacy system that is not undergoing further modifications and will be replaced, then the TD is not a concern. Second, in the last decade, agility has emerged as a significant focus in software development. A fundamental aspect of agility is "embracing change," which entails welcoming ongoing changes in an IT system [Bec+01]; [Din+12]. This concept is also evident in topics such as "continuous integration" and "continuous deployment" [PWL18]; [SE20]. These days, a code base and its associated environment are evolving continuously, in some instances on a minute-by-minute basis. This is why TD has become increasingly important in recent years: its effects become visible when a system is changed, which we now aim to do continuously.

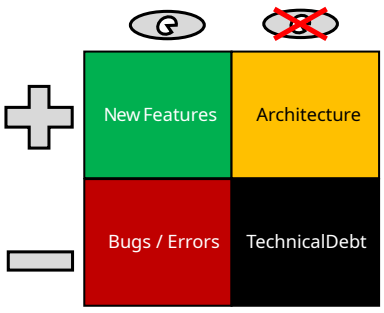


FIGURE 2.1: The Four Colors of the Backlog according to Kruchten et al. [KNO19]

Another way to classify TD is based on the "four colors of the backlog" according to Kruchten et al. [KNO19]. According to this classification, backlog issues in an issue-tracking

system can be distinguished by whether they represent a positive or negative value for the system and by whether they are visible or invisible to business stakeholders (i. e., customers or users). This results in the classification shown in Figure 2.1. While new features and architecture provide positive value to the system, errors and TD represent negative value. New features and bugs can be observed by business stakeholders, which leads them to prioritize these issues. In contrast, creating or improving a system's architecture and repaying TD need to be explained and justified.

2.3 Technical Debt Conceptual Model

As a result of the 2016 Dagstuhl workshop on managing TD, researchers introduced a consolidated conceptual model of TD [Avg+16a], which is presented in Figure 2.2.

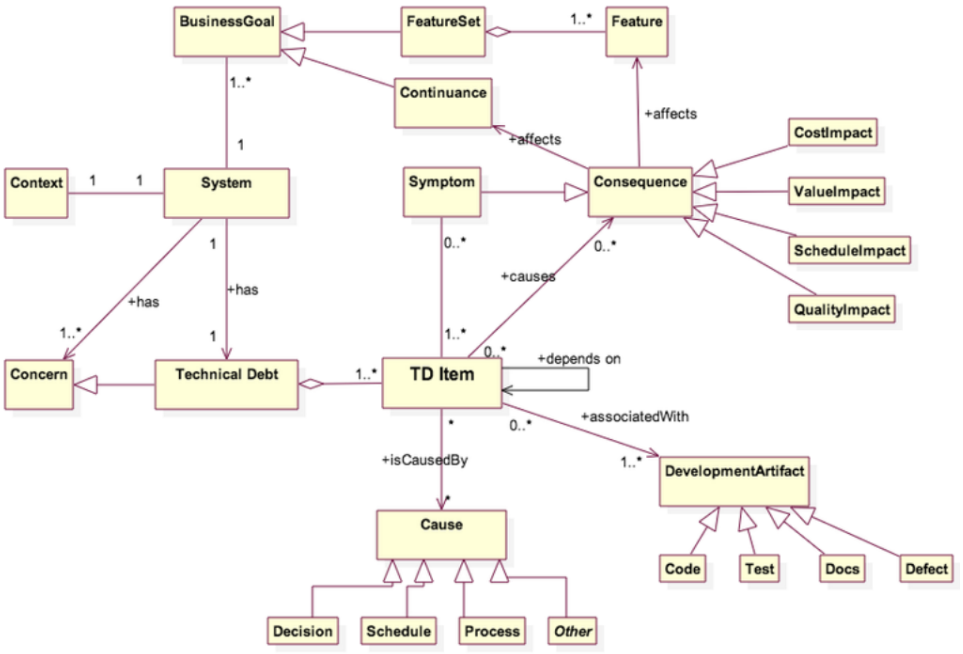


FIGURE 2.2: Conceptual Model of TD [Avg+16a]

The model revolves around a *TD item*, i. e., a single instance of TD. TD *causes* are of various natures and lead to a *TD item*. The entirety of all TD items is termed the *TD* of a *system*, which is one *concern* of a *system*. A *TD item* is associated with a *development artifact* like code, test, or documentation. It leads to various *consequences*, which can be observed by their *symptoms*. The *consequences* may affect a *feature* and the *system's business goals*.

Two tertiary studies [RMS18]; [JT22] and one study on architectural TD (ATD) [Ver+21] extended this model. Rios et al. added *TD types* (without naming them as such) and *TD management*, including TD *activities* and cost *measurements* for TD items [RMS18]. We present the extended model of Rios et al. in Figure 2.3, as the incorporated extensions are relevant to this thesis and further described in subsequent sections. The left two blue boxes of this Figure 2.3 represent the TD types, which are further outlined in Section 2.4, and the left blue box indicates the TDM with its respective TD activities as detailed in Section 2.6. Junior et al., similar to Rios et al., incorporated *TD types*, including the differentiation between *intentionally* and *unintentionally* incurred TD [JT22]. They further added the notion that consequences may be positive or negative, and some management activities they termed *actions*. Verdeccia et al. [Ver+21] created a similar model focusing on ATD. Compared to the original model, they included *person* and *communication* as instances, along with *ATD management strategies* similar to Rios et al. Although this model is designed for ATD, the new extensions, particularly the communication component, may be adopted across all TD types and constitute a focus of this thesis.

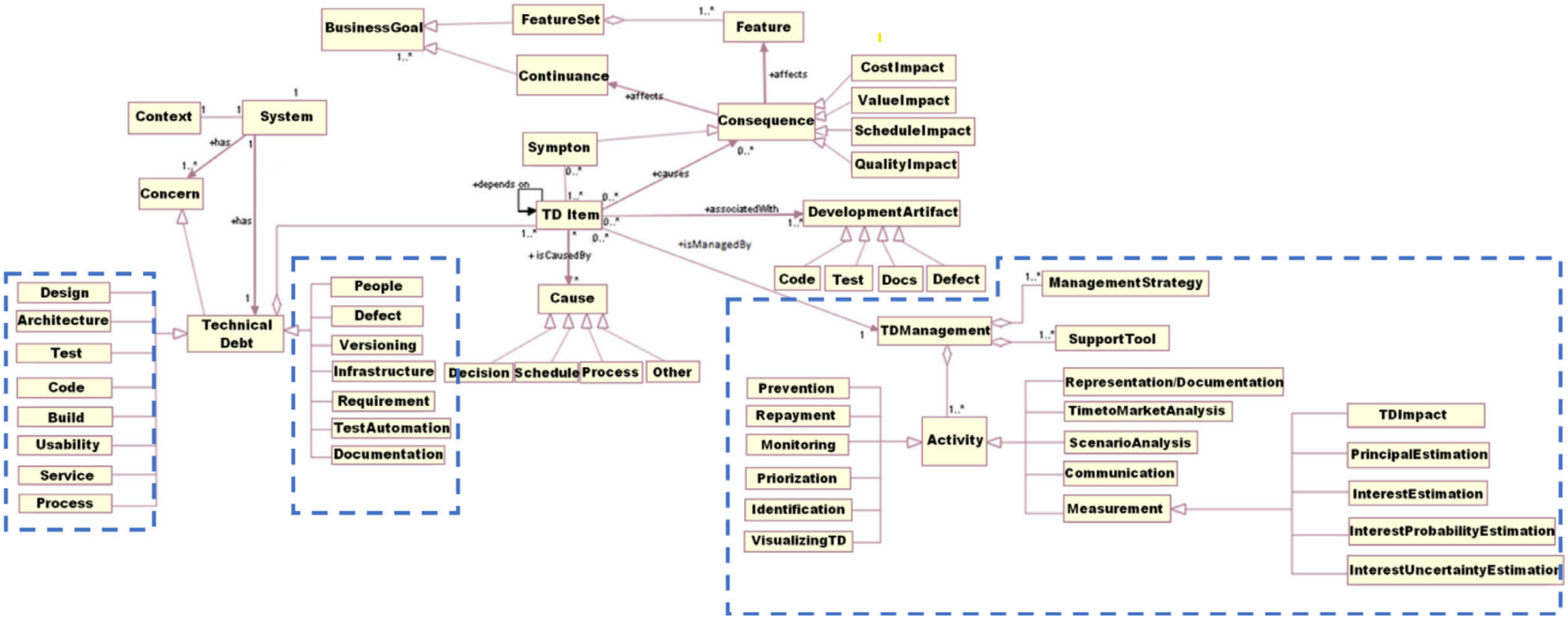


FIGURE 2.3: Conceptual Model of TD [RMS18]

The subsequent sections provide further information and examples of TD types, TD causes & consequences, and TD management, including the TD activities.

2.4 Technical Debt Types

Various studies describe different types of TD based on which aspects of software development are affected. In their tertiary studies, Rios et al. identified 15 distinct TD types as listed in Figure 2.3 [RMS18], while Junior et al. categorized these types according to the phases of software development: documentation, architecture, requirements, service, code, design, test, build, and versioning [JT22]. Junior further distinguishes between intentionally and unintentionally incurred TD items and refers to these as TD types [JT22]. However, while this distinction is common in the field, it is typically not referred to as TD type. In their book on TD, Ernst et al. structure the content according to TD type, i. e., requirements, design& architecture, implementation, testing, deployment, documentation, machine learning, and social debt [EKD21]. Additionally, new types continue to emerge, such as security debt [Ahm+21]. It is important to note that naming conventions for TD types are not always consistent across studies. For instance, Ernst et al. use "Implementation Debt" instead of the more common term "Code Debt", the term "Management and Social Debt" and Rios et al. use the term "People Debt", which is quite overlapping with "People Debt" defined by Rios et al. [LAL15]; [RMS18]; [EKD21].

Table 2.1 provides an overview of some of the most common TD types.

Type	Examples
Requirements	Important non-functional requirements not taken into account
Architecture	Outdated architectural patterns, such as shared databases
Design	Complex class structure; code smells
Code	Unreadable code; code duplication
Test	Lack of test environment; insufficient test coverage
Build/Automation	No automated deployment; unstructured deployment pipeline
Documentation	Missing or outdated documentation
Infrastructure	Outdated servers; operating systems not updated
Versioning	Unreadable code due to A/B testing or feature flags; lack of API versioning
Security	Passwords in plain text; unsecured interfaces
Social/People	Communication barriers due to a company's organizational structure; error-intolerant corporate culture

TABLE 2.1: Common TD types according to [LAL15]; [RMS18]; [EKD21]; [Ahm+21]

2.5 Technical Debt Causes & Consequences

As explained in Section 2.3, TD arises from various causes and triggers consequences (see Figure 2.4. Causes can be actions, such as implementing a workaround to accelerate development, or a lack of action, e. g., failing to update 3rd-party libraries or to improve a system's architecture as it evolves.



FIGURE 2.4: Causes and Consequences of TD

McConnell and Fowler distinguish between TD caused intentionally or unintentionally, i. e., unconsciously [McC08]; [Fow09]. Intentional TD items arise from a conscious decision, for example, accepting a workaround to meet a deadline. Unintentional TD items arise from missteps, such as poor code-level design or architectural decisions, that, over time, prove unfavorable.

Fowler provides a simple overview of the causes of TD in the form of a TD quadrant presented in Figure 2.5 [Fow09]. The quadrant distinguishes between TD that is deliberately

accepted (conscious/intentional) and that which is inadvertently incurred (unconscious/unintentional). It also differentiates between careless (reckless) and wise (prudent) decisions that can lead to TD and provides examples to understand each combination.

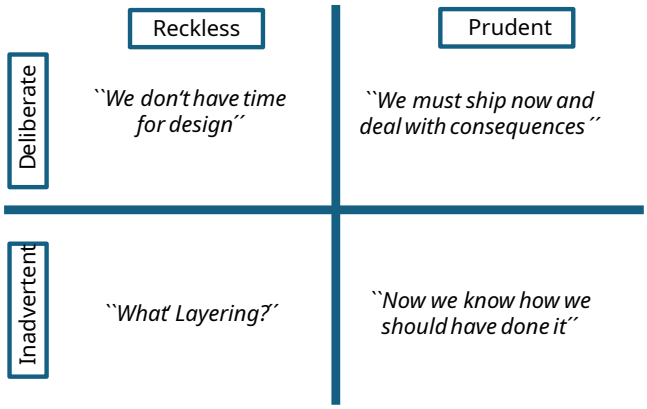


FIGURE 2.5: Causes of TD according to M. Fowler [Fow09]

The most intense research on causes and consequences of TD has been performed by the InsighTD project, in which TD aspects have been analyzed through a questionnaire completed by more than 500 practitioners from multiple countries, such as Chile, Colombia, Costa Rica, Serbia, and the United States [Rio+18b]; [Rio+19]; [Ram+21]; [Fre+21]. The research resulted in a unidirectional model of TD causes and consequences presented in the form of a fish bone diagram, as shown in Figures 2.6 to 2.7. Rios categorizes the causes into eight categories – Planning & Management, Methodology, People, External factors, Infrastructure, Organizational, and Lack of knowledge – and the consequences, which they call effects, into six categories – Quality, People, Maintenance, Organizational, Development Issues, and Planning & Management.

Notably, management-related causes dominate many lists of TD causes. Rios et al. found that in 27.1% of the TD occurrences, TD is incurred as a result of a cause from the "Planning and Management" category [Rio+19]. Ramač et al. conducted an extensive survey that identified 725 "Planning and Management"-cause occurrences, with the second-most-populated category being "Development Issues" (473 occurrences) [Ram+21] . Vogel-Heuser et al. found that 33% of the TD items they studied in the mechatronic domain were initiated by management decisions [VHB21].

Regarding the consequences, Schmid distinguishes between potential and effective TD [Sch13a]. Effective TD represents TD items that are currently creating interest, e. g., poorly readable code that must be frequently changed, and for which every change requires interest to be paid. Potential TD represents TD items that are rarely or never changed, e. g., poorly readable code in a legacy system that will soon be discontinued.

Kruchten et al. and the conceptual model of Avgeriou et al., as explained above, extend the unidirectional model of causes and effects by distinguishing between consequences and symptoms [KNO19]; [Avg+16a]. Consequences are interpreted as the direct results of the TD item, while symptoms represent the visible impact of the TD item. However, Stochel et al. identified that a *"smell may be considered as a symptom of a TD Item, rather than the TD Item itself."*, which contradicts this idea of visibility, at least on a business level [SCW20]. This illustrates that the distinction between consequences and symptoms is hard to capture.

In our publication on IT managers' perspectives on TD (publication No. 3 *(IT Managers' Perspective)*), we developed an additional model to extend the initial conceptual model because we were unable to reconcile the findings from our interview study with the current version [WB23]. Instead of distinguishing between consequences and symptoms, we added chains of causes and consequences, noting that both can be visible or invisible to business stakeholders. Additionally, we integrated vicious cycles, which have been described previously by Martini et al., but have not been completely included in the conceptual model [MB15].

As a result, the V4CTD model of **V**isibility, **C**ycles & **C**hains of **C**auses & **C**onsequences of **TD** provides a comprehensive overview, as stated by one of the article's reviewers: *"Despite*

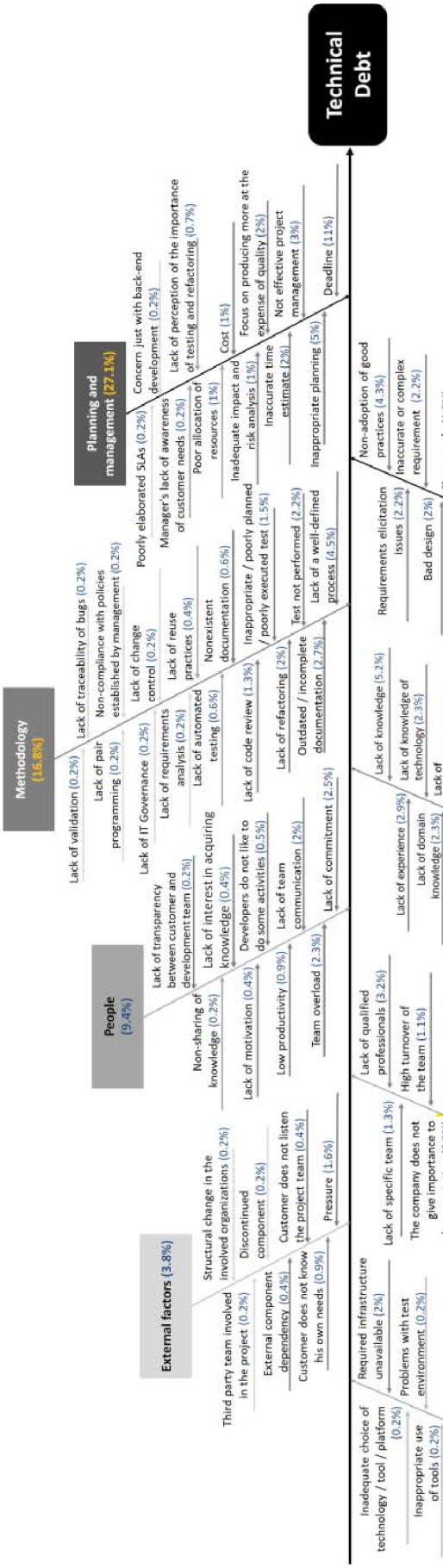


FIGURE 2.6: Causes of TD [Rio+19], p. 6

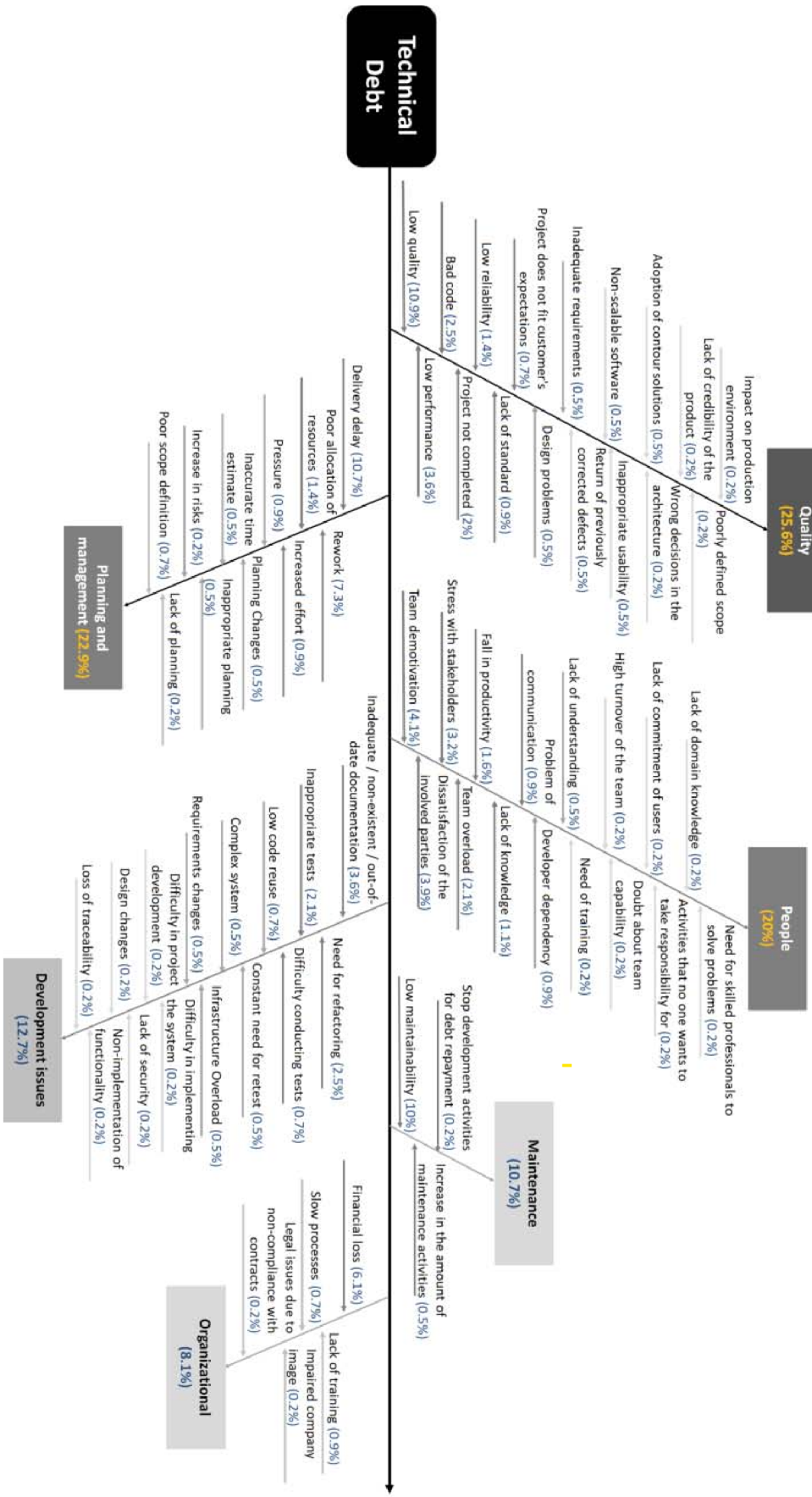


FIGURE 2.7: Consequences (Effects) of TD [Rio+19], p. 7

the concern on chains of causes and effects is not new, I think this was the first work that investigated it in such detail. The conceptual (V4CTD) model . . . is particularly useful to explain it." Our model integrates visibility aspects, categories of causes and consequences, and incorporates chains and vicious cycles into a comprehensive framework. Our study found that cause and consequence categories are often identical and form vicious cycles [WB23].

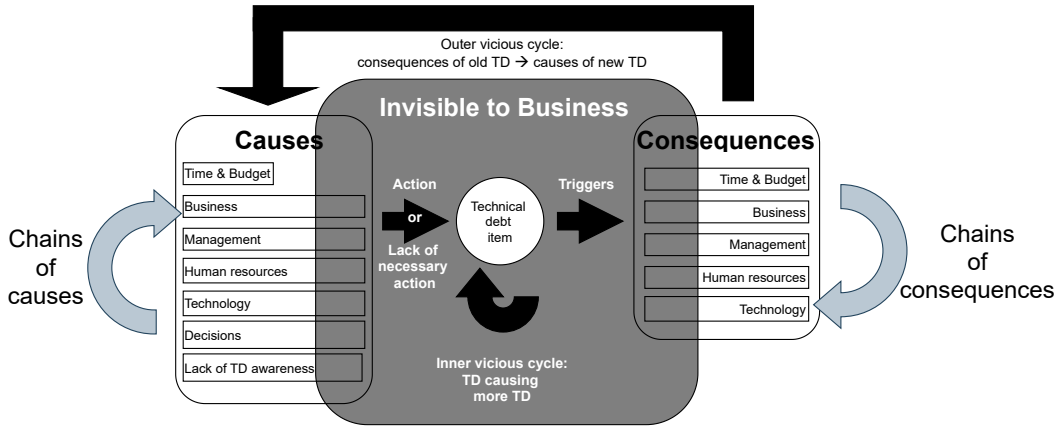


FIGURE 2.8: V4CTD model of Visibility, Cycles & Chains of Causes & Consequences of TD [WB23]

Figure 2.8 illustrates this model, highlighting the categories of TD causes and consequences (detailed in Tables 2.2 to 2.3) along with the following linking mechanisms:

- *Chains of causes:* TD causes can trigger other causes, e. g., a business needs to be first on the market, which leads to time pressure, which again may lead to a decision to implement a workaround rather than a suitable, long-term solution.
- *Chains of consequences:* TD consequences can trigger other consequences, e. g., TD might lead to more effort, which changes the estimated time for a task, which then can have effects on the project plan, and consequently on the business if a feature is delivered too late and customers demand compensation.
- *Visibility:* Business stakeholders can observe only the surface causes and consequences, i. e., the start of a cause chain and end of a consequence chain, not the inner linking mechanisms. They can "see" the time pressure developers work under, and they can "see" if the IT system is unreliable and has errors or outages, but they cannot combine the cause with the consequence. This may explain communication difficulties and misunderstandings.
- *Inner viscous cycles:* The TD items themselves may cause additional TD, e. g., if no test suite is installed and no unit tests are created, each new feature introduces new TD items because the number of missing unit tests increases.
- *Outer viscous cycles:* The consequences of one TD item may become a cause for incurring more TD, e. g., developers that are frustrated with the bad shape of their system, i. e., the growing amount of TD items, and the need to implement more and more workarounds, may leave the company, leading to a lack of knowledgeable developers, which can cause TD.

Overall, we found categories of TD causes and consequences that are quite similar to the findings from Rios et al. [Rio+19]. However, due to our interview partners, our list is less comprehensive regarding technical or development-related causes and consequences. It places greater emphasis on other areas, such as business-related causes. In the contributions in Section 6.1.2, we will discuss the differences in more detail.

2.6 Technical Debt Management

TD Management (TDM) has been intensively researched over the last decade [LAL15]; [RMS18]; [JT22]. In recent years, this has led to the establishment of comprehensive management

Category	Explanation/Examples
Time-related	Incurring TD by enforcing short-cuts in the development
Budget-related	Funding-related causes, like cutting a project's budget, license costs, or the preference for cheap, but less reliable technologies
Management	Social or organizational issue, e. g., complicated communication across teams
Technological	Not accounting for the evolution of technologies and growing knowledge, e. g., regarding best practices
Human resources	Lack of IT specialists, absence of specialized or knowledgeable developers, and a high turnover rate
(Lack of) Awareness	Decision-makers being unaware of TD or of the consequences a TD item might have are more prone to incur TD
(Faulty) Decisions	Conscious rational decisions, as well as mistakes, suboptimal, or reckless decisions. Developers make mistakes or write difficult-to-read code. Architects make decisions without comparing options. Business managers set a deadline without considering the required time and effort

TABLE 2.2: Categories of TD causes [WB23]

Category	Explanation/Examples
Time-related	Time overhead needed to add new features or fix errors caused by TD items
Budget-related	The budget for repaying TD, e. g., by a rewrite. Projects may become more expensive, e. g., if additional licenses are required due to performance issues that are not addressed through code optimization
Business	Business expectations about features that were not met, or losing the customers' trust
Management	Management of projects or project pipelines, and the risk TD poses to IT management, e. g., if the project timeline cannot be adhered to because previously unaccounted for TD items must be addressed before further implementation can proceed
Technological	Using outdated technology and architectural erosion may lead to maintenance problems, unstable systems, difficulties adding new features, or performance issues
Human resources	Influences on the development team (developer pain), e. g., when developers need to work overtime or leave the company

TABLE 2.3: Categories of TD consequences [WB23]

guidelines [KNO19]; [EKD21] accessible to software practitioners. TDM generally refers to the TD activities outlined in Table 2.4 and characterized in Section 2.6.1. Furthermore, several maturity models for the adoption of TDM processes have been developed and are presented in Section 2.6.2.

Type	Examples
Prevention	Avoiding TD items before they are incurred,e. g., by making more conscious and intentional decisions and preventing carelessness
Identification	Locating TD items, e. g., through static code analysis or based on the developers' knowledge
Measurement	Evaluating TD items and determining or estimating quantifiable characteristics, e. g., principal and interest
Documentation	Recording TD items with relevant information (e. g., description, list of subtasks for repayment, assessment of consequences, and cost factors) – for example, in an issue-tracking tool – in order to gain an overview of TD items
Prioritization	Ranking and prioritizing TD items in relation to other TD elements, or in comparison to functional requirements, based on the assessed measures and documentation
Repayment	Methods for structuring and sensibly organizing the removal of TD items, for example, through refactorings, but also by the discontinuation of debt-ridden systems or system parts
Monitoring	Observing how the entirety of TD items develop over time, using the recorded and measured attributes to evaluate the system's health
Visualization	Visualizing important TD attributes for monitoring or prioritization purposes

TABLE 2.4: TD activities according to [LAL15]; [RMS18]

2.6.1 Technical Debt Activities

The following paragraphs provide a detailed description of each of the TD activities presented in Table 2.4.

TD prevention TD prevention is the activity aimed at avoiding the incurrence of TD. The InsighTD project identified various TD prevention strategies used in the industry [Rio+20]; [Fre+20b]; [Pér+21], i.e.:

- Well-defined scope/requirements
- Code evaluation/standardization
- Following well-defined project processes
- Well-defined architecture/design
- TD awareness/management
- Adoption of good practices
- Improving tests/coverage
- Good communication
- Reviews

However, terms like *well-defined* or *good* allow for ambiguous interpretations and are not defined in detail. Besker et al. compared encouraging, rewarding, forcing, and penalizing strategies for TD reduction and found that some companies used educational sessions to prevent TD [BMB20]. In their research on mechatronic systems, Vogel-Heuser et al. recommended the use of guidelines to prevent TD [VHB21].

In addition, the TAP (**T**echnical Debt **A**ware **P**roject Management) framework, detailed in Section 6.2.1, provides a method to avoid TD by incorporating a TDM approach into project management. This framework holds the project team responsible for repaying the TD incurred during their project – after reaching a deadline, but before ending the project phase. This approach creates a feedback loop in team behavior in which the project team only incurs TD after rational discussions and as informed decisions [Wie+22] (publications No. 1 *(TAP Framework)* and No. 2 *(TAP Framework – extended)*).

Finally, Borowa et al. introduced debiasing techniques that prevent cognitive biases and, as such, may avoid TD incurrence in a system's architecture [BZK21]. These techniques comprise:

- Double-checking and challenging all decisions
- Developing an environment based on trust
- Gathering information about alternatives
- Using procedures to reduce low-quality, reckless changes
- Documenting and passing on knowledge
- Registering all accounts of TD
- Regularly checking for new TD and re-evaluating old TD
- Defining and recording project responsibilities

To make these recommendations more feasible, they developed a debiasing workshop for practitioners that focuses on three techniques: explicitly searching for at least one solution alternative, at least one drawback per alternative, and at least one risk per alternative. First, they tested the workshop with IT students [Bor+22], followed by a second study (contributing to this thesis) in which they tested it with practitioners [BAW25]. Applying these techniques, they showed that the optimism bias, anchoring effect, and confirmation bias could be counteracted [Bor+22]; [BAW25].

TD identification TD identification comprises methods to detect TD, such as (1) static code analysis, e. g., SonarCloud [Son25], (2) analyzing self-admitted TD (SATD), e. g., in "todo" tags in code comments [LSA22], or (3) recording TD in a TD backlog, e. g., as an issue in an issue-tracking system [KNO19]; [JT22]. Although automatic TD detection has been extensively studied, automation tools often have limitations in the types of TD items they identify, e. g., they may identify only code debt rather than architecture debt [Avg+23]; [Avg+25]. Thus, manual TD identification is still frequently used, e. g., [OGS15].

During the course of this thesis, we found that practitioners struggle to distinguish TD issues from other issues – a topic that also falls within this activity. Using the TD definitions and explanations outlined in Section 2.2 has been insufficient. The definition of Avgeriou

et al. [Avg+16a] focuses too much on intentionally incurred TD. Using the *colors of the backlog* [KNO12] is based on the customer or user visibility, which is a factor that is not applicable when a team creates central elements of a system and has no direct customer or user, but instead supports other teams. In their vision paper, Averiou et al. already pinpointed that asking whether interest is incurred might be a more practical strategy [Avg+23]

To support the practitioners, we developed a simple decision matrix as shown in Figure 2.9 as part of the *Practical Guide to Establishing Technical Debt Management*, which is an essential element of the additional material of publication No. 7 *(Practical Guide)* [Wie26]. The decision matrix is based on two essential and usually simple-to-answer questions to make the decision as easy as possible, even if there might be exceptions to this rule, i. e., matrix outcome:

1. Does the item incur interest? Or put more bluntly for practitioners: Who suffers from not addressing the issue?
2. Who is willing to pay for addressing the issue?

Only if the team is suffering from a recorded issue and no one is willing to pay for its removal without a detailed explanation and negotiation, is it considered a TD issue.

		Who suffers?	
		Your Customer	You
Who is willing to pay?	Your Customer	No TD	(Probably no) TD
	No one	No TD	TD

FIGURE 2.9: Matrix for Identifying TD Items [WSB26], Chapter A

Further challenges and improvement ideas to identify a TD issue in a backlog are outlined in the contributions in Section 6.1.7.2 and Section 6.2.6.

TD measurement This activity aims to quantify TD, e. g., by estimating interest and principal costs [Guo+11]; [TAV13]; [McC08]; [LAL15]. Notably, in the IT domain, the largest cost factor is developer time, so most measures are expressed as time estimates, i. e., the effort required to perform a task [Guo+11]. Static analysis tools such as SonarQube [Son25] aim to measure TD and provide related measurements, e. g., remediation time. However, those measurements are limited to code debt and have proven to be mostly inaccurate [Saa+19]; [Avg+21]; [Bal+20]; [Kat+22].

In their Mapping study, Li et al. identified the first five cost factors presented in Table 2.5 with an explanation and an example. The sixth factor – contagiousness – was later identified by Martini et al. [MB15]. In their systematic mapping study on quantification approaches, Perera et al. found that most research focuses on cost aspects, and benefits are rarely analyzed [Per+24]. Moreover, they found that research focuses on the cost retrospectively, i. e., on evaluating TD items after they are incurred, instead of using cost estimations to guide decisions on incurring TD [Per+24]. Similarly, in our case study analyzing the TAP framework, we identified that the effort needed for the debt-incurring solution should be compared to a non-debt-incurring solution, which is important information to make rational decisions and potentially avoid TD [Wie+22]; [WRS21]. Table 2.5 lists these as metrics seven and eight.

TD documentation TD documentation refers to the recording of TD items and relevant TD attributes in lists. These attributes include cost estimates derived from measurements, as well as textual evaluations, e. g., a description of the TD item and its repayment strategy [KNO19]; [Yan+19]; [Ton+17]. Yang et al. propose the following attributes in their work on commercial-off-the-shelf (COTS) systems [Yan+19]:

metric	Explanation	Examples
(1) Principal	The cost of resolving the TD item	3 days to restructure a complex class structure, e. g., by adding a new module, including testing and documenting the changes
(2) Interest	The extra effort needed if the TD item is not repaid	Half a day of additional effort to understand the complex structure before changing it
(3) Interest probability	The probability that a TD item will have actual consequences and thus interest accrues	The complex structure is in a central module, which is changed once every 3 months
(4) Interest rate	The relation of the interest to the principal	3 days/0.5 days
(5) (Overall) Cost	Principal + Interest	The result depends on the interest probability and the overall time frame of the calculation, e. g., cost in this year = 3 days + 4 * 0.5 days = 5 days
(6) Contagiousness	The increase of the principal over time	An API with TD initially has only one consumer, which must be changed along with the API. Over time, additional consumers are added, all of whom must also be updated if the TD item is repaid later.
(7) Effort debt-incurring	Estimated effort needed for the debt-incurring solution	Ignoring layering, a graphical user interface (GUI) directly acquires data from a database, skipping the business layer (e. g., half a day effort)
(8) Effort non-debt-incurring	Estimated effort needed for a non-debt-incurring solution	Following layering, a GUI communicates with the business layer, which receives data from the database (e. g., 1.5 days effort)

TABLE 2.5: TD metrics, i. e., cost factors according to [LAL15]; [MB15]

- General information, such as Id, Name, Open/Close date
- Location
- Type
- Description
- Cost factors, i. e., Principal, Interest amount, Interest probability, Contagion
- Propagation rule
- Context
- Mitigation strategy
- Accountable party and Intentionality

Tonin et al. suggested fewer but similar attributes as a result of a classroom study: Name, Date, Responsible, Type, Location, Description, Principal, Interest amount, Interest probability [Ton+17].

TD items can be implemented as an additional issue type within an issue-tracking system to augment the development backlog [KNO19]. In issue-tracking systems, work items are commonly referred to as issues. Accordingly, one TD item is represented by one TD issue. However, even a list in an Excel might be sufficient to create a TD backlog. TD backlogs and lists lead to more visibility, make TD explicit, raise the perception of TD, and by this, raise the overall awareness for TD [KNO12]; [Eli+15]; [GSS16]; [BMB19].

TD prioritization TD prioritization involves ranking TD items based on their significance. Various factors that affect the priority of a TD item have been recognized in academic research, such as prioritization

- dependent on the system's evolution, i. e., prioritizing actual over potential TD [Sch13a]
- by business value of the affected system component [Reb+18]
- by costs as calculated by TD measurement [TAV13]
- by the relevance of a negatively affected quality attributes [RR22]

In their systematic literature review on TD prioritization, Lenarduzzi et al. uncovered 53 prioritization factors organized into the following categories [Len+20]:

- Business, e. g., competitive advantage or business value

- Customer, e.g, customer satisfaction or expectations
- Evolution, e. g., impact on critical or upcoming features
- Maintenance, e.g, number of bugs or maintenance cost
- System qualities, e. g., security, scalability, or performance
- Quality debt, e. g., co-occurrence of issues and errors
- Productivity, e. g., wasted time or coding outlet
- Project factors, e. g., availability of resources or project size and complexity
- Social factors, e. g., developers' morals, team culture
- Other factors, e. g., system's history, number of affected users, future costs
- Not specified, e. g., risk, or severity

While prioritizing TD can be a one-off task, it is frequently an ongoing process, as the priority of a TD item may vary depending on the present context [Len+20]. For example, if a new feature is planned, reanalyzing TD items within the same component as the new feature may reveal items that pose risks to the implementation, thereby increasing their priority.

TD can be prioritized in relation to functional requirements or in comparison with other TD items [Len+20]. Typically, TD items are evaluated against each other to define a repayment strategy (see next paragraph). Lenarduzzi et al. found no research that has analyzed how TD items can be evaluated against functional requirements, e. g., by demonstrating their direct value to the business [Len+20].

TD repayment Following the financial metaphor, repayment refers to resolving the TD item [Cun92]. Buschmann et al. introduced three categories for resolving TD: *ignore*, *refactor*, and *rewrite* [Bus11].

The *ignore* category involves addressing TD only when it blocks a functional requirement [Fre+23] or simply paying the interest on the debt if the interest payment is low [Bus11].

The *rewrite* category entails waiting for a software system to be discontinued or migrated. With discontinuation or migration, the TD items associated with the system are automatically resolved [Sch13b]; [Sch13a].

Refactoring includes actively resolving TD items for which there are four different approaches. First, repaying the TD item, e. g., through code refactoring, is reasonable if the cost-analysis proves the positive effect [McC08]; [TAV13]. Second, some teams chose to assign a fixed repayment quota, e. g., 10% of each sprint, or assign every fifth sprint to TD repayment [McC08]; [Wie+22]. Third, TD may be repaid if they become actual TD, hindering the implementation of a project or feature, i. e., as the system evolves [Sch13a]. Fourth, after a project or feature implementation, TD items may be repaid if they were incurred due to an implementation deadline, as we introduced in our work on the TAP framework [Wie+22] (publications No. 1 *(TAP Framework)* and No. 2 *(TAP Framework – extended)*).

TD monitoring & visualization Monitoring concerns observing a system's TD development over time to assess its health [LAL15]. Visualizations of TD development based on lists of TD items support monitoring TD, but can also be used for other purposes, e. g., guiding the repayment strategy. Various tools for TD visualization have been developed, e.g.:

- *DebtFlag* acquires and visualizes SATD in a colored form of a class diagram, thereby presenting the location and the amount of TD items [HL13].
- *MultiDimEr* analyzes and visualizes the number of bug reports across a system's releases to identify emerging TD at an early stage [SUW22].
- *VisminerTD* combines TDM activities such as identifying and monitoring TD to visualize the evolution of the amount of TD items in a system [Men+19].
- *MIND* quantifies and visualizes the interest associated with TD [FR15].
- *TEDMA* supports TDM by capturing static analysis metrics over the course of software evolution [FS+17].

Most visualizations are proprietary or rely on proprietary tools. They often focus on (automatically detected) code debt or, in the case of MultiDimEr, even on bugs as symptoms of TD. The related papers offer no guidance for transferring the visualizations to other contexts.

2.6.2 Technical Debt Management Maturity Models

Maturity models assess an organization's TDM based on certain characteristics of the current approach. Yli-Huumo et al. [YHMS16] describe three levels of organizational maturity as presented in Table 2.6.

Type	Examples
Unorganized	The team does not invest effort in managing TD, or only with a very limited focus.
Received	The team has acknowledged the need for a certain TDM activity, and a few people address TD on some level, but there is no iterative process.
Organized	The team recognized that an iterative TDM process is an essential element of software development.

TABLE 2.6: TDM maturity Model according to Yli-Huumo et al.[YHMS16]

Martini et al. [MBB18] describe seven levels of adoption for TD tracking, displayed in Table 2.7.

Type	Examples
Unaware	There is no awareness of TD.
No tracking	The TD metaphor is known by practitioners, but not in the company.
Ad-hoc	There is some management of TD on a personal (or team) level, but without allocated time or proper tools.
Systematic	Teams document TD, and see this as an iterative process with budget/time allocated to it.
Measured	TD is automatically identified with automated indicators, and measurements
Institutionalized	There are standardized TDM processes across the company, including TD prioritization.
Fully automated	Automated decisions are made based on statistical and historical data of the system.

TABLE 2.7: TDM maturity model according to Matrini et al. [MBB18]

Kruchten et al. [KNO19] distinguish five awareness levels, which are similar to a maturity model and are presented in Table 2.8.

Type	Examples
Level 1	No one has any awareness of the term TD at all, and no one does understand the problem.
Level 2	Some people have heard of or read about the TD concepts.
Level 3	Some people know their own system's TD, but there is no acceptance from the management or business.
Level 4	There are team-level strategies to make TD visible and manage it.
Level 5	The company has complete control over TD.

TABLE 2.8: TDM maturity model Kruchten et al. [KNO19]

All three models show similarities, e. g., they start with not knowing what TD even is [YHMS16]; [MBB18]; [KNO19], the importance of recognizing TDM as an iterative process [YHMS16]; [MBB18], or distinguishing between a TDM process on a personal or team level versus a company-wide strategy [MBB18]; [KNO19]. In contrast to the other two, the model of Yli-Huumo et al. [YHMS16] keeps a team-level focus. The model of Martini et al. [MBB18] differs in that it places significant emphasis on automating a TDM process, which is unrealistic, though desirable, given that no tooling for such automation exists yet.

Full maturity in TDM is not yet widespread in industrial settings. This can be clearly observed by the results of this study, as only 7.2% of respondents reported having a systematic approach to TD tracking [MBB18]. Similarly, in our study on IT managers' perspectives on TD, we found that only 1 of 19 managers had established a TDM approach [WB23]. Finally, both tertiary studies on TDM show that a holistic view, including all TD activities and TD types, has not been researched so far [RMS18]; [JT22].

Chapter 3

Related Work and Research Gaps

This section presents the related work to this thesis and outlines respective research gaps, structured by the topics of the research questions: TD communication in Section 3.1, TD Awareness in Section 3.2, TD Prevention in Section 3.3, and TDM in Section 3.4. Each section comprises a "Related Work and a "Research Gaps" subsection. In the related work sections, we present studies similar to ours in terms of their goals and research approaches. In the research gaps section, we present works that focus on the respective research topic and either directly pose open research questions or show that specific aspects have been overlooked by previous studies, e. g., when no studies focus on these aspects. These sections highlight areas requiring further investigation and include questions that address them. In Chapter 7, we outline how this thesis's contributions provided new insights to address these research gaps. Notably, the research gaps presented may stem from the studies that contributed to Step I of this thesis.

3.1 Technical Debt Communication

3.1.1 Related Work

Various studies suggest that communication about TD is an important factor in TDM. Even the initial rationale behind the metaphor was to communicate the TD concepts and related issues [Cun92]. Klaus Schmid outlined the boundaries of the metaphor in a theoretical analysis [Sch13b]. Most TD communication research focuses on communication with business stakeholders [KNO12]; [TAV13]; [YHMS16]; [Reb19].

As shown by Kruchten et al., TD effects may be visible (symptoms) and invisible (consequences) for business stakeholders. The invisibility of the consequences of TD items for business stakeholders makes communication difficult [KNO12].

Tom et al. state that poor communication is a cause of inadvertent TD. In their study, they also identified the low visibility of TD as a major impediment to communication and suggested increasing its visibility, for example, by incorporating it into daily stand-up meetings or burn-down charts [TAV13].

Yli-Huumo et al. identified that most communication on TD does not transfer to the business stakeholders, which means that the time required for TD repayment is not allocated. He suggests communicating about TD in various meetings, making it a constant topic. Moreover, product owners should enable communication between technical and non-technical stakeholders [YHMS16].

One method to enhance communication with business stakeholders is to engage them in the TDM process, as proposed by de Almeida et al., which integrates both business and technical stakeholders in business-oriented TD prioritization [Reb+18]; [Reb19]; [Reb+21].

Verdecchia et al. highlight communication as one of the core categories influencing ATD and propose three measures to mitigate the communication issues [Ver+21]: (1) investing time to make maintenance activities clear to stakeholders, (2) creating the perception that the system is evolving, even when development is primarily focused on refactoring, and (3) intentionally emphasizing the problems of a system to motivate the need for significant refactorings.

In contrast to the communication directed towards business managers, Besker et al. analyzed the effect of communication coming from IT managers and showed how their encouragement to keep TD levels low can significantly impact the amount of TD in a system [BMB20].

3.1.2 Research Gaps

First, the original purpose of the metaphor was to support communication [Cun92]. However, the metaphor seems to be flawed from a theoretical perspective [Sch13b]. Furthermore, TD-communication is still a critical issue despite the metaphor's invention and use [KNO12]; [YHMS16]; [Reb+18]; [WB23]. This raises questions about the metaphor's relevance. We need to assess whether the metaphor is really used for communication in practice. If it is not used, the logical follow-up question is whether and how it can be improved **(C1)**.

Second, Merriam-Webster dictionary defines communication as *"the act or process of using words, sounds, signs, or behavior to express or exchange information or to express thoughts, feelings, etc., to another person"* [Mer]. This indicates that communication is about exchanging information, and there is a wide variety of information about TD, e. g., measurements [McC08]; [TAV13]; [LAL15], on causes and consequences [Rio+18b], prevention methods [Pér+21], and even maturity models [MBB18]. However, no work analyzes which of this wide variety of information is relevant to communicate about to manage TD **(C2)**.

Third, we identified many studies that focus on communicating TD-related issues to business stakeholders [KNO12]; [YHMS16]; [Reb+18], which was the metaphor's original goal. Only T. Besker conducted a study on communication from the management to developers [BMB20], and few works focused on communicating information to educate students on TD topics [Gan14]; [Ton+17] **(C3)**. However, research on other communication partners, i. e., communication between other stakeholders than business stakeholders or other communication directions, might also be relevant but is rare.

Fourth, in our own study, we found that TD communication remains an important issue in practice, i. e., for IT-managers [WB23]. This raises the need to identify why the communication methods identified and presented by research [Reb+18]; [BMB20] are not transferred. Possible reasons may be that they are too ambiguous and not concrete enough, or too context-specific to be applicable. After identifying the reasons, the follow-up question is how to change this **(C4)**.

Fifth, communication about TD entails addressing a critical topic amid conflicting stakeholder concerns [BMB20]; [Wie+22], e. g., such communication might be misinterpreted as an admission of error. The Dagstuhl manifesto emphasizes the importance of psychological safety and trust between technical and business stakeholders ([Avg+25], Value 1), but research on these aspects is rare to date, as only T. Besker focused some of her work on the psychological aspects that management activities might have on developers [BMB20]. This introduces further questions about which psychological effects regarding a TDM process exist and how to address them, e. g., how to create a psychologically safe environment **(C5)**.

Finally, one of our Step I studies showed that a TDM process optimizes communication [Wie+22], raising questions about the interplay between TDM and communication and whether the identified positive effects can be replicated. Furthermore, it is interesting to know for which activities in a TDM process communication is relevant (e. g., needed or possible) and which elements of a TDM process trigger communication **(C6)**.

Research Gaps:

C1. *Metaphor:* Is the metaphor used for communication in practice? Can the metaphor or its usage be improved?
C2. *Information:* Which information related to TD is suitable and relevant to communicate about?
C3. *Communication Partners:* Which (other) communication partners and directions may be relevant?
C4. *Transferability:* Why are the identified methods not transferable? For example, are they too context-specific, or too ambiguous to be applicable? How can we change this?
C5. *Psychology:* What are the potential psychological effects associated with a TDM process? How can we address those?
C6. *TDM Process and Communication:* What are the interchanges between a TDM process and communication? For which activities of a TDM process is communication relevant, e. g., needed or possible? Which elements of a TDM process trigger communication?

3.2 Technical Debt Awareness

3.2.1 Related Work

In 2012, Kruchten et al. already mentioned the importance of TD awareness: *"The first step is awareness: identifying debt and its causes"* [KNO12]. In a survey with more than 1,800 participants, Ernst et al. found that 79% of the participants strongly agree that *"Lack of awareness of TD is a problem"* [Ern+15]. Multiple other papers state that a lack of awareness poses a problem [TS10]; [Guo+11]; [Mor+12]; [TAV13]; [Ern+15]; [SSS15]; [MB15]; [VHN17]; [Fon+17]; [BMB19]; [Apa+20]; [MTR20]; [VHB21]; [Mor+21]. Finally, some papers also mentioned TD awareness as an initial requirement for TDM [Bes+17]; [Hol+18]; [KNO19].

Unfortunately, the term "TD awareness" is used ambiguously, which may lead to misinterpretation and potentially to misunderstandings. As of today, there is no consensus on how to define and measure TD awareness. Studies interpreted it as:

- Knowing what TD is [Apa+20]; [Bes+17]; [Hol+18]; [BM19]; [Zam+21]
- Knowing the TD items in ones own system [KNO12]; [Bes+17]; [Liu+20]
- Not inadvertently incurring TD [RZN17]; [Wie+22]

Few papers study how to raise awareness. Some of them have found that visualization helps raise awareness of TD. Others used gamification or studied how communication influences awareness.

To visualize TD to raise TD awareness, Kaiser et al. presented the "Code Christmas Tree", a visual representation of a system's code coverage. This visualization was displayed in the hallway and prompted discussions. As a result, the visualization raised awareness among different stakeholders [KR11]. Baysal et al. and Treude et al. examined the positive effects of dashboards of issues on developers' awareness in their projects [TS10]; [BHG13]. Eliasson et al. proposed a method to visualize dependencies in the automotive domain to make TD visible. They found their method useful for raising stakeholders' awareness of ATD [Eli+15]. Gupta et al. found that so-called "information radiators", i. e., visualizations displayed where people can frequently see them, were an effective way to raise awareness [Gup+16]. Guo et al. presented a TD management approach that included a list of TD (i. e., a form of visualization), thereby increasing awareness of TD. In this case study, the project manager emphasized the positive effect of the TD list [GSS16].

Two studies used gamification to raise awareness for TD. Al Mamun et al. investigated TD in miniature self-driving cars. They focused on code debt and proposed that tools detecting rule violations would also increase awareness [ABH14]. Baars et al. used a gamification

approach to make developers aware of harmful patterns in their code base by turning the patterns into monsters in Minecraft[1] [BM19].

Just communicating about TD might already raise awareness. Shrama et al. found that awareness of the benefits of refactoring was lacking. They reported on a strategy to conduct workshops on the benefits of refactorings and refactoring tools to raise awareness [SSS15]. Letouzey et al. also mentioned an awareness workshop for top managers to introduce TD and the SQUALE [Squ] method to them [LI12].

Finally, some publications claimed that the introduction of a TDM process increased awareness. Martini et al. proposed a method, AnaConDebt, to determine whether and when Architecture TD should be refactored. They found this method is useful to raise awareness of proactively refactoring ATD [MB16]. Tonin et al. [Ton+17] researched the effects of TD awareness by conducting a classroom study. The students were informed about the concept of TD and were instructed to use TD boards during the project. The effects of this were a changed attitude, more discussions, and more conscious incurrence of TD.

3.2.2 Research Gaps

As shown above, we found multiple ambiguities in the definition of TD awareness [Apa+20]; [Liu+20]; [RZN17]. Each publication may use a different definition, but to ensure comparability, researchers should define their understanding of TD awareness before discussing it. In this regard, it is also critical to analyze which definition is helpful for the respective research goal. For example, to assess whether participants are aware of their TD items, a static analysis tool might be helpful. However, to analyze whether TD is inadvertently incurred due to a lack of awareness, another approach is needed **(A1)**.

To evaluate TD awareness, researchers commonly use questionnaires [Ber+21]; [Wie+22], which have the disadvantage of being a subjective and retrospective measure open to biases. As mentioned above, others inferred TD awareness from indicators derived by static analysis tools [Cre+22], which provide only indirect evidence of TD awareness. What is more, many studies, including some of the ones listed before, claim to raise awareness but do not measure it and, thus, fail to prove this claim, e. g., [Eli+15]; [Ock+19]; [BM19]; [Bal+20]; [STAF21]; [RCM22]. Consequently, we need to identify approaches for measuring TD awareness based on a given definition. These measurement approaches should include the definition of metrics and data-gathering methods. **(A2)**

Some studies have claimed that TD awareness is essential to the TDM process [Bes+17]; [KNO19]. Others claim that a TDM process improves awareness [KR11]; [GSS16]; [MB16]. This diversity raises the question of how much, and through which measures (activities), a TDM process influences TD awareness. Given the lack of substantial evidence for TD awareness in these studies, we should analyze whether we can support these claims with metrics **(A3)**.

Research Gaps:

A1. *Definition:* How could and should TD awareness be defined? Which definition is helpful in research?

A2. *Measuring Effects:* How can TD awareness be measured, given a previously clarified definition? What are valid metrics, and how can those be gathered?

A3. *TDM Process and Awareness:* How much and through which measures does a TDM process influence the TD awareness? Can we support these claims with metrics?

3.3 Technical Debt Prevention

3.3.1 Related Work

Similar to TD awareness, TD Prevention is mainly mentioned in research as a problem [He+16]; [Apo+16]; [SWA17]; [Fon+17]; [Rio+18a]; [Dig+20]; [Fre+20a]; [Tso+20]; [VHB21];

[1] https://www.minecraft.net/

[Rio+21]. In a systematic literature review from 2015, Li et al. [LAL15] found TD prevention to be understudied. Subsequent research on how to prevent TD beyond raising TD awareness has been conducted. Many studies on prevention strategies focus on inadvertent code debt and on tools that help identify these debts, e. g., avoiding code smells through the support of an IDE (Integrated Development Environment) [Sne14]; [YHMS16]; [Apa+20]; [Tso+20]; [Dig+20].

Papers focusing on the prevention of TD at the design and architectural level, which are mostly incurred during decision-making, are fewer. Morgenthaler et al. shed light on how Google manages TD, outlining their prevention methods under three categories: "Automation", "Make it easy to do the right thing", and "Make it hard to do the wrong thing" [Mor+12]. Yli-Huumo et al. discovered that developing minimum viable products can aid in minimizing TD by concentrating on prevention in areas significant to customers while discarding elements that lack genuine business value [YH+15]. The focus of Charalampidou et al.'s research was on avoiding documentation debt during the requirements engineering phase. They incorporated requirements specifications directly into an IDE [Cha+18]. Lenarduzzi et al. recommended improving requirements and obtaining early customer feedback, which prevents TD through enhanced effort estimation and improved quality via better testing and documentation practices [Len+19]. Yang et al. advised that for systems that rely on commercial-off-the-shelf (COTS) components, it is beneficial to train the assessment team and explore contracting options with COTS suppliers [Yan+19]. In their analysis of strategies such as encouragement, rewards, enforcement, and penalties to reduce TD, Besker et al. found that some organizations offered educational sessions on avoiding TD [BMB20]. The InsighTD project conducted an international survey study, where participants frequently indicated that their TD could have been avoided [Rio+18a]; [Rio+20]; [Rio+21]. Various papers in this study examined the causes of TD and the strategies practitioners employ to prevent it, which are listed in Section 2.6.1. Finally, Borowa et al. introduced strategies to mitigate cognitive biases that can lead to incurring ATD, as detailed in Section 2.6.1 [BZK21].

3.3.2 Research Gaps

Multiple studies, particularly from the InsighTD project, collected data on how TD is currently prevented in practice [Rio+20]; [Rio+21]. However, many TD items are still inadvertently incurred, raising the topic of how to prevent those. Research should focus on finding further strategies for TD prevention beyond those already adopted by practitioners **(P1)**.

The research findings are largely vague, e. g., train stakeholders [Yan+19], good communication and well-defined process [Rio+18a], make the right things easy [Mor+12]. Concrete instructions for achieving them are missing. Consequently, research should focus on operationalizing these research findings by giving concrete advice and instructions to practitioners **(P2)**.

Many research studies note that time pressure is one of the most relevant factors for TD incurrence [MBC14]; [Ern+15]; [Rio+20]; [Ver+21]; [Fre+21]; [Sol+21a]. Thus, special focus should be placed on how TD intentionally incurred due to time pressure from (project) deadlines can be prevented **(P3)**.

The success of TDM and TD prevention strategies is often measured by the number of issues detected by an issue-tracking tool [BM19]; [DMR22]; [Kat+22]; [Cre+22], thereby limiting evaluation to code debt. Another way is to analyze the number of TD items in a backlog, which is often used in timeline visualizations, e. g., [San+13]. However, a decrease in TD items may also be due to non-recording, rather than repayment activities, suggesting a poor rather than a satisfactory TDM process. Research still lacks a data-driven assessment of the success of TDM and TD prevention strategies. Consequently, it is essential to find a method to assess how many TD items have been prevented **(P4)**.

Revisiting the topic of TD communication from the previous section, it seems vital to analyze how prevention and communication might interplay. However, no research has examined which communication topics might foster TD prevention, or which stakeholders should communicate with one another in which situations to prevent TD **(P5)**.

Research Gaps:

P1. *Strategies:* How can we prevent inadvertently incurred TD items? Are there other strategies for TD prevention beyond those already adopted by practitioners?
P2. *Operationalization:* What are concrete measures that can be utilized by practitioners?
P3. *Time Pressure:* How can TD be prevented under time pressure from (project) deadlines when there is not enough time to implement the debt-free solution?
P4. *Measuring Effects:* How can we assess the success of prevention strategies? How can we measure how many TD items have been prevented?
P5. *TD communication:* Which kind of communication can be effective in preventing TD? Which stakeholders have to communicate with each other in which situations?

3.4 Technical Debt Management

3.4.1 Related Work

The tertiary studies by Rios et al. [RMS18], and by Junior et al. [JT22] provide an overview of the TDM research field. While they analyzed multiple studies on TDM processes, they found no research on the systematic establishment of them. Both agree that a holistic approach to TDM, encompassing diverse TD activities and types, is lacking, which makes it difficult for practitioners to benefit from the research findings. Similarly, the vision paper by Avgeriou et al. identified that TDM research needs to focus on multiple TD types instead of only focusing on code debt and should include more industrial datasets [Avg+23]. This coincides with our finding that IT managers seem to lack guidelines on how to establish TDM [WB23] (publication No. 3 *(IT Managers' Perspective)*).

Kruchten et al. published a book that can be used as a guide for establishing TDM, particularly when following the "What can you do today?" sections [KNO19]. Although the guide relies on research findings, its practical applicability has not been scientifically evaluated and may require improvement.

To analyze TDM processes in practice, mainly case studies have been performed. Santos et al. shared insights from an architecture team overseeing 25 agile teams. As a result, they recommended the use of a TD board to manage and visualize high-level TD [San+13]. Guo et al. concentrated their case study on identifying the costs associated with their approach to managing TD, which revolves around a TD list and includes five steps for evaluating and prioritizing TD items [GSS16]. As they implemented this method, they faced several challenges, which they detailed in a separate paper [Guo+16]. Ultimately, they simulated their approach with historical data from actual company releases rather than applying it directly in a practical setting. Ramasubbu et al. merged TD management into a quality management process [RK19]. They established a TD register to catalog TD items linked to their requirements and the resulting defects. Additionally, they contemplated design moves to implement requirements or repay TD, and analyzed the design moves' risks. They performed a cost-benefit analysis of this information and suggested a method to oversee TD. Their research was targeted at companies that are already engaged in quality assurance processes, and their strategy did not prioritize TD prevention. Malakuti et al. outlined the TD management strategy for a particular project, along with the challenges faced when introducing TD concepts to a company. Due to those challenges, they ultimately did not offer a practical solution [MO20]. Reboucas de Almeida et al. created the Tracy framework for prioritizing TD based on business needs. They proposed this framework to prioritize TD repayment efforts not solely on technical requirements but also considering the business value associated with the configuration item containing the TD item [Reb19]; [Reb+21].

In addition to the case studies, five action research studies to implement and analyze TDM processes have been performed so far.

To confirm the feasibility of the TDM framework proposed by Guo et al. [SG11], Oliveira et al. conducted an action research study encompassing three action cycles over the course of five months [OGS15]. They focused their research on identifying, documenting, and prioritizing existing TD items.

Similarly, Yli-Huumo et al. concentrated on the identification, documentation, and prioritization of TD [YH+16]. Their research involved assessing the risks and benefits associated with these TD activities and included five distinct steps: problem identification, TD identification, TD documentation, TD evaluation (i. e., measurement), and TD prioritization.

The study by Borup et al. focused on TD deliberation, which encompasses communicating about, weighing, and reflecting on TD [Bor+21b]. They conducted two interventions over a period of ten months. They addressed some gaps left by previous studies, particularly concerning TD prevention. Yet, their work did not include TD management activities such as documentation or prioritization.

Detofeno et al. carried out two action research studies to implement a TDM process within a Brazilian company [DRA21]; [DMR22]. The first study, which included two cycles over two years, established a TDM guild – a community of practice focused on TD [DRA21]. This guild contributed to the identification, monitoring, prevention, prioritization, and repayment of TD. In the second study, they developed a detailed TD prioritization mechanism during two action cycles conducted over two years and eight months [DMR22]. However, their studies were limited to static code analysis of PHP code, i. e., targeting only code debt.

3.4.2 Research Gaps

The adoption of research methods for TDM in practice is still rare [MBB18]; [WB23]; [Avg+25]. This suggests we may have overlooked practical issues in the (solution) approaches we developed for TDM processes. Understanding the problem space is essential to developing feasible solutions. Hence, the question arises whether the research approaches sufficiently studied the problem space before suggesting solutions, or if essential elements are missing in the proposed approaches. **(M1)**

Developing and establishing a TDM process in practice raises logical questions that must be addressed for the approach to succeed. However, in several cases, we were unable to find answers to these questions in the existing research, thereby identifying research gaps.

First, we need to identify the preconditions for establishing a TDM process, e. g., regarding authority, support, time, tools, and commitment. Each case study and action research listed in Section 3.4.1 detailed the context of their respective case companies, which is similar to the preconditions of the respective studies. However, these are retrospective perspectives, static existing facts. We were unable to find comprehensive research on the preconditions a team should meet to establish a TDM process **(M2)**.

IT teams are often under significant time pressure, which is frequently cited as a cause of TD [Guo+16]; [Rio+20]; [Fre+20b]; [Pér+21]. Consequently, we need to address this issue when establishing a TDM process as well. We must ensure that we do not overburden practitioners with the additional time required to uphold the process. The effort invested in the process must be in a positive ratio to the outcome it creates for the team. Otherwise, the time pressure will force the team to abandon the process after a short time **(M3)**.

Most companies have adopted agile development methods [Bec+01]; [Din+12]. Consequently, when establishing a TDM process, it must be integrated with the predominant agile, iterative, and self-determined mindset in the IT industry. While there are studies on TD and TDM in agile work environments, most focus on the problem space [Beh+17]; [Hol+18]. Approaches to TDM are usually rather context-specific and do not sufficiently outline how to transfer them to other agile teams [San+13] or do not evaluate the proposed approaches in practice [ANDG23] **(M4)**. Similarly, Guo et al. noted that a *"Lack of process integration"* is an obstacle to adapting research methods to practical environments [Guo+16]. Accordingly, a successful TDM process must be optimally integrated not only into the agile work environment, but also into existing mindsets, workflows, and tooling **(M5)**.

Besker et al. observed that TD management *"promotes a culture within organizations in which developers are supported and appreciated for managing their TD"* and that it *"leads to a virtuous cycle where the right culture and TD prevention mechanisms reinforce each other"* [Bes+20]. However, there is no research on how to initiate such cultural change. Research on establishing a TDM process should aim to initiate this cultural change by identifying relevant psychological effects and taking them into account **(M6)**. In accordance with this issue, Martini et al. found that the *"split of budget in project budget and maintenance budget boosts the accumulation of debt"* as people responsible for the project budget do not take TD into account, as

this is a maintenance issue [MBC14]. Consequently, overcoming this split may reduce the amount of incurred TD. No prior research has addressed this issue **(M7)**.

Most studies, including the ones listed in Section 3.4.1, present methods focusing on one or a few specific TD activities, like TD identification [Zaz+13], TD documentation [San+13], TD measurement [NVK11]; [CSS12]; [RK19], TD prioritization & repayment [Sch13a], or TD prioritization [Sea+12]; [Reb+18]. Moreover, many studies are limited to code debt or self-admitted-TD in code comments [Kat+22]; [ZBA22]; [DRA21]; [DMR22]. A holistic process that includes all TD activities and TD types is still missing [RMS18]; [JT22] and calls for further research **(M8)**.

All action research studies report on TDM processes of single companies or teams even though transferability is an important quality for action research studies [OGS15]; [YH+16]; [Bor+21b]; [DRA21]; [DMR22]. Action research studies that produced transferable, less context-sensitive TDM processes do not exist yet **(M9)**.

Since many studies focus on code debt, numerous studies have also used static code analysis to assess the success of an approach by measuring the amount of TD items in a system [DRA21]; [DMR22]. This reduces the measurement to code debt and often does not factor in the priority of the repaid TD. Otherwise, the evaluation of a TDM process's success has relied on individuals' perceptions gathered through interviews or questionnaires, often yielding only anecdotal evidence. This implies that no reliable metrics to assess a TDM process have been developed. Developing entirely new measurement methods or adding methods to the existing ones for triangulation might improve results, particularly when the goal is to analyze multiple TD types **(M10)**.

Research Gaps:

- M1. *Problem Space:* Did we already comprehensively study the problem space? Did we overlook practical issues in establishing a TDM process?
- M2. *Preconditions:* What are the preconditions for establishing a TDM process, e. g., in terms of authority, support, time, tools, and commitment?
- M3. *Required Time:* How can we establish a TDM process without overburdening the practitioners with the time needed for establishing and upholding the process?
- M4. *Process Integration – Establishment:* How can a TDM establishment be integrated into the predominant agile, iterative, and self-determined mindset in the IT industry?
- M5. *Process Integration – Resulting Process:* How can we ensure that the TDM process is optimally integrated into existing mindsets, workflows, and tooling?
- M6. *Culture Change and Psychological Effects:* How can such a change in culture be initiated? Which psychological effects are relevant and must be taken into consideration when establishing a TDM process?
- M7. *Budget Split:* How can we overcome the split of the budget in the project budget and the maintenance budget that boosts the accumulation of debt?
- M8. *Holistic Process:* How can a holistic TDM process that includes all TD activities and is not limited to specific TD types look like?
- M9. *Transferability:* How can we ensure this transferability between companies?
- M10. *Measuring Effects:* How can the effects of establishing a TDM process be measured, particularly with regard to other TD types than code debt?

Chapter 4

Research Design

As explained in Section 1.2, the ultimate goal of this study is to define a set of best practices for TDM as a starting point and a method for establishing TDM in practice.

To reach this goal, the research of this thesis is separated into three parts:

- Step I focuses on the analysis of the research area to identify the requirements for a TDM process and methods or hindrances to establishing a TDM process.
- Step II is the design of a method to establish TDM.
- Step III evaluates this method in practice and deduces TDM best practices.

Step I follows an inductive research approach, analyzing the TDM context with the goal of developing a method for establishing TDM. This method serves as the resulting theory of inductive research. The theory can be phrased as follows: "The method will support practitioners in establishing a TDM process in their teams and companies." Step II is the active part of conceptualizing the results from Step I and designing a research artifact, i. e., the method for establishing TDM. Part III comprises testing the aforementioned theory, i. e., evaluating the research artifact. Consequently, this part of the thesis is deductive research aimed at proving the theory.

4.1 Research Paradigm

This thesis is structured using Stol et al. [SF15] *Research Path Schema*, as presented in Figure 4.1. According to these guidelines for software engineering research, research is divided into three aspects, referred to as *"Research Element Domain"*:

- ***Substantive Domain***: The *"phenomena and real-world systems"* [SF15] under research.
- ***Conceptual Domain***: The *"concepts, models, frameworks, and theories"* used to *"describe the properties of, and relationships among the 'things' found in the substantive domain"* [SF15].
- ***Methodological Domain***: The *"methods and techniques to gather data about a study topic (substantive domain) or theories (conceptual domain)."* [SF15]

All domains are relevant to each study. However, research goals differ: some aim to generate new theories, others to evaluate hypotheses, or to demonstrate the value of a method or technique. Depending on these research goals, the researcher's primary interest is in one of the three domains. Studying one of those domains (primary domain) yields an *intermediate research product* supported by a secondary domain. From the *intermediate research product*, the final, tertiary domain emanates. The order in which the researcher selects the domains to produce a *intermediate research product* determines the research path.

For the *observational research path*, researching the *substantive domain* using methods from the *methodological domain* yields a set of observations that can then be conceptualized, thereby forming or extending part of the *conceptual domain*.

Concepts from the *conceptual domain*, supported by corresponding methods from the *methodological domain*, forge a study design that can then be tested in the *substantive domain* to validate the proposed idea. This is called *study design research path*.

For the *hypothetical research path*, concepts from the *conceptual domain* supported by analyzing real-world situations of the *substantive domain* form a set of hypotheses. The selection

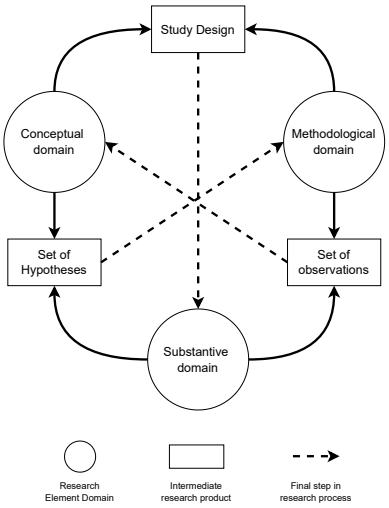


FIGURE 4.1: Research Path Schema by Stol et al. [SF15]

of an adequate research method from the *methodological domain* is the final step in this research path.

For each research path, the primary and secondary domains may be swapped, yielding a second variant for that path, resulting in the following six possible research paths:

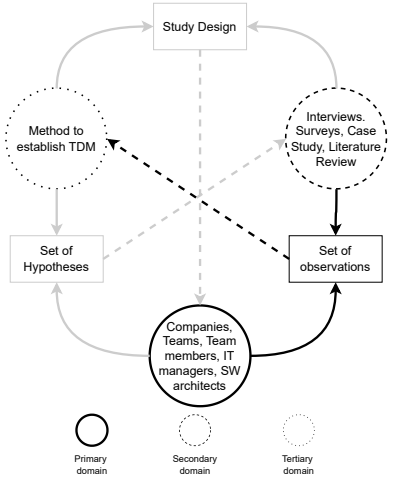
- Concept-driven hypothetical research path
- System-driven hypothetical research path
- Method-driven observational research path
- System-driven observational research path
- Concept-driven study design research path
- Method-driven study design research path

Following these schema definitions, we can identify the following main research paths for this thesis:

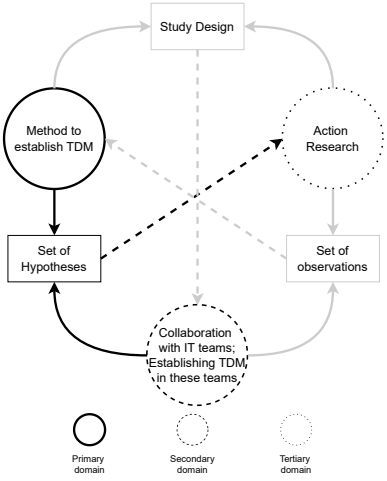
Step I follows a *system-driven observational research path* as presented in Figure 4.2(a). The research analyzes the substantive domain using various methods from the methodological domain to derive observations that inform the conceptualization of a model or framework. The method to establish TDM is the resulting conceptualization, which constitutes Step II of this thesis.

Step III follows a *concept-driven hypothetical research path* as presented in Figure 4.2(b). The theoretical framework, i. e., the method for establishing TDM, is part of the conceptual domain, which is the primary domain. The collaboration with practitioners and establishment of the method in practice forms the secondary domain, i. e., the substantive domain. Action research ultimately proved to be an appropriate method, representing an element of the methodological area, i. e., the tertiary domain of this research.

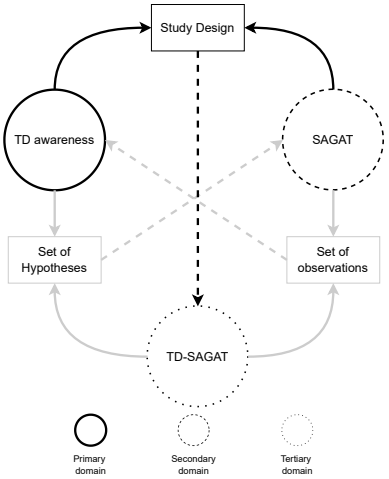
Neither part of this research follows a *study design research path*, which means the chosen methodologies stem from the research goals of the respective studies rather than guiding the research itself. However, we adapted the SAGAT (Situation Awareness Global Assessment Technique) method from the psychological domain to measure TD awareness, which we called TD-SAGAT. This development could be interpreted as such a *study design research path*, as presented in Figure 4.2(c). The TD awareness concepts and the SAGAT method from psychology constituted the study design, which was then applied as TD-SAGAT in Step III of this thesis.



(a) Step I – system-driven observational research path



(b) Step III – concept-driven hypothetical research path



(c) Development of TD-SAGAT – concept-driven study design research path



FIGURE 4.2: Research Paths Pursued in this Thesis

4.2 Research Strategy

As introduced in the previous Section, this thesis is separated into three parts. The following sections introduce the research strategy for each Step of this thesis.

4.2.1 Step I – Analysis of Research Area and Gaps

The goal of Step I was to analyze the research area by conducting inductive studies that applied various methods across multiple substantive domains to obtain a broad view. As explained in Section 4.1, this follows a *system-driven observational research path*, in which the goal is to develop a concept from observations in the substantive domain.

First, an extensive literature review provided an overview of the research domain, its terminology, the problem space, and existing solutions. The results of this literature review

motivate each publication in this thesis and serve as background information in Chapter 2 and related work in Chapter 3. However, a published literature review study is not included in this thesis because multiple systematic literature reviews ([Amp+15]; [BMB16]; [Bec+18]; [Len+21]; [LCP20]), systematic mapping studies ([LAL15]; [Alv+16]; [Rib+16]; [KM20]; [Sar+21]; [Das+22]; [Per+23]; [SRB23]; [Lei+24]), and even tertiary studies ([RMS18]; [JT22]) have already been conducted by other researchers. The literature review revealed research gaps in the application of solutions in real-world scenarios [Avg+23]; [Avg+25] and in providing a holistic TDM process that encompasses various TD types and TDM activities [RMS18]; [JT22]. Additionally, it revealed that insights from IT managers, i. e., stakeholders that are highly relevant to establishing a TDM process, are underrepresented.

Publications No. 1 *(TAP Framework)* and 2 *(TAP Framework – extended)* describe an initial case study that we performed at Gruner + Jahr Deutschland GmbH. This case study provided insights into a team that developed a TDM process without research interference or knowledge from research. We conducted surveys with team members and managers and compared the results to those of a team that did not adopt any TDM approach. This provided information on the effects of a TDM process on a team, specifically regarding TD awareness, communication, and prevention, which then became the primary focus of the ongoing research on establishing a TDM process in practice. We specifically identified the relevance of determining TD cost factors for each TD item to enable more rational discussions and decisions. However, we could not determine if the effects result from the specific characteristics of the chosen TDM process or if any TDM process would produce similar outcomes. Finally, the TDM process introduced a new repayment method for TD, in which TD is repaid upon its incurrence as part of the same project or feature development, but after reaching a given deadline.

Publication No. 3 *(IT Managers' Perspective)* fills the gap on the IT managers' insights. In this interview study, we interviewed 16 IT managers and conducted a 3-person focus group discussion to assess existing knowledge about TD, identify TDM strategies from practice, and pinpoint IT managers' concerns regarding TD. To this end, we transcribed, translated, and coded the interviews to identify insights and to conceptualize them. From the gathered TDM knowledge and existing processes, we developed a model of the causes and consequences of TD – the V4CTD model of Visibility, Cycles & Chains of Causes & Consequences of TD. We found that initiating a TDM process, i. e., establishing one, is particularly challenging for IT managers. This realization shifted the focus of the further research to establishing a TDM process rather than solely to defining an adequate one. Furthermore, communication regarding TD appears to be a significant issue for IT managers, both in terms of communicating with business stakeholders as well as with developers.

Both studies identified communication about TD as a significant aspect, prompting the next study on handling and communicating TD. As ATD is the hardest to repay and is understudied in comparison to code debt, this study focuses on ATD, i. e., mistakes in architecture or architectural decisions that proved suboptimal over time. Publication No. 4 *(Learning from each other)* presents the results of this interview study, which followed the grounded theory method. We interviewed software architects to ask how they define architectural mistakes and how they handle them. We identified several relevant aspects of open communication, e. g., providing safe spaces to discuss mistakes. We also showed that failing to provide this psychological safe space prevents architects from communicating these mistakes and, consequently, from handling them at all. This includes the learning opportunity arising from mistakes, which can be used to prevent them in the future. One method to provide a safe space and foster learning opportunities across a company is a *community of practice (CoP)*, which we used to scale the process of one team to the whole company. Furthermore, this study's results are integrated into the resulting method for TDM establishment, as we, as a first step toward establishing TDM, focus on creating a safe space to discuss TD-related issues.

Three co-authored publications provided further details relevant to establishing a TDM process.

Publication No. 9 *(Debiasing)* evaluates a workshop to debias architectural decision-making. All kinds of decisions are prone to bias, i. e., to mistakes our brain makes when we do not consciously focus on the decision. In terms of TD, this means that less TD gets incurred

if we avoid those mistakes, i. e., if we make more rational decisions, as shown in publications No. 1 *(TAP Framework)* and 2 *(TAP Framework – extended)*. The debiasing workshop demonstrates how to make architectural decisions consciously by identifying alternative solutions and their associated drawbacks and risks. In an experimental set-up, two practitioners (either developers or IT students) discuss a software architecture they are both familiar with. One (test) practitioner participates in the workshop, and the other serves as the control group. Both are then separately asked to identify optimization potential for the discussed architecture in a think-aloud protocol. By coding the think-aloud protocol and evaluating biased and unbiased decisions, we showed that test practitioners are better at debiasing their decisions. The workshop slides of this publication are an element of the TDM establishment method, informing practitioners about the effects of debiasing their decisions.

Publication No 8 *(AK in Web Search Engines)* focuses on the acquisition of architectural knowledge through utilizing web search engines. Relevant to this thesis, the study shows that drawbacks are underrepresented in the results, which makes biased decisions more likely, as we have demonstrated in publication No. 9 *(Debiasing)* . We used this knowledge to inform practitioners about this risk during the establishment of the TDM process.

4.2.2 Step II – Artifact and Study Preparation

This section explains how the previously gathered insights are conceptualized, resulting in a 5-step workshop concept for establishing a TDM process and determining an appropriate research method.

The literature review uncovered many research results that might be helpful in practice, but were not adopted. Consequently, the focus of this thesis was not to develop further research approaches for TDM but to analyze which of the already existing approaches identified or developed by scientists might be helpful and feasible in practice, and what the possible reasons were for not adopting these. The question arose whether the lack of adoption was due to a broad lack of knowledge transfer from science to practice, the non-transferability of research results, or other challenges that hindered the adoption of existing approaches.

4.2.2.1 5-Step Workshop Concept

Our study aimed to develop a TDM process in collaboration with our industry partner, drawing on prior academic findings. From a business perspective, this resembled a small change-management project, as work processes were modified [Fü21]. For such a change process, it was essential to include the employees who would work with these processes in the future [Fü21]. Moreover, the agile development methods adopted in most IT companies rely heavily on team self-organization and do not promote process changes enforced by higher hierarchical levels [Mey14]; [Max18].

As a result of these considerations, we developed a workshop concept that presents various TD research results to practitioners, structured around the TD activities. The workshops iteratively introduced each TD activity in five subsequent steps, forming the final process by adding new TD activities with each step, as depicted in Figure 4.3.

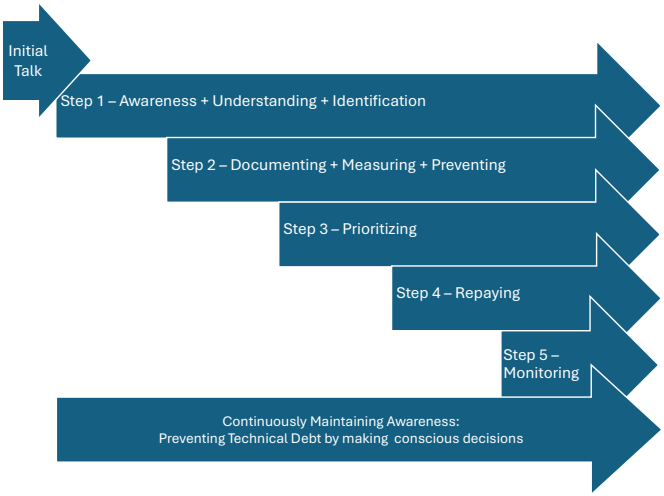


FIGURE 4.3: 5-Step Workshop Concept

We let these practitioners decide which results from previous research they wish to adopt, to adapt to their needs, or to reject. This self-determined approach aligns best with the agile mindset of a self-organized team rather than providing a predefined TDM process. In this setting, the practitioners were also able to specify their own ideas for further approaches. There were six to eight weeks between workshops, although we were prepared for situations in practice to cause further delays. Due to this separation of topics, participants could test approaches and reject them in subsequent workshops. They could also assess the effort each approach required and whether they were willing to invest this effort to obtain the resulting outcome. This procedure aligns with the agile concept of "inspect&adapt" [Moe13]. Such an agile workshop approach is highly flexible and accounts for the specifics of each team and company. Accordingly, we assumed that it is less context-specific than other approaches and can be adopted in multiple settings.

While the second to fifth workshops were designed to introduce academic approaches to the practitioners, the first workshop had two different purposes: First, the workshop should foster a sound foundation for encouraging open communication, as we identified this as a significant issue in TDM in the publications No. 1 *(TAP Framework)* , No. 2 *(TAP Framework – extended)* , and No. 3 *(IT Managers' Perspective)*. Second, the workshop was designed to identify the team's current TDM approach in order to establish a baseline for research and help the team become aware of its current approach, including the associated problems.

The findings from publication No. 4 *(Learning from each other)* informed the establishment of an appropriate communication environment by encouraging teams to discuss their TD items and the issues related to their TDM process. Additionally, we developed a TD Game to foster communication and discussion about TD in a non-critical, emulated, and fun environment. We introduced and evaluated the Game in publication No. 5 *(TD Game)*. To assess whether the pre-specified learning goals were achieved and whether participants planned to change their attitudes or behaviors after playing the game, 46 IT practitioners participated in 13 game sessions of the TechDebt Game. After the game, they completed a survey asking them whether they encountered the learning goals and if they changed their attitude and planned to change their behavior. We found that the game was particularly instructive for junior developers and business stakeholders. Senior developers did not gain much new insight, but they identified many real-world elements in the game, thereby proving that the game provides an appropriate emulation of real life. Notably, we found that participants changed their attitudes and planned behavior changes, making the game a good starting point for the first workshop.

After playing the game, we diagnosed the current TD management process in three subsequent steps:

1. We collected concrete examples of TD, i. e., "TD stories" on TD incurrence/causes, TD consequences, and TD repayment, to gain initial insight and to give participants

some room to present their TD experiences. This also supported changing a potentially frustrated basic mood into a more constructive mood.

2. We identified the current approaches to managing TD by discussing the following questions.
   - How do you currently identify TD items?
   - What are you currently doing after you have identified a TD item? (e. g., document, communicate, repay, ignore)
   - In which situation do you intentionally incur TD? On which information do you base this decision?
3. We gathered requirements for process improvement.

The second through fifth workshops were structured similarly. Each workshop began by reminding participants of the workshop's goal and recapitulating the current status of the TDM process. Then, the researchers presented their evaluation results of the period since the last workshop. Participants were asked to discuss, evaluate, and, if appropriate, improve the previous interventions and the current process. In this phase, previously adopted approaches may be rejected due to newly gained insights, e. g., if the effort required by the approach does not justify the outcome. Finally, the researchers introduced the new topics, namely TDM approaches to each TD activity, in presentations for the respective workshops, each lasting no more than 45 minutes. An overview of all workshop topics is given in Table 4.1, and the respective academic approaches are presented in Table 4.2. More detailed information on the topics covered and the methods used for each workshop is presented in the publication No. 6 *(Establishing TDM)*. After the presentation, the participants discussed the approaches and decided on which ones to adopt, adapt, or reject. At the end of each day, we compiled the topics raised and the measures the team decided on, and we assigned at least one responsible participant to each measure.

Figure 4.4 shows an overview of how the RQs' results contribute to the workshop concept (green arrows, Step 1) and how the RQs will be answered by its evaluation (orange arrows, Step 3).

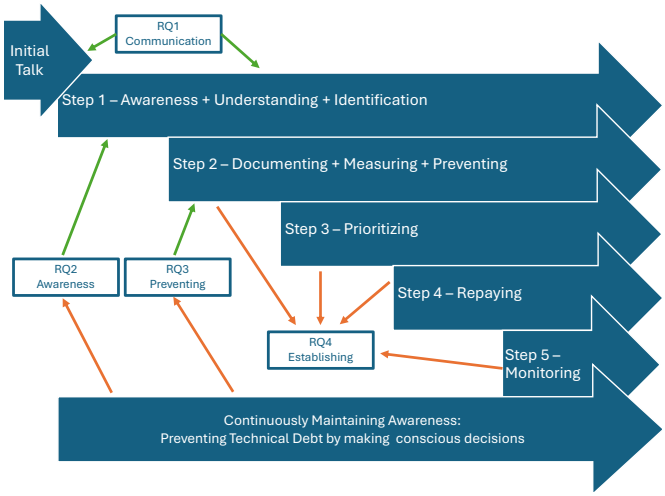


FIGURE 4.4: Alignment of the Workshop Concept and the Research Questions

4.2.2.2 Measuring TD Awareness

The final open topic for evaluating changes in TD awareness was the identification of a suitable method for measuring it. A literature review did not reveal a corresponding method in software engineering, prompting a search for suitable measurement methods in other domains, such as business or psychology, to adapt them to the TD domain. As explained in Section 4.1, this follows a *concept-driven study design research path*. Given that the concept is TD awareness, we seek a method to form a study design that will then be applied in practice.

No.	Workshop topics	Approaches from TD activity
Introduction Talk (a 1hr. Talk prior to the workshops)	TD definition & TD types Causes & consequences Cost & benefit estimation Prioritization factors TD backlog	TD definition TD identification TD measurement TD prioritization TD repayment
1	Study introduction Short summary of the introduction talk TD Game Previous TD instances Current TDM process Wishes and requirements for a TDM process TD manager & meetings	TD identification & definition (as above) (Multiple; the game's learning goals) TD organization
2	TD in mechatronics Quality attributes Cognitive biases TD tickets attributes	TD prevention TD prevention TD prevention TD measurements & documentation
3	Prioritization TD vs. TD or functional requirements Prioritize one-time vs. recurring Prioritization factors Calculate a priority vs. gut feeling	TD prioritization TD prioritization TD prioritization TD prioritization
4	TD repayment strategies (Ignore, Refactor, Rewrite)	TD repayment
5	TD Visualizations from research Story-telling with visualization Visualization ideas based on participant interviews Visualization ideas based on ROI	TD visualization TD visualization TD visualization & monitoring TD visualization & monitoring

TABLE 4.1: Workshop topics and their assignment to the TD activities

TD activity	**Presented approaches**
TD organization	Establishing a TD champion [JG23]
TD identification & definition	Automatically by static analysis tools Manually by Dagstuhl definition [Avg+16a] Manually by backlog colors [KNO12] Manually by identifying whether TD interest occurs [Avg+23]
TD prevention	Well-defined requirements, code evaluation/standardization, following well-defined project processes, well-defined architecture/design, using good practices, improving tests, and reviews [Rio+20]; [Fre+20b]; [Pér+21] Educational sessions on how to avoid TD [BMB20] Incorporating a TDM approach into project management [Wie+22] (publication No. 2 *(TAP Framework – extended)*) Debiasing decision-making by listing at least one solution alternative, one drawback, and one risk per alternative [BZK21] Better decision-making by considering quality attributes [ISO23]; [BCK12] Counteract TD inheritance between disciplines in mechatronics [DOVH19]; [VHB21]
TD measurement	Interest, interest probability, effort/principal for prioritization [TAV13]; [McC08] Interest, interest probability, effort/principal for decision-making [Per+24]
TD documentation	Recording TD in a backlog [KNO19]; [JT22]
TD prioritization	Prioritizing TD against functional requirements [Len+20] Prioritizing TD against each other [Len+20] Prioritizing once [Len+20] Prioritizing constantly [Len+20] Based on the systems evolution [Sch13a] Based on the business value [Reb+18] Based on cost attributes (TD measurements) [TAV13] Based on the affected quality attributes [RR22]
TD repayment	Repay TD if it impedes a functional requirement [Fre+23] Pay the interest [Bus11] Wait for discontinuing or migrating of the affected software system [Sch13a]; [Sch13b] Repay TD if the positive effect is proven by analyzing the costs [McC08]; [TAV13] Repay TD by assigning a quota [McC08]; [Wie+22] Repay if TD might hinder the system's evolution in the near future [Sch13a] Repay if TD was incurred due to a project deadline [Wie+22] (publication No. 2 *(TAP Framework – extended)*)
TD monitoring & visualization	DebtFlag [HL13], MultiDimEr [SUW22], VisminerTD [Men+19], MIND [FR15], TEDMA [FS+17] Self-developed TD visualization using Microsoft PowerBI [Mic25c]

TABLE 4.2: Presented approaches with references

The literature in other domains uncovered three different types of awareness in the psychology domain with conceptual similarities to TD awareness:

- *Contingency awareness* pertains to the connection between two occurrences, namely a conditional and an unconditional stimuli [DS87]. This seems to correlate with the TD management issue, in which stakeholders frequently fail to associate TD causes with their consequences [WB23] (publication No. 3 *(IT Managers' Perspective)*).
- *Bounded awareness* refers to people focusing on a limited portion of available information, leading to decisions based on that narrow view [CB07]. This bias often arises from the specific information emphasized. For instance, if a developer aims to complete an assignment on time, they may prioritize creating functional software over optimizing it and reducing TD.
- *Situation awareness* was introduced by Endsley [End88] and "refers to the perception of the elements of the environment within a volume of time and space, the comprehension of their meaning and the projection of their status in the near future" [Zho+23]. In the context of TD, these perceived elements can be interpreted as the information available regarding a TD item. Stakeholders need to comprehend the information and project the future effects of TD-related decisions.

In collaboration with a researcher in psychology, we analyzed these three concepts and identified an appropriate measurement method for TD awareness. *Contingency awareness* demonstrates limited adaptability to TD Awareness because the stimuli (the occurrence of TD and TD's consequences) must be presented to participants simultaneously, yet they are overly abstract in the case of TDM. Regarding *bounded awareness*, there is no specific metric for its assessment that we could have adopted. Instead, metrics have been developed for particular cases that do not fit our purpose [BS16]. We determined that *Situation Awareness* [End88] corresponds with our focus, as it emphasizes decision-making and the collection of the information required to make these decisions. We viewed decision-making as the core issue of TDM, as IT stakeholders face two crucial decisions in TD: Whether to incur TD or not, and whether to pay it back or not [Bus11]; [WB23] (publication No. 3 *(IT Managers' Perspective)*). As a result, we concluded that the Situation Awareness Global Assessment Technique (SAGAT) [End00] was the most suitable research method for our investigation. Furthermore, this method provided the most objective measurement of awareness.

We adopted Endsley's situation awareness definition for TD awareness as follows: *TD awareness is the (1) perception of the prerequisites (e. g., alternatives, benefits, drawbacks) of a decision (e. g., incurring TD) during decision-making situations (e. g., refinement meetings), (2) the comprehension of these prerequisites' meaning, and (3) the projection of their status in the future (e. g., TD's consequences).* (Endsley often uses the term requirements instead of prerequisites. We changed the term to avoid confusion with the terms for functional and non-functional requirements in the software engineering domain.)

To identify relevant prerequisites, the SAGAT method directs us to determine the main goals & sub-goals and the decisions required to achieve them. Prerequisites for situation awareness are derived from these decisions, leading to questions in a questionnaire regarding participants' consideration of these prerequisites [End00]. Applying this approach as detailed in publication No. 6 *(Establishing TDM)* and improved in publication No. 7 *(Practical Guide)*, we identified the following prerequisites for TD decisions: (1) solution alternatives, (2) solution's benefits, (3) solution's drawbacks, (4) solution's risks, (5) effort for implementing a solution, (6) TD item's principal, (7) TD item's interest, (8) component affected by TD, and (9) quality attributes affected by TD.

The concept of SAGAT involves interrupting participants while they are engaged in their work or in a simulation of that work (e. g., a flight simulation) and questioning them about their current situation to gather their awareness of it [End00]. We adopted this approach, which we named TD-SAGAT, and interrupted participants during meetings in which they discussed backlog issues, including functional requirements, maintenance issues, and TD repayment. In agile development methods, these meetings are commonly referred to as refinement meetings. We interrupted these meetings at two to three randomly selected points every second or third meeting. Then, we asked the participants to complete the TD-SAGAT questionnaire, in which they indicated, using 'yes' or 'no', whether they considered the (randomly ordered) prerequisites of the previously discussed backlog issue.

We explained this TD-SAGAT approach in more detail in publication No. 6 *(Establishing TDM)* and utilized it for publications No. 6 *(Establishing TDM)* and No. 7 *(Practical Guide)*.

4.2.3 Step III – Artifact Evaluation

The last part of this thesis is the evaluation of the resulting workshop concept with industry partners. As explained in Section 4.1, this part of the thesis follows a *concept-driven hypothetical research path*, with the concept being the 5-step workshop concept. This implies the concept is now evaluated in a real-world setting, and the appropriate research method will emerge. Accordingly, the following requirements determined the research method, each of which further narrows down the range of possible methods:

1. The method should be suitable for research applied in practice. In terms of the "ABC of software engineering research" by Stol and Fitzgerald [SF18], the research is situated in a natural setting, i. e., a field experiment or a field study.
2. Given the workshop concept's interactive nature, the goal is not only to observe practitioners but also to interact with them. Thus, the method should allow for intrusion. This means that our research must be conducted as a field experiment, i. e., an evaluative case study, a quasi-experiment, or an action research [SF18].
3. The iterative workshop concept aligns with practitioners' agile work environment and a change-management approach for business-domain processes [Fü21]. For this iterative approach with interventions at regular intervals, i. e., the five workshops, action research is the most appropriate method, as it comprises iterative action cycles that likewise align with the iterative approach [Sta20].

The goal of action research is to change work processes in a practical environment. In contrast to design science research, action research does not produce an artifact suitable for practical use but rather a process for practice. However, both allow for researchers to interact with the practitioners, which is essential to provide a high degree of realism. The drawback of field experiments is that the accuracy of measurements is influenced by confounding contextual factors, making it impossible to statistically generalize the results [SF18]. Furthermore, action research within a specific organization may not be readily applicable to other organizations, as the researcher's interventions are likely influenced by that organization's unique context [SF18]. In the context of action research, the empirical standards set by the *ACM Special Interest Group on Software Engineering* prioritize aspects such as reflexivity, credibility, resonance, usefulness, and transferability over concerns regarding generalizability or reliability [SIG25]. Nonetheless, we have taken measures to mitigate biases, which we explained in publication No. 6 *(Establishing TDM)*. Additionally, we replicated this action research in three companies with the specific goal to analyze whether the results are transferable. This means evaluating whether the workshop concept is transferable and whether the resulting processes exhibit similarities.

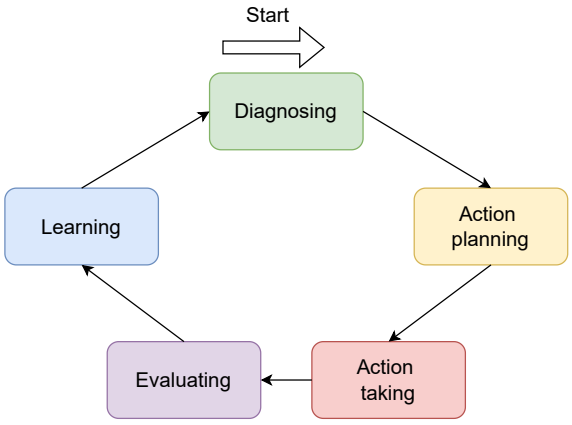


FIGURE 4.5: Action Cycles by Staron [Sta20]
(Colors added to indicate the alignment Figure 4.6)

Action research is conducted in iteratively repeating action cycles, which comprise five phases each [Sta20] as shown in Figure 4.5:

- *Diagnosing:* Identifying which problem should be solved
- *Action planning:* Identifying a set of actions to solve the problem
- *Action taking:* Executing the identified actions
- *Evaluating:* Analyzing the actions' effects and whether they solved the problem
- *Learning:* Identifying improvements to enhance the action's efficacy in effectively addressing the problem.

Action research should terminate when the underlying problems are solved, making it crucial to define a termination criterion before initiating the research. Due to the lack of reliable, objective measures of a successful TDM process, we concluded our action research when participants indicated satisfaction with the developed process in a questionnaire. This also assures participants' motivation throughout the establishment process.

The iterative action research design, including all five workshop topics and major actions taken, is presented in Figure 4.6. The workshop topics of this figure align with those introduced in Table 4.1. We conducted action research studies across three teams from different companies. The teams and companies are presented in Tables 4.3 to 4.4, which show the variation in contexts. (The order of the entries in Table 4.3 has been randomized to anonymize the assignment to the teams and thus the results per company.) More details, e. g., on the participants, are provided in publication No. 7 *(Practical Guide)*.

Company	Domain	Employees	Organizational Form
DATEV eG.	Finances & Taxes	$>$ 8,000	Cooperatively organized
anonymus	Consumer goods	$>$ 2,000	Family business
TRUMPF SE & Co KG	Machine tools	$>$ 15,000	Family business

TABLE 4.3: Participating companies

Team	Size	Team's main task	Issue-tracking System
A	8	Providing a central service	Azure DevOps [Mic25a]
B	5	Frontend (+Backend for Frontend)	Jira [Jir]
C	8	Central communication platform	GitLab Issues [Git]

TABLE 4.4: Participating teams

Tables 4.5 to 4.6 provide an overview of the collected data during the action research studies, i. e., Step III of the thesis, which was used for the following analyses:

- The results of the first workshops, together with the answers to its questionnaire, enabled us to assess each team's current handling of TDM as a basis for our research.
- We used the TD-SAGAT prerequisite questions during interruptions of refinement meetings, in questionnaires at each workshop, and as part of an observation protocol to evaluate TD awareness across three data sources.
- In the observed meetings, we identified potentially debt-incurring decisions and discussed them with the team during the subsequent workshops.
- The *actions taken* show which approaches are feasible for practitioners. Comparing the approaches across all three studies enabled us to distinguish those that are generally feasible from those that are not feasible or feasible only in specific contexts.
- We compiled a list of challenges observed in meetings and discussed in workshops. For each challenge, we either provided a solution or identified research gaps to be addressed in subsequent research. Comparing the challenges across all three studies revealed common issues that practitioners may encounter with research results, common questions that arise, and relevant research gaps that researchers should prioritize.
- Feedback on the workshops collected in questionnaires during the workshops allowed us to improve the workshop for the subsequent studies or provide ideas to improve it even further for future usage.

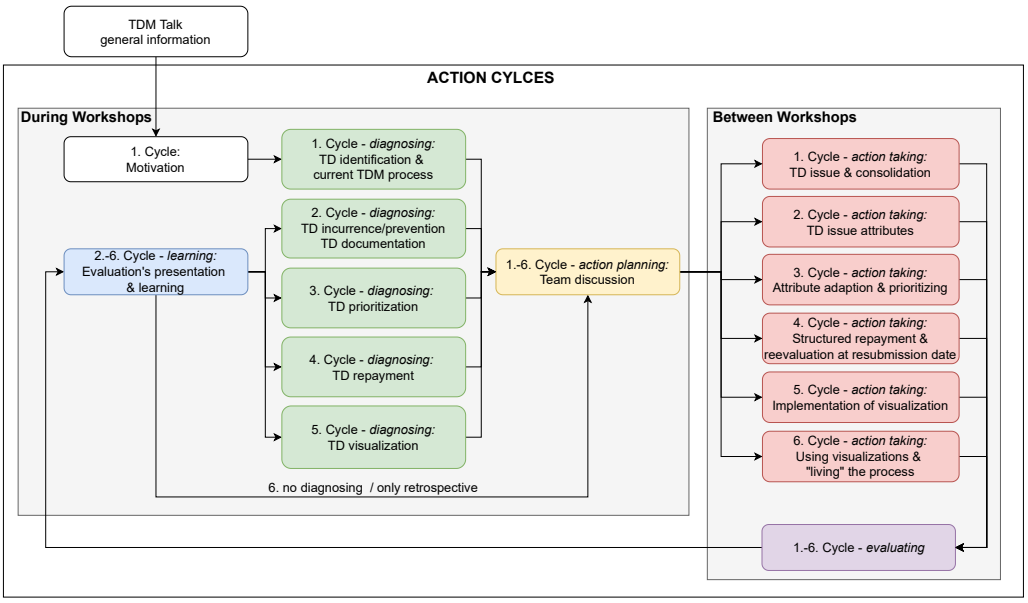


FIGURE 4.6: Study Design presenting Action Cycles, Introduced TD Activities (Diagnosing Phase), and Measures (Action-Taking Phase) [WSB26]

Team	Time			Work-	Retro-	Action
	Months	Begin	End	shops	spectives	cycles
A	16	2023-05	2024-08	5	1	5
B	16	2024-06	2025-09	5	2	6
C	11	2024-12	2025-10	5	1	5
Sum	30 (net; 2023/05–2025/10; 43 gross)			15	4	16

TABLE 4.5: Data collection per team (time and interventions)

Team	Meetings				TD-SAGAT	
	Cnt.	Hrs	Discussed issues	Observed by 2 researchers	in <x> meetings	No. of answers
A	40	24.5	488	5	12	90
B	57	53.5	507	14	10	57
C	11	18	137	5	6	93
Sum	108	96	1132	24	28	240

TABLE 4.6: Data collection per team (meetings and TD-SAGAT)

Comparing the results from all three studies, we assembled "best practices", i. e., approaches adopted by all teams, "nice-to-haves", i. e., approaches found suitable by some teams, common questions, and common challenges. As a result, we created the TDM Guide, a Whitepaper for practitioners that includes the following sections:

- An introductory part on TD background,
- An outline of the generally feasible approaches,
- A step-by-step guide to introduce TDM,
- A collection of typical challenges with their respective solutions if available,
- A collection of typical questions with answers, and
- An outlook on how to scale the resulting process to a whole company.

4.3 Research Methods

The following section comprehensively provides more details on the research methods used. As Section 4.2.2 already introduced TD-SAGAT and Section 4.2.3 action research, this section

does not further detail these methods.

4.3.1 Data Collection

Overall, we used the following methods for data collection, which are introduced below.

- Questionnaires
- Interviews
- Observations
- Analysis of backlog issues

We used questionnaires in publications No. 1 *(TAP Framework)*, No. 2 *(TAP Framework – extended)*, No. 5 *(TD Game)*, No. 6 *(Establishing TDM)*, and No. 7 *(Practical Guide)*. All questionnaires comprised both closed-ended questions (i. e., questions that can be answered by selecting an option) and open-ended questions (i. e., questions that can be answered by providing text). While open-ended questions may yield more detail and insight, closed-ended questions are easier and usually faster for participants to answer and for researchers to evaluate. Because participants lose focus when completing questionnaires that take too long, we designed all questionnaires to be completed in less than 15 minutes. Consequently, all questionnaires were designed to include as many closed-ended questions as possible, while still allowing for open-ended feedback, e. g., by asking whether the participant has further comments to share. The only exception to this is the TD-SAGAT questionnaire, which comprises only "yes or no" answers, as it must be completed in very few minutes to keep interruptions as short as possible. Closed-ended questions were analyzed using quantitative methods, e. g., descriptive or inductive statistics, and open-ended questions were coded using *initial (open) coding*, i. e., qualitative methods. All questionnaires were reviewed by one or two practitioners to identify issues and misunderstandings in the questions or in the predefined responses. During the action research studies, all questionnaires were completed in the researcher's presence, which reduces the risk of misunderstandings by allowing direct feedback from the researcher if questions arose. To distribute the questionnaires, we used Google Forms [Goo] for the publications No. 1 *(TAP Framework)* and No. 2 *(TAP Framework – extended)*. For all subsequent questionnaires, we used LimeSurvey [Lim].

Interviews were part of publications No. 3 *(IT Managers' Perspective)*, No. 4 *(Learning from each other)*, and to some extent of publication No. 6 *(Establishing TDM)*. For the interviews, we developed interview guidelines outlining the questions we planned to ask the interviewee. The guidelines typically comprised seven to ten main questions plus sub-questions detailing the required information to be gathered from the interviewee. During the interviews, we strictly stuck to the main questions, but sometimes differed in sub-questions depending on the progress of the interview and the interviewee's knowledge. All interviews were held online, and sessions were recorded and transcribed. For international collaboration in publication No. 3 *(IT Managers' Perspective)*, all interviews were additionally translated into English to enable all researchers to gain insights from them. Furthermore, for publication No. 3 *(IT Managers' Perspective)*, we produced analytic memos based on the interviews. Memos can originate from the interviews themselves as well as from analyzing the interview transcripts during the data analysis phase. They incorporate various reflections from the researcher and often serve as a basis for potential hypotheses to be investigated further in the study. We coded the transcribed interviews as described in the following subsection.

Additionally, we used an observation protocol for the action research studies reported in publications No. 6 *(Establishing TDM)* and No. 7 *(Practical Guide)*, to systematically observe the team's meetings and collect data in a structured, comparable manner. As data collection must be conducted swiftly to follow the ongoing discussion, the protocol contained mostly check boxes. We recorded:

1. Basic information such as date, issue ID, and title as text,
2. The issue type (TD or not TD) as recorded by the team, as well as our assessment in check boxes,
3. Whether the researchers identified a risk for potential TD incurrence in a check box,
4. Whether the TD-SAGAT prerequisites were mentioned in the respective check boxes, and

5. Remarks as text.

As remarks, we documented the following:

- The reasons behind our evaluation of the differences for the issue type,
- The specifics related to possible TD incurrence, and
- Significant events we aimed to address with the participants in the upcoming workshop.

An overview of meetings per team can be derived from Table 4.6. To reduce bias in the results, 24 meetings were observed by two researchers, who subsequently deliberated on discrepancies in their assessments.

Finally, we analyzed the teams' backlog data for publications No. 1 *(TAP Framework)*, No. 2 *(TAP Framework – extended)*, No. 6 *(Establishing TDM)*, and No. 7 *(Practical Guide)*. To this end, we counted the number of TD items opened and closed per month over the observed time period and presented the results as descriptive statistics. This was the extent of analysis for publications No. 1 *(TAP Framework)* and No. 2 *(TAP Framework – extended)*. For the action research studies (publications No. 6 *(Establishing TDM)* and No. 7 *(Practical Guide)*), we also evaluated specific TDM attributes that were added to the backlog issues as part of the ongoing TDM process establishment and presented them during the workshops. For example, we calculated the interest burden as the product of interest and interest probability, or the return on investment per TD item, utilizing interest, interest probability, and principal. In these studies, the presentation of backlog information was not only relevant for research but also part of the resulting TDM process in the last step of "TD visualization & monitoring." Consequently, the final descriptive statistics were taken from the TD visualization used by the team and did not need to be calculated or visualized by the researchers.

4.3.2 Data Analysis

Overall, we used the following methods for data analysis as introduced below:

- Coding of qualitative data
- Descriptive statistics for quantitative data
- Inductive statistics for quantitative data

For interviews and open-ended questions, we coded the responses to identify similarities and structure the insights. Furthermore, we used coding in the publication No. 9 *(Debiasing)* to analyze the "think-aloud protocols" of architecture improvement ideas.

We used the "The Coding Manual" from J. Saldaña [Sal13] for direction on the optimal coding procedure and the MAXQDA tool [Max] to execute the coding.

The coding techniques that we used included:

- *Initial coding*, often referred to as *open coding*, involves separating qualitative data into distinct segments, carefully analyzing them, and identifying similarities and differences. This type of coding is not strictly defined by a specific formulaic approach. The objective is to keep an open mind to all potential avenues.
- *Structural coding* means indicating a part of the transcript that contains information related to a specific subject. Structural codes are collected prior to analysis. We used *structural coding* to create a relation of the data to the RQs or other specific questions we had, e. g., "What are the causes for TD incurrence?"
- *Process coding* means identifying segments that present specific actions. Each process code included a verb in its gerund form (ending in "ing"). For TDM, TD activities are typical process codes; e. g., "Prioritizing TD" was used to highlight sections in which participants discussed how they prioritized TD items.
- *Value coding* is used to mark participants' values, attitudes, and beliefs. In our study, the names of these codes were always preceded by "V:" for Value, "A:" for Attitude, or "B:" for Belief. For example, "V: TD(M) is an important topic" was a code used to mark a segment in which the participants stated that they believe that managing TD is important to their work.

- *Descriptive coding* summarizes the basic topic from a particular segment or passage in a word or short phrase. In relation to TDM, we, for example, used the code "Good Developers are leaving" to label sections in which participants described situations in which frustrated developers left the company because TD was not addressed.
- *Focused coding* is a more advanced coding. It organizes coded information according to thematic or conceptual similarity. As such, it merges codes, categories, and concepts and selects core categories on which the further analysis focuses.

For open-ended questions, we always used initial coding, with one main researcher coding and a second researcher who checked the results. For our interview studies in publications No. 3 *(IT Managers' Perspective)* and No. 4 *(Learning from each other)*, we used two coders and continually reviewed, extended, and reorganized the coding lists, iterating through interviews multiple times in cross-coding phases to ensure that the codes were comparable and identical in meaning. To organize the collaboration among researchers, we used the coding guidelines of Campbell et al. [Cam+13] to establish a common coding scheme. Accordingly, we solved the "Unitization Problem With Units of Meaning" [Cam+13] (problems with the mismatch of the length of the coded text) by deciding that the segments that we will be coding should always be blocks between punctuation marks. We applied the negotiated agreement approach [Gar+06] to resolve the "Discriminant Capability Problem" [Cam+13] (discrepancies in coding by different coders). Subsequently, all coding differences were discussed until full consensus was reached.

In contrast, for publication No. 9 *(Debiasing)* , we used context-free coding [ES93], i. e., the researchers split their transcripts into meaningfully cohesive sections. In this case, we split the transcripts into distinct decisions made, including their rationales. A single coder then coded the randomized sections.

Descriptive statistics were used in nearly all studies except for the interview studies No. 3 *(IT Managers' Perspective)* and No. 4 *(Learning from each other)*. In the simplest form, we counted occurrences of events (e. g., "yes" responses to a questionnaire item, checked boxes in the observation protocol) and presented them as a bar chart. In most cases, we compute the percentage of occurrences by dividing the number of occurrences by the total number of possible occurrences. We used this simple statistic in all publications (except the interview studies), e. g., for backlog data or for the TD-SAGAT prerequisites, gathered through the workshop questionnaire, the interruptions questionnaire, and observations. As explained earlier, we, furthermore, calculated more complex KPIs, such as interest burden and return on investment, for the publications No. 6 *(Establishing TDM)* and No. 7 *(Practical Guide)*. We performed the underlying calculations and created the presentations using Microsoft Excel [Mic25b]. For publications No. 1 *(TAP Framework)* and No. 2 *(TAP Framework – extended)*, we also calculated the ϕ-coefficient of Pearson's $\tilde{\chi}^2$-correlation [Coh88]; [Pea01] for dichotomous variables to identify correlations between the participants' answers. However, because we had only a few participants, the results must be evaluated with caution and only indicate effects rather than prove them.

Finally, we used inductive statistics for the publications No. 1 *(TAP Framework)*, No. 2 *(TAP Framework – extended)*, and No. 9 *(Debiasing)* , i. e., we formulated hypotheses and tested them using the adequate statistical tests. For publications No. 1 *(TAP Framework)* and 2 *(TAP Framework – extended)*, we used the Mann-Whitney U-Test [MW47], as it can be applied to ordinal data with independent variables that are not normally distributed. For publication No. 9 *(Debiasing)* , we employed the nonparametric Wilcoxon Signed-Rank test [Wil92] for dependent variables and a small sample size. For correlation calculations and hypothesis tests, we used IBM SPSS Statistics [Sps].

Chapter 5

Publications

5.1 Publications

Table 5.1 shows all publications I contributed to during the work on my thesis. All publications are related to the thesis's research topic, but not all contribute directly to it. The publications that contributed to this thesis are marked in the table and listed in detail in the following Section 5.2.

Year	Cont.	Citations
2020		*Introducing a Framework for Managing Technical Debt Developed by Practitioners*, Wiese M., 22. Workshop Software-Reengineering & -Evolution 2020 [Wie20]
2021	x	*Exploring Web Search Engines to Find Architectural Knowledge*, Soliman M., Wiese M., Li Y., Riebisch M., Avgeriou P., 2021 IEEE 18th International Conference on Software Architecture (ICSA) [Sol+21b]
	x	*Preventing Technical Debt by Technical Debt Aware Project Management*, Wiese M., Riebisch M., Schwarze J., 2021 IEEE/ACM International Conference on Technical Debt (TechDebt) [WRS21]
		A Marketplace for Industrial Case Studies in Software Engineering, Borowa K., Wiese M., Tamzalit D., Allen C., Singjai A., 8th ACM Celebration of Women in Computing: womENcourage™ [Bor+21a]
2022	x	*Preventing Technical Debt with the TAP framework for Technical Debt Aware Management*, Wiese M., Rachow P., Riebisch M., Schwarze J., Information and Software Technology, Volume 148 [Wie+22]
2023		*Taxonomy of Architecture Maintainability Smells*, Rachow P., Wiese M., van Hoorn A., 30th Asia-Pacific Software Engineering Conference (published by IEEE) [RWH23]
	x	*IT managers' perspective on Technical Debt Management*, Wiese M., Borowa K., Journal of Systems and Software, Volume 202 [WB23]
	x	*Learning From Each Other: How Are Architectural Mistakes Communicated in Industry?*, Wiese M., Brand A.-F., van Hoorn A., 17th European Conference on Software Architecture (ECSA) [WBH23]
2025	x	*The TechDebt Game - Enabling Discussions about Technical Debt*, Wiese M., Heinrichs A., Rusieshvili N., de Almeida, R. R., Borowa K., 2025 IEEE/ACM International Conference on Technical Debt (TechDebt) [Wie+25]
	x	*Debiasing Architectural Decision-Making: An Experiment With Students and Practitioners*, Borowa K., de Almeida R. R., M. Wiese M., 2025 IEEE 22nd International Conference on Software Architecture (ICSA) [BAW25]
		How to Mitigate Technical Debt in Startups: Extending the Entrepreneurial Software Engineering Model, Klünder J., Droste J., Heintz F., Wiese M., Kuhrmann M., Projektmanagement und Vorgehensmodelle 2025 [Klu+25]
2026	x	*Establishing Technical Debt Management – A Five-Step Workshop Approach and an Action Research Study*, Wiese M., Serwa K., Besier A., Marion-Jetten A., Bittner E., Journal of Systems and Software, Volume 231, [Wie+26]
	x	*A Practical Guide for Establishing a Technical Debt Management Process*, Wiese M., Serwa K., Bittner E., 2026 IEEE/ACM International Conference on Technical Debt (TechDebt) [WSB26]

TABLE 5.1: Publications published during the thesis
(Cont. – Publication contributes to the thesis)

5.2 Contributing Publications

This section presents portraits of all papers that contributed to this thesis.

I published three journal articles and five conference papers as the main author, which are presented in Section 5.2.1. Three additional conference papers that I co-authored are outlined in Section 5.2.2. Two publications outline the final workshop concept and introduce the results of its analysis, i. e., No. 6 *(Establishing TDM)* and No. 7 *(Practical Guide)*. They, therefore, represent Steps II and III of this thesis, while the other publications are elements of Step I.

5.2.1 Publications as Main Author

This section presents portraits of publications in which I was the main author and that contributed to this thesis.

CONFERENCE	
Citation	M. Wiese, M. Riebisch and J. Schwarze, *Preventing Technical Debt by Technical Debt Aware Project Management*, 2021 IEEE/ACM International Conference on Technical Debt (TechDebt), 2021, Madrid, Spain, pp. 84-93, https://doi.org/10.1109/TechDebt52882.2021.00018
Ranking	Core2023: B
Research Goal	This research analyzes a TDM process developed by practitioners that specifically focuses on the TD cause of tight deadlines. The approach distinguishes four issue types in their ticketing system: maintenance, maintenance projects, TD, and deconstruction. For them, TD issues are those that were intentionally incurred, and they have developed an approach to ensure repayment as part of a project.
Methodology	Comparative case study, descriptive statistics, correlation analysis, hypothesis testing
Research Contribution	• The paper presents a new TD repayment approach in which TD items are collected upon incurring them and must be repaid as part of the project or feature development before the project is deemed complete. → RQ4 *(Establishment)* • We showed that team members were less prone to, and less willing to incur, TD when they knew they would have to repay them in the near future, thereby creating a feedback loop in team behavior. → RQ2 *(Awareness)* ; RQ3 *(Prevention)* • We showed that the approach led to more timely repayment of TD directly after the project ended. → RQ4 *(Establishment)* • By comparing the results of the two units, we showed that adopting the framework significantly led to a more conscious comparison of different solution alternatives and their respective costs. → RQ2 *(Awareness)* ; RQ4 *(Establishment)* • We found significant correlations between comparing solutions alternatives and making more rational decisions. Similarly, estimating implementation and principal costs correlates significantly with leading more rational discussions and making more rational decisions. → RQ1 *(Communcation)* ; RQ4 *(Establishment)* • There were no significant correlations to TD prevention on the 5% significance level. However, the significance of the correlation between comparing solution options and TD prevention was near this level. → RQ3 *(Prevention)*
Co-author's Contribution	***M. Wiese:*** Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Project administration, Resources, Supervision, Validation, Visualization, Writing (original draft), Writing (review & editing) *M. Riebisch:* Supervision, Writing (review & editing) *J. Schwarze:* Resources

TABLE 5.2: Portrait of conference publication No. 1 *(TAP Framework)*

	JOURNAL
Citation	M. Wiese, P. Rachow, M. Riebisch, J. Schwarze, *Preventing technical debt with the TAP framework for Technical Debt Aware Management*, Information and Software Technology, Volume 148, 2022, 106926, ISSN 0950-5849, https://doi.org/10.1016/j.infsof.2022.106926
Ranking	Core2020: A VHB-Rating-2024: B
Research Goal	This research extends publication No. 1 *(TAP Framework)*. We extended the evaluation by conducting a qualitative analysis of the team leads' and IT managers' feedback on the effects of introducing the TDM approach for their management.
Methodology	Questionnaire with open questions, initial coding
Research Contribution	• The extended analysis shows how IT managers benefit from the TDM approach by gaining an overview of TD, thereby enabling them to plan the project pipeline and TD repayment activities more effectively. → RQ4 *(Establishment)* • Managers report that they are better able to communicate TD-related issues to their customers, thereby again demonstrating that the TDM approach optimized overall communication. → RQ1 *(Communcation)* • The IT managers specifically mentioned that awareness of TD was raised among them due to the established TDM process. → RQ2 *(Awareness)*
Co-author's Contribution	***M. Wiese:*** Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Project administration, Resources, Supervision, Validation, Visualization, Writing (original draft), Writing (review & editing) *P. Rachow:* Formal analysis, Investigation, Writing (review & editing) *M. Riebisch:* Supervision, Writing (review & editing) *J. Schwarze*: Resources

TABLE 5.3: Portrait of journal publication No. 2 *(TAP Framework – extended)*

JOURNAL	
Citation	Marion Wiese, Klara Borowa, *IT managers' perspective on Technical Debt Management*, Journal of Systems and Software, Volume 202, 2023, 111700, ISSN 0164-1212, https://doi.org/10.1016/j.jss.2023.111700
Year	2023
Ranking	Core2020: A VHB-Rating-2024: B
Research Goal	While the developers', architects', and even project managers' perspectives have been considered in dedicated studies, no study has focused on the IT managers' perspective on TD. However, IT managers are usually responsible for all aspects of IT management, including TDM. Consequently, the goal of this study was to examine IT managers' perspectives on TD to identify research gaps and areas requiring further attention. We identified the IT managers' knowledge on TD, their management approaches, their main concerns in terms of managing TD, and the measures that they propose to improve TDM.
Methodology	Inductive, interviews, focus group discussion, structural, process, value, and descriptive coding, analytic memoing
Research Contribution	• We found that the original metaphor may not be helpful for conveying the meaning of TD and proposed alternatives used by IT managers. → RQ1 *(Communcation)* • We identified causes and consequences on different visibility levels for business stakeholders, IT managers, and development teams. These different levels may be responsible for further communication issues. → RQ1 *(Communcation)* • We developed the V4CTD model of Visibility, Cycles & Chains of Causes & Consequences of TD as an extension of the TD conceptual model defined at the Dagstuhl seminar 2016. Additionally, we provided guidelines for using this model to create a 3-minute pitch slide that enhances communication with stakeholders. → RQ1 *(Communcation)* • We uncovered the duality of the IT managers' problems in communicating with developers: the "developers crying wolf" and the "it works" developer. → RQ1 *(Communcation)* • We observed that the status quo bias (an irrational preference for maintaining the current state) hinders the establishment of a TDM process. We further identified three subconcerns (raising awareness of TD, creating an overview of TD items, and measuring and prioritizing TD items) that must be addressed to overcome it. Two of these sub-concerns can be addressed by establishing a basic TDM process as a starting point, creating a cyclic dependency between concerns about the TDM's establishment and the TDM's establishment itself, which must be broken by "just starting". → RQ4 *(Establishment)* • We reported the values of the old systems and called them vintage systems. Top managers notably mentioned these systems and were most prevalent in older, established companies. Although it is a significant issue in practice, research in this field appears to be limited. → /
Co-author's Contribution	***M. Wiese:*** Conceptualization, Data curation, Investigation, Methodology, Project administration, Resources, Validation, Visualization, Writing (original draft), Writing (review & editing) *K. Borowa:* Conceptualization, Data curation, Investigation, Methodology, Resources, Validation, Visualization, Writing (original draft), Writing (review & editing)

TABLE 5.4: Portrait of journal publication No. 3 *(IT Managers' Perspective)*

CONFERENCE	
Citation	M. Wiese, A.F. Brand, A. van Hoorn, *Learning from Each Other: How Are Architectural Mistakes Communicated in Industry?*, 17th European Conference on Software Architecture (ECSA), 2023, Istanbul, Turkey, Lecture Notes in Computer Science, vol 14212. Springer, Cham. https://doi.org/10.1007/978-3-031-42592-9_22
Ranking	Core2023: A
Research Goal	In software architecture, we make mistakes like in every other domain. By sharing our experiences with mistakes we made, we increase the likelihood that other architects will not repeat them. Thereby, they can avoid architecture TD. However, in software architecture, mistakes are not commonly discussed topics. Even using the term "mistake" in the context of software architecture may seem offensive. The question of how architects handle and communicate software architecture mistakes has not yet been systematically researched and is the goal of this study.
Methodology	Grounded Theory, interviews, initial and focus coding, constant comparison, theoretical saturation
Research Contribution	• We present a definition of software architecture mistakes from the practitioners' perspective. →/ • We developed a conceptual model for the management of software architecture mistakes in practice. This model includes (1) the mistakes' detection, (2) how they are handled, (3) how we can learn from them, and (4) how we can avoid repeating them in the future. → RQ1 *(Communcation)* ; RQ3 *(Prevention)* • We identified how the mistakes are communicated, including factors that facilitate or hinder the communication. Moreover, we present strategies for optimizing communication from practitioners' perspectives. → RQ1 *(Communcation)* ; RQ4 *(Establishment)* • To optimize the handling of software architecture mistakes, we specify that detection triggers should be enhanced on a personal level but also on an organizational level, e. g., by architecture reviews. → / • As an open research question, practitioners asked for guidelines on when to change or revise an architecture due to a mistake and when not to. → RQ4 *(Establishment)* • We further determine that communication opportunities should be ensured, particularly on a company or cross-company level, e. g., by launching communities of practice for this purpose. → RQ1 *(Communcation)* ; RQ4 *(Establishment)*
Co-author's Contribution	***M. Wiese:*** Conceptualization, Formal analysis, Methodology, Project administration, Supervision, Validation, Visualization, Writing (original draft), Writing (review & editing) *A. F. Brand:* Data curation, Formal analysis, Investigation, Methodology *A. v. Hoorn:* Supervision, Writing (review & editing)

TABLE 5.5: Portrait of conference publication No. 4 *(Learning from each other)*

CONFERENCE	
Citation	M. Wiese, A. Heinrichs, N. Rusieshvili, R. R. d. Almeida and K. Borowa, *The TechDebt Game - Enabling Discussions about Technical Debt*, 2025 IEEE/ACM International Conference on Technical Debt (TechDebt), 2025, Ottawa, ON, Canada, pp. 23-33, https://doi.org/10.1109/TechDebt66644.2025.00010
Ranking	Core2023: B
Research Goal	Previous research has shown that communication about TD is challenging, prone to misunderstandings among stakeholders, and subject to psychological factors given its critical nature. To improve communication, we created a game that emulates TD concepts and allows players to immerse themselves in the issues developers face when TD is amassed. During the game, aimed at all stakeholders, discussion among them is encouraged, fostering a safe environment for developing mutual understanding of each other's issues. This study evaluates the success of this TD game.
Methodology	Iterative game development (tabletop and online variants), experiment, Surveys, descriptive statistics, initial coding
Research Contribution	• We showed that the game provides a realistic emulation of the real world. → RQ1 *(Communcation)* , RQ4 *(Establishment)* • While all players gain new insights, business stakeholders and junior stakeholders gain the most knowledge from playing the game. → RQ1 *(Communcation)* • Junior developers learned that ignoring TD is sometimes valid, and to be more careful in implementations to avoid TD. In the future, they might demonstrate greater understanding in discussions with experienced developers. → RQ1 *(Communcation)* ; RQ3 *(Prevention)* • Business stakeholders learned more about the characteristics of TD and that some characteristics might not be visible to them. This knowledge may be useful for business stakeholders to place greater trust in the technical stakeholders' evaluation of TD items. → RQ1 *(Communcation)* • We showed that most players are open to changing their attitudes and behaviors toward TD and TD management after playing the game. Players unwilling to change their attitudes or behaviors justified this by claiming they already knew about TD and had tried to manage it. → RQ1 *(Communcation)* , RQ4 *(Establishment)*
Co-author's Contribution	**M. Wiese:** Conceptualization, Data curation, Formal analysis, Funding acquisition, Investigation, Methodology, Project administration, Resources, Software, Supervision, Validation, Visualization, Writing (original draft), Writing (review & editing) *A. Heinrichs, N. Rusieshvili, anonymous student:* Data curation, Investigation, Methodology, Software (Game Creation) *R. R. de Almeida:* Investigation, Writing (review & editing) *K. Borowa:* Investigation, Writing (review & editing)

TABLE 5.6: Portrait of conference publication No. 5 *(TD Game)*

	JOURNAL
Citation	M. Wiese, K. Serwa, A. Besier, A S. Marion-Jetten, E. Bittner, *Establishing technical debt management — A five-step workshop approach and an action research study*, Journal of Systems and Software, Volume 231, 2026, 112606, ISSN 0164-1212, https://doi.org/10.1016/j.jss.2025.112606
Ranking	Core2020: A VHB-Rating-2024: B
Research Goal	In our previous publication (No. 3 *(IT Managers' Perspective)*), we found that establishing a TDM process is among IT managers' main concerns. Consequently, this study aimed to evaluate an approach for establishing a TDM process in an IT company based on a predefined workshop concept. To align with the predominance of agile development, we adopted an agile, i. e., iterative and self-organized approach for establishing the TDM process. Following the iterative principle of agile work, we iterated over five workshops, which also aligns with the iterative approach to changing business processes. In addition to evaluating the workshop concept and identifying useful approaches, we sought to analyze the effort required to establish TDM and the influence of the establishment on participants' TD awareness.
Methodology	Action research, questionnaires, TD-SAGAT, systematic meeting observation, backlog data, descriptive statistics
Research Contribution	• The study showed that the iterative and self-determined workshop concept is suitable for establishing a TDM process. → RQ4 *(Establishment)* • This study identified the approaches the IT team adopted and those it rejected, thereby demonstrating the feasibility and transferability of these research approaches. → RQ4 *(Establishment)* • We introduced the newly developed TD-SAGAT approach adopted from psychology to measure TD awareness, including the nine prerequisites. → RQ2 *(Awareness)* • We showed that establishing the TDM process affects participants' TD awareness in decision-making situations. → RQ2 *(Awareness)* • For TD prevention, we found that presenting information, e. g., during workshops, is insufficient, whereas reminders in the backlog's issue templates, e. g., a Talked about TD checkbox, or attributes to record alternatives, their drawbacks and risks, lead to sustained change. → RQ1 *(Communcation)* , RQ2 *(Awareness)* , RQ3 *(Prevention)* , RQ4 *(Establishment)* • We found that two methods for TD prioritization and repayment were preferred, which are (1) repayment based on the system's evolution and (2) repayment of low-hanging fruits, i. e., based on cost calculations. → RQ1 *(Communcation)* , RQ4 *(Establishment)* • Various new ideas for TDM emerged from this study: (1) using a resubmission date, (2) using a Talked about TD checkbox, (3) using educated guesses for priority instead of calculations, (4) using a ROI-based priority to improve the educated guess and (5) using visualizations for TD prioritization, e. g., to identify the low-hanging fruits. → RQ1 *(Communcation)* , RQ2 *(Awareness)* , RQ4 *(Establishment)* • We found that the team invested less than 3% of their time in establishing and maintaining the TDM process. → RQ1 *(Communcation)*
Co-author's Contribution	***M. Wiese:*** Conceptualization, Data curation, Formal analysis, Funding acquisition, Investigation, Methodology, Project administration, Resources, Supervision, Validation, Visualization, Writing (original draft), Writing (review & editing) *K. Serwa:* Data curation, Formal analysis, Investigation, Writing (review & editing) *A. Marion-Jetten:* Methodology, Writing (original draft), Writing (review & editing) *A. Besier:* Data curation, Formal analysis, Investigation, Software, Visualization, Writing (original draft), Writing (review & editing) *E. Bittner:* Supervision, Writing (review & editing)

TABLE 5.7: Portrait of journal publication No. 6 *(Establishing TDM)* (Part I)

CONFERENCE	
Citation	M. Wiese, K. Serwa, E. Bittner, *A Practical Guide for Establishing a Technical Debt Management Process*, 2026 IEEE/ACM International Conference on Technical Debt (TechDebt), 2026, Rio de Janeiro, Brazil, doi = 10.1145/3794915.3795789
Ranking	Core2023: B
Research Goal	This study aims to identify which previously studied TDM approaches are feasible in practice and the typical challenges that arise during TDM establishment to develop guidelines for establishing a TDM process. The TDM Guide should serve as a starting point for defining best practices by replicating the workshop approach from publication No. 6 *(Establishing TDM)* in two additional companies, i. e., TRUMPF SE + Co KG and DATEV e. g., and comparing the results across all three studies.
Methodology	Action research, questionnaires, TD-SAGAT, systematic meeting observation, backlog data, descriptive statistics
Research Contribution	• We identified best practices based on adoption by all three teams for TD communication, identification, documentation, prevention, measurement, prioritization, and repayment. → RQ1 *(Communcation)* , RQ3 *(Prevention)* , RQ4 *(Establishment)* • We identified four primary attributes that should be documented to support TD prioritization: principal, interest, interest frequency, and contagiousness. Additionally, we identified a method to organize the content of these attributes to guide practitioners. → RQ4 *(Establishment)* • The use of these attributes for prioritization differed across the three companies: one used an educated guess, one used a return-on-investment calculation, and one used a mean value based on these attributes. We were unable to identify a common prioritization strategy. → RQ4 *(Establishment)* • We showed that adding attributes such as solution alternatives, their drawbacks, and associated risks to all issues improved TD awareness. → RQ2 *(Awareness)* , RQ3 *(Prevention)* • We contributed a white paper (*Practical Guide to Establishing Technical Debt Management*) for practitioners outlining the establishment of a TDM process, i. e., the TDM Guide. It provides an introduction to TDM, guidelines for the TDM process, a step-by-step guide to establishing the process, a collection of common mistakes and questions with corresponding suggestions and answers, and an outlook on how to scale such a process at the company level. However, this was not evaluated as part of this thesis, and participants stated that external support may be necessary to establish the TDM process. → RQ4 *(Establishment)* • For researchers, we identified research gaps relevant for practice, including TD identification issues, debiasing design decisions, dealing with multiple interests, or supporting the identification of alternative solutions. → RQ4 *(Establishment)* • We urge researchers to provide more concrete instructions for practitioners when their research claims to be beneficial to practitioners. → *(research goal)* • We encourage stronger collaboration with vendors of issue-tracking systems, as many challenges were related to these tools, specifically in terms of visualization. → RQ1 *(Communcation)* , RQ4 *(Establishment)* • We identified a limitation of this approach as its scalability. Methods for establishing company-wide, similar processes require further research. → RQ4 *(Establishment)*
Co-author's Contribution	***M. Wiese:*** Conceptualization, Data curation, Formal analysis, Funding acquisition, Investigation, Methodology, Project administration, Resources, Supervision, Validation, Visualization, Writing (original draft), Writing (review & editing) *K. Serwa:* Data curation, Formal analysis, Investigation, Software, Visualization, Writing (review & editing) *E. Bittner:* Supervision, Writing (review & editing)

TABLE 5.8: Portrait of conference publication No. 7 *(Practical Guide)*

5.2.2 Publications as Co-Author

This section presents portraits of co-authored publications that contributed to this thesis, specifically to the design of the workshop concept.

CONFERENCE	
Citation	M. Soliman, M. Wiese, Y. Li, M. Riebisch and P. Avgeriou, *Exploring Web Search Engines to Find Architectural Knowledge*, 2021 IEEE 18th International Conference on Software Architecture (ICSA), 2021, Stuttgart, Germany, pp. 162-172, https://doi.org/10.1109/ICSA51549.2021.00023
Ranking	Core2023: A
Research Goal	With this study, we aimed to identify the architectural knowledge (AK) that a software architect can find using web search engines. More specifically, we aimed to identify (1) the sources of AK (i. e., vendor documentation, forums, blogs, scientific content, source code repositories, knowledge repositories), (2) the provided AK concepts (i. e., alternatives, benefits, drawbacks, descriptions and use cases), and (3) the supported steps of the "Attribute Driven Design" (ADD) methodology for designing architectures (i. e., identify design concepts, select design concepts, and instantiate architecture elements). With this information, we intended to determine which AK concepts make a web page relevant to a design decision.
Methodology	Exploratory case study, descriptive statistics, inductive statistics
Research Contribution	• We provide an overview of the various sources of AK and their associated AK concepts on the web, which may guide the extension of AK-capturing approaches. → / • We determined differences in Google's effectiveness during the ADD steps, which help better understand the main capabilities of web search engines for AK. → / • We identified AK concepts on the web that provide the greatest benefit to practitioners when making design decisions, namely a solution's drawbacks and information on use cases, i. e., design decisions successfully made by other parties. This can provide guidance to practitioners on sharing their AK on the web. → RQ3 *(Prevention)* • We examined the availability of AK concepts in web search results and found that, although drawbacks and use cases are the most relevant, they are rarely retrieved by web search engines. → RQ3 *(Prevention)*
Co-author's Contribution	*M. Soliman:* Conceptualization, Data curation, Formal analysis, Methodology, Project administration, Software, Supervision, Validation, Visualization, Writing (original draft), Writing (review & editing) ***M. Wiese:*** Data curation, Formal analysis, Investigation, Visualization, Writing (original draft), Writing (review & editing) *Y. Li:* Data curation, Formal analysis, Visualization, Writing (original draft), Writing (review & editing) *M. Riebisch:* Investigation, Supervision, Writing (review & editing) *P. Avgeriou:* Supervision, Writing (review & editing)

TABLE 5.9: Portrait of conference publication No. 8 *(AK in Web Search Engines)*

CONFERENCE	
Citation	K. Borowa, R. R. de Almeida and M. Wiese, *Debiasing Architectural Decision-Making: An Experiment With Students and Practitioners*, 2025 IEEE 22nd International Conference on Software Architecture (ICSA), 2025, Odense, Denmark, pp. 210-220, https://doi.org/10.1109/ICSA65012.2025.00029
Year	2025
Ranking	Core2023: A
Research Goal	Cognitive biases are predictable and systematic errors in human reasoning that influence decision-making in various areas. Debiasing is a process that improves judgment by mitigating the effects of specific cognitive biases. In architectural decision-making, numerous decisions must be made, which makes the process prone to bias. For example, a common bias is anchoring, which can lead architects to unconsciously favor the first solution they propose, overlooking other alternatives. Prior research has shown that training individuals in debiasing techniques during practical workshops can help reduce the impact of these biases. The goal of this study is to conduct a workshop as a debiasing intervention and evaluate its effectiveness using a control group. We aim to investigate its impact on individuals at different stages in their careers, including students and practitioners.
Methodology	Experiments, test and compare group, context-free coding, descriptive statistics, inductive statistics
Research Contribution	• The use of the proposed debiasing techniques that were taught increased. In particular, the use of "discussing drawbacks" and "listing multiple solutions" techniques increased significantly. → RQ4 *(Establishment)* • The occurrences of cognitive bias for all three researched biases (anchoring, confirmation bias, and optimism bias) decreased. This decrease was significant for anchoring and optimism bias. → RQ3 *(Prevention)* • Practitioners were commonly more influenced by all researched biases than students, and as such, the improvement achieved in their case was more prominent. → RQ3 *(Prevention)* • We provided a set of teaching suggestions based on factors that negatively impacted some of the study's participants. → /
Co-author's Contribution	*K. Borowa:* Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Project administration, Supervision, Validation, Visualization, Writing (original draft), Writing (review & editing) *R.R. de Almeida:* Data curation, Formal analysis, Investigation, Writing (review & editing) ***M. Wiese:*** Data curation, Investigation, Writing (review & editing)

TABLE 5.10: Portrait of conference publication No. 9 *(Debiasing)*

Chapter 6

Contributions

The thesis aimed to analyze why TDM is not common in practice and why research findings are not widely adopted, and to propose ways to address these gaps. As an outcome, the thesis aimed to provide an initial set of best practices for TDM and a method for establishing TDM in practice. In pursuit of this goal, the main contributions of this thesis are of a practical nature. However, the thesis developed theoretical models and identified research gaps that must be addressed for TDM to succeed in practice. Section 6.1 introduces the contributions relevant to researchers and Section 6.2 the practical contributions.

6.1 Research Contributions

The research contributions of the contributing publications that are relevant to this thesis are introduced below and summarized in Table 6.1.

Section	Title	Contribution
Section 6.1.1	Technical Debt Metaphor and Communication Issues	An analysis of TD communication issues IT managers face
Section 6.1.2	V4CTD Model	The V4CTD model of Visibility, Cycles & Chains of Causes & Consequences of TD
Section 6.1.3	Model for Managing Software Architecture Mistakes	A model for handling and communicating architectural mistakes
Section 6.1.4	Effects of a Technical Debt Management Process	An evaluation of the effects the TAP framework had on decision-making and discussions
Section 6.1.5	Measuring Technical Debt Awareness with TD-SAGAT	The new method for measuring TD awareness adopted from psychology
Section 6.1.6	Awareness Effects of the Technical Debt Management Establishment	New insights into the effects on TD awareness when teams establish a TDM process measured by TD-SAGAT prerequisites
Section 6.1.7	Uncovered new Research Areas	A list of new research gaps arising from the challenges we observed in collaboration with practitioners, including the status quo bias, TD identification across stakeholder levels, debiasing design decisions, handling multiple interests per TD item, scaling the workshop concept
Section 6.1.8	Proposal for Improving Research Applicability in Practice	A proposal to improve research's applicability in practice
Section 6.1.9	Improving the Collaboration with Tool Vendors	A call to deepen the collaboration with tool vendors

TABLE 6.1: Overview of research contributions

6.1.1 Technical Debt Metaphor and Communication Issues

Ward Cunningham formed the financial metaphor to improve communication with business stakeholders [Cun92]. Klaus Schmid presented the limitations of the metaphor from a theoretical perspective [Sch13b]. In publication No. 3 *(IT Managers' Perspective)*, we showed that most participants had developed their own alternative metaphor to explain TD to their business stakeholders, which shows that the financial metaphor does not provide the desired benefits. For example, the alternative metaphors compared TD to a train car, an injured knee, quick sand, or a bicycle.

Furthermore, this publication shows that communication issues, although previously studied [KNO12]; [Reb19]; [BMB22], remain a critical concern for IT managers. Compared

with prior research on communication with business managers, we showed that IT managers also have difficulties communicating with developers on this topic. Overall. IT managers demonstrated threefold communication issues:

- *With business managers:* IT managers expressed a desire for improved communication regarding business decisions and for business stakeholders to participate more actively in projects. Another concern was the level of trust that business stakeholders placed in IT managers. IT managers wanted business stakeholders to recognize the significance of TD and to have a clearer understanding of its consequences. Lastly, IT managers needed sufficient time to manage TD without the need for ongoing negotiations for each TDM item.
- *With"developers crying wolf":* The issue that emerges with developers who prioritize quality is what we referred to as "developers crying wolf", drawing from the fable "The boy who cried wolf" [Aes02]. These developers often express concerns about TD items without considering their effect on the business, specifically regarding the effectiveness of the TD item [Sch13a]. This situation complicates the recognition and prioritization of important TD issues. IT managers might become numb to the concept of TD, leading to developers' concerns being overlooked during actual crises.
- *With "just do it" developers:* These types of developers typically avoid considering alternative solutions, work independently, do not challenge requirements, and withhold their concerns. The IT managers indicated that, with respect to TDM, particularly TD prevention, they view it as their responsibility to change these developers' mindsets.

These finding indicates that metaphors are beneficial, but the financial metaphor is overrated with respect to understandability and communication support. However, the metaphor is widely known, so the terminology should not be changed. We also showed that TD communication remains an essential issue with no practically feasible solution yet, and that this communication involves individuals and personalities that must be assessed differently. Consequently, further research on communication support is warranted and may include an investigation of how metaphors can be beneficial in this context and how communication depends on personality rather than stakeholder classification.

6.1.2 V4CTD Model

Publication No. 3 *(IT Managers' Perspective)* introduces the V4CTD model of Visibility, Cycles & Chains of Causes & Consequences of TD as an advancement of the conceptual model proposed by Avgeriou et al. [Avg+16a]. The V4CTD model, introduced in Section 2.5, provides a comprehensive overview that integrates visibility aspects, categories of causes and consequences, and chains and vicious cycles into a unified framework. In contrast to the Dagstuhl conceptual model, the V4CTD model adds chains, outer cycles, categories, and another perspective on visibility.

The model distinguishes categories of TD causes and consequences that are quite similar to the findings from Rios et al. [Rio+19], who identified such categories as part of the InsighTD project (see Section 2.5). Given the nature of our interview partners, our list is less comprehensive regarding technical or development-related causes and consequences and places greater emphasis on other areas, such as business-related causes. Consequently, we have predominantly categorized the causes and consequences by their primary role within a company (e. g., management, business, technology, or human resources), thereby identifying the source of or the affected party of TD, which corresponds to our chain model. Additionally, we identified *time- and budget-related* causes as the most prominent causes, and *decisions* and *lack of awareness* as cross-role causes, as they are not directly associated with any single role. Rios et al. [Rio+19], however, were mostly guided by the technical perspective, not distinguishing between roles. For example, they identified *methodology* as a cause encompassing methodological mishaps at multiple levels, including a *lack of code reviews* at the developer level and a *lack of IT governance* at the management level. Similarly, the category *organizational* includes *lack of qualified professionals* at the developer level and *inadequate management decisions* attributable to management.

On the consequences side, Rios et al. found a *quality* category, which encompasses architectural issues, like *low reliability*, as well as *poorly defined scope* or *project not completed*, which may most likely be caused by the IT management or even the business itself. Overall, we identified more similar categories for consequences: the *people* category is similar to our *human resources* category, the *organizational* category to our *business* category, and *planning & management* to our *management* category. All other categories found by Rios (*quality*, *maintenance*, and *development*) are technical issues, which we summarized as *technological*. Again, the differences in terminology indicate our more management- and chain-based perspective.

The research community can use this model to obtain a structured overview of TD mechanisms associated with its causes and consequences, thereby improving understanding of the related problem space. Moreover, our research identified chains that had not been described previously, as they are likely visible only to IT managers rather than developers or other stakeholders with a more limited view.
The V4CTD model also contributed to the workshop concept. We used it after the first workshop to organize our teams' current TD landscape based on their descriptions during the workshop and the first survey. We presented the teams with the results during the second workshop to help them focus on relevant chains and to break potential cycles.

6.1.3 Model for Managing Software Architecture Mistakes

Mistakes in software architecture, i. e., design choices that ultimately prove to be inefficient, are one category of ATD. Much of the research emphasizes enhancing software architecture processes, especially concerning decision-making [Bha+20]; [Tan+17] and the collection of software architecture knowledge [WG16]. Architects address such mistakes by conducting "post-mortems" [BWA09] or by defining anti-patterns [Ric15] and architectural smells [MSB21]. However, the question of how to share mistakes among architects effectively, with the aim of learning from them and preventing similar ATD in other systems, has not been systematically investigated.

As a central contribution, publication No. 4 *(Learning from each other)* introduced a model that clusters the management of software architecture mistakes into four subsequent actions and visualizes the communication-based inter-dependencies among these actions, as depicted in Figure 6.1. Below, a brief description of these actions is provided; further details can be found in the original publication No. 4 *(Learning from each other)*.

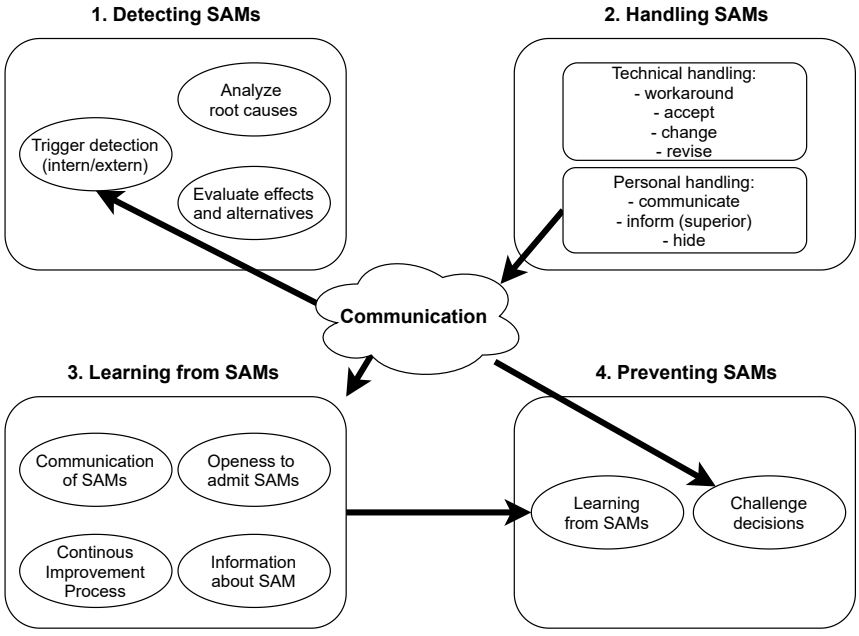


FIGURE 6.1: Model of Managing Software Architecture Mistakes (SAM) [WBH23]

1. *Detecting software architecture mistakes:* A software architecture mistake is identified when a trigger prompts a reassessment of a situation. Once detected, the situation is evaluated, e. g., by identifying root causes of the mistake, and alternatives are explored, prompting a subsequent action.
2. *Handling software architecture mistakes:* Software architecture mistakes are addressed through both technical solutions and personal approaches. The technical approach involves developing workarounds and either accepting, modifying, or revising the software architecture. The personal approach includes discussions with coworkers and supervisors.
3. *Learning from software architecture mistakes* Architects must communicate software architecture mistakes to other stakeholders to learn from them. The capacity to learn from software architecture mistakes relies on the willingness to acknowledge them openly, both personally and within the organization. A well-defined *Continuous Improvement Process* facilitates the identification of software architecture mistakes, communication, and knowledge collection. Key details of software architecture mistakes, including the decisions made and their underlying rationale, should be documented.
4. *Preventing software architecture mistakes* To prevent mistakes in software architecture, it is essential to communicate and learn from previous experiences. Additionally, engaging in discussions about difficult software architecture choices in advance can help mitigate errors in this area.

We provided a theoretical basis, i. e., a model to explain how practitioners manage software architecture mistakes. The model provides an initial overview but should be explored and improved, and may be extended through research ideas that practitioners can adopt.
Furthermore, we identified two new research questions. The first question is how to determine whether to accept a software architecture mistake, modify the architecture, or revise it completely. The second question concerns how to optimize communication about software architecture mistakes at the enterprise level, in addition to the ideas mentioned by our participants, like initiating a CoP. For the second question, we further provided first insights by listing facilitating and hindering factors.
The results of this communication analysis informed the first workshop, which aimed to create a friendly, psychologically safe environment.

6.1.4 Effects of a Technical Debt Management Process

In a comparative case study, reported in publications No. 1 *(TAP Framework)* and No. 2 *(TAP Framework – extended)*, we assessed the effects of a TDM process on TD communication, prevention, and decision-making. Many case studies reported on TDM processes or introduced and analyzed them in practice [GSS16]; [Reb+21]; [RK19]; [MO20]. In contrast, we compared a team that introduced a TDM process with one that did not. Consequently, the following results were derived from a comparative case study that used statistical tests to compare participants' perceived effects, rather than reporting observations. To the best of the authors' knowledge, no other comparative case studies have been performed to research TDM effects.

In terms of TD awareness, 75% of participants reported being aware of incurring TD at the time it occurred, with no notable difference between the observed and the comparison units. However, responses to open-ended questions support the finding that awareness was considerably raised. Participants began to think more critically about when to incur TD, how to manage it, and when to avoid it. The managers of these teams also suggested that awareness had increased. Additionally, all IT managers pointed out that making TD items visible increased overall TD awareness.

The IT managers noted a clearer understanding and more productive discussions. Communication between IT management and the teams has improved significantly. The enhanced overview has also led to better communication with customers and greater acceptance of repayment measures. Furthermore, our calculation of correlations allows us to connect solution comparisons and cost estimations to more rational discussions, i. e., improved communication, and more rational decisions, i. e., potentially prevention of TD incurrence.

The correlations do not support direct associations for TD prevention or for the estimation of interest costs. The lack of correlations in interest costs may be attributable to the fact that only one-third of participants estimated these costs. The absence of correlations for TD prevention suggests that solution comparisons or cost estimates do not directly influence it. Regardless, 82.4% of participants identified TD prevention as a benefit of the framework, a finding further supported by open-ended responses. Similarly, managers noted that decisions to incur TD were made more consciously, thereby facilitating its prevention. Finally, it is reasonable to infer that rational decision-making, for which we observed correlations, supports TD prevention.

The perceived benefits of a TD overview included more logical decision-making and improved planning of the project pipeline, as it provided justification for TD repayments and incorporated them into projects. Specifically, project pipeline management benefited from increased visibility. Project durations (including the time allocated to TD repayment) became clearer, allowing managers to defer upcoming projects as needed.

This case study underscores that establishing a TDM process may yield benefits beyond managing TD. The process may improve communication, foster TD prevention through improved decision-making, and enable more effective project planning. Given these benefits, this study motivates further research into the effects of establishing a TDM process, which should include cost estimation and the comparison of alternatives.

6.1.5 Measuring Technical Debt Awareness with TD-SAGAT

A literature review revealed that many research publications have addressed TD awareness [MB15]; [Ton+17]; [Rio+20]; [Ram+21]. Most studies mention TD awareness as a problem that needs to be addressed [Bes+17]; [KNO19]. However, some studies claimed that their approach increased TD awareness. To evaluate TD awareness, researchers commonly use retrospective questionnaires [Ton+17]; [Ber+21], which are subject to bias.

To improve awareness measurement, we identified and described a method for measuring TD awareness, adapted from the psychology domain, i. e., SAGAT (Situation Awareness Global Assessment Technique). We call the adaptation of this method to TD, TD-SAGAT. The basic idea of SAGAT is to interrupt participants in their work with short, rapid questionnaires that ask for the prerequisites for rational decision-making in these situations. In our case, we interrupted practitioners during meetings, in which they discussed the implementation approaches for their tasks. Further details on TD-SAGAT and the rationale for it were already explained in Section 4.2.2.2.

As part of SAGAT adaptation, we systematically identified the underlying prerequisites for TD awareness, which researchers can now apply in research scenarios beyond meeting interruptions, e. g., during observations. Moreover, the interruption can be applied in various research-relevant situations, e. g., during code reviews, thereby making this method a contribution to the TD and the software engineering communities. In our action research studies in Step III of this thesis, we used TD-SAGAT to assess participants' TD awareness and related the results to the workshop topics and TDM process changes.

6.1.6 Awareness Effects of the Technical Debt Management Establishment

As noted in the previous section, a few studies measured TD awareness using retrospective questionnaires to gather participants' general TD awareness [Ton+17]; [Ber+21], but most studies reported effects on TD awareness. Additionally, most studies do not clearly define the ambiguous term "TD awareness". For example, some studies use the term to mean knowing that TD exists and how it is defined [Apa+20]; [Bes+17]; [Hol+18], others interpret the term as knowing about the TD in their own system [Bes+17]; [Liu+20], and others mean by TD awareness recognizing unintentional TD incurrence and avoiding them [RZN17]; [Wie+22].

As part of the action research studies in Step II of this thesis, we evaluated whether establishing the TDM process increased TD awareness. In alignment with the SAGAT definition, we clearly defined our understanding of TD awareness as the perception of the prerequisites (e. g., alternatives, drawbacks, interest) of a decision (e. g., incurring TD) during decision-making situations (e. g., refinement meetings), the comprehension of these prerequisites' meaning, and the projection of their status in the future (e. g., TD's consequences). We used the TD-SAGAT prerequisites to gather data across three data-collection methods (general questionnaires, TD-SAGAT interruptions, and observations).

As a result, we found that the TD Game from publication No. 5 *(TD Game)* increased awareness only right after the event, making it effective only as an initial impetus. This became especially apparent because all teams initially failed to consider the evaluation of alternatives, drawbacks, and risks associated with non-TD issues. Only after we included the attribute fields for non-TD issue types did the team's awareness of these prerequisites, and thus a potential TD incurrence improve. Feedback from practitioners, observed challenges, and measured TD awareness prerequisites further suggested that a verbal reminder from the researchers was necessary at the beginning of the process to help internalize the use of these attributes.

For researchers, we demonstrated that measuring TD awareness is important to identify the source of changes in TD awareness. This implies that a clear separation of the changed variables (in our case, achieved through an iterative approach) and the measurement of each change's impact on TD awareness are necessary to accurately assess TD awareness. The results further indicate that investments in training and consulting appear necessary to sustainably establish a TDM process. Future research should examine whether this external support is necessary and, if so, how to modify the approach to eliminate this need.

6.1.7 Uncovered new Research Areas

The studies contributing to this thesis identified research gaps and promising directions for TDM research. These research gaps show issues that need to be addressed for practitioners to use a TDM process. Thus, they are potential reasons why TDM is not widely adopted. Consequently, they are directly related to the research goal and a contribution to this thesis, rather than merely the subject of future work. Some of these gaps were addressed in this thesis, for example, by a method for establishing TDM within a team or by improving TD awareness and communication through a game. However, other research gaps remain or require deeper investigation in the future. We present initial ideas for addressing these gaps, but the final analysis and evaluation of these ideas is outside the scope of this thesis.

6.1.7.1 Status-Quo Bias in Technical Debt Management

In publication No. 3 *(IT Managers' Perspective)*, IT managers showed a contradiction between their assessment of TD as an important topic and their actions, i. e., not managing TD. This may be a phenomenon linked to status quo bias, a cognitive bias characterized by an irrational preference for maintaining existing conditions [KKT91]. The status quo bias can lead individuals to favor the default option of not making any changes. IT managers play a vital role in TDM, as establishing a TDM process and engaging with business stakeholders are nearly impossible without their support. While all IT managers had some level of influence over time or budget, the majority did not associate their management of these essential factors with TDM.

We assessed that to overcome status quo bias, IT managers need more guidance on establishing a TDM process, preferably by leveraging existing processes and simple methods. We suggested investigating how to establish a TDM process effectively, rather than focusing solely on its structure and activities, which motivated RQ4 *(Establishment)*. The workshop concept and the resulting TDM Guide were created to fill this research gap.

6.1.7.2 Technical Debt Identification across Stakeholder Levels

As explained in the foundations in Section 2.6, we initially used the Dagstuhl definition of TD [Avg+16a] and the backlog color scheme by Kruchten et al. [KNO19] to guide participants in deciding whether an issue is a TD item. However, during the action research studies, we repeatedly encountered the challenge that practitioners still struggled to determine whether an issue should be classified as a TD item. To support them, we developed the simple decision matrix outlined in Section 2.6, which is based on the questions of who is suffering from the issue and who is willing to pay for it. To address further ambiguities, we developed an extended matrix that includes multiple stakeholder or organizational levels, a flowchart, and a list of indicators for TD, all presented in the TDM Guide. The resulting flowchart and the list of indicators are straightforward and introduced as part of the practical contributions in Section 6.1.7.2. However, the extended matrix, depicted in Figure 6.2, visualizes an issue that warrants further research.

Who is willing to pay? \ Who suffers?	Your Customer	Your Company	Your IT unit	You (Your Team)
Your Customer	No TD	(Probably no) TD	(Probably no) TD	(Probably no) TD
Your Company	No TD	(Probably no) TD	Probably TD	Probably TD
Your IT unit	No TD	Probably TD	Probably TD	TD
No one	No TD	Probably TD	TD	TD

FIGURE 6.2: Matrix for Identifying TD Items across Stakeholders [WSB26], Chapter A

In a corporate setting, what may be regarded as a TD item by one stakeholder may not be recognized as such by another. The question remains whether to add such an issue to the backlog as a TD issue or as a functional requirement, for example, if the TD is at a higher abstraction level. An example is switching to a new development tool (e. g., another issue-tracking tool). For a company's IT department, this might be considered a TD that should be repaid to reduce costs (e. g., license fees) or to standardize the company's tool landscape. However, at the team level, this causes effort, whereas no interest arises if the team stays with the old tool. Similarly, directions from a company, e. g., to switch to open-source software to become more vendor independent, might be considered TD by the business but may not generate interest within the IT department or IT teams.

We identified an area of TD research that, to the best of the authors' knowledge, has not been investigated to date. TD items can occur at different levels of a company hierarchy. It remains unclear whether they can always be addressed in the same way, and how a TD item at one level is perceived by another. As a logical consequence, we need to investigate how TD items can be integrated across different levels.

6.1.7.3 Debiasing Decisions on a Design Level

We used the debiasing workshop from Publication No. 9 *(Debiasing)* as one element of our workshop design [BAW25]. However, this workshop is directed toward architects and their decision-making processes. In the publication No. 7 *(Practical Guide)*, we suggested conducting further research on debiasing decisions at a lower level of abstraction, e. g., at the design or even implementation level. Specifically, we should develop concepts to support the identification of alternative solutions, given participants' reported difficulties with this task. As

a first measurement, we reordered the attributes in the backlog issue. Instead of starting by defining alternatives and then recording their respective benefits, drawbacks, and risks, we reversed the order and ended with the alternatives. We guided participants to first identify quality attributes affected by the initial solution, and, from this, to derive a drawback or risk if a negative effect was identified. The search for an alternative solution should then focus on mitigating these drawbacks or risks. This order made it easier for participants to understand the abstract concept of an alternative solution.

We identified the relevance of presenting attributes to practitioners in a specific order, which may be similarly helpful for analysis in other research areas, i. e., (practical) situations. Further research in this area could examine how other methods, e. g., supported by LLMs, can guide the identification of alternative solutions.

6.1.7.4 Dealing with Multiple Interests

The term *interest* was introduced as part of the metaphor and describes the extra effort required if a TD item is not repaid [Guo+11]; [McC08]; [LAL15]. However, all publications on TD interest treat it as a single instance and do not address the real possibility that one TD item can cause multiple interests. For example, a single TD item, such as unreadable code, may complicate future changes, increase the risk of error, and cause performance problems at the same time. As another example, not updating third-party libraries often led to multiple effects: increased risk of bugs, inability to use new features that might make development easier and faster, security vulnerabilities, and performance issues. In the publication No. 7 *(Practical Guide)*, we describe this issue of dealing with multiple interests. We pose the question of whether practitioners should focus on a single interest (e. g., the most impactful) or whether all interests should be estimated and then combined into an overall interest. In our studies, participants evaluated the most impactful interest and mentioned other interests in the attribute for *TD consequences*. However, this raises a new issue, i. e., how to identify the most impactful interest. Participants relied on their intuition because they were not able to invest the additional effort to systematically evaluate all interest attributes across all interests. However, this solution is again prone to bias.

Addressing the issue of multiple interests per TD item warrants further investigation. With this, we identified an area of TD research that, to the best of the authors' knowledge, has not been investigated to date.

6.1.7.5 Scaling the Workshop Concept

Often, case studies report on TDM approaches at the team level, and only a few studies have examined how such a process might be scaled for larger companies. Detofeno et al. performed an action research study to implement a TDM process in a Brazilian company, starting on an organizational level by establishing a TDM guild, i. e., a CoP, on the TD topic [DRA21]. However, this study focuses on a PHP code debt, which makes it hard to transfer the results. Martini et al. describe the CAFFEA framework [MB17] for scaling an agile architecting framework (that includes TD management) to large companies. Still, this framework is inflexible and imposed on the organization from the top down. Overall, the workshop concept described in publications No. 6 *(Establishing TDM)* and No. 7 *(Practical Guide)*, which develops a TDM process from the bottom up, is not yet scalable for large companies. Organizing workshops for all teams within a company would incur substantial costs. Moreover, this workshop approach would enable each team to create its own processes, making it impossible to evaluate them across the entire company. Therefore, it is crucial to identify strategies to improve the approach's scalability. Ideas and preliminary results to solve this issue are discussed as part of our future work in Section 9.2.

As with TD identification across organizational levels, research on TDM approaches should specify the organizational level under study. Furthermore, each TDM approach should include information on how it can be integrated with approaches at other levels or scaled.
For the thesis's approach, we discussed potential scaling strategies as future work, but their evaluation falls outside the scope of this thesis.

6.1.8 Proposals for Improving Research Applicability in Practice

The action research studies documented in publications No. 6 *(Establishing TDM)* and No. 7 *(Practical Guide)* revealed several instances in which the theoretical concepts regarding TD were too abstract for practitioners to grasp and implement. For instance, while the ideas of *interest probability* [Guo+11]; [LAL15] and *contagiousness* [MB15]; [MB17] were generally clear during discussions, they were fully comprehended only after we introduced a guiding question (e. g., 'Does the principal rise over time?' for *contagiousness*) and changed the term *interest probability* to *interest frequency*. Additionally, we encountered challenges with defining the content of an attribute. Although the terms *interest* and *interest probability* were widely understood, their practical application required participants to discuss the content of these issue attributes, specifically the details of the combo boxes. The identification and formulation of categorized values for the attributes took some time and detours (e. g., using a low to high scale or T-shirt sizes (XS to XL)) before the final time-related categorization (e. g., interest = {15 min., 1 hr, 4 hrs, 8 hrs, $>$ 8 hrs.}) was determined.

One conclusion of this thesis is that researchers need to provide more specific instructions for practitioners for research to be applicable in practice. This can be done, for example, by providing application notes in the supplementary materials of research papers that claim to benefit practitioners. For example, in our publication No. 4 *(Learning from each other)*, we claimed a practitioner's benefit but provided no concrete instructions for applying our results in practice. We optimized this in publications No. 6 *(Establishing TDM)* , where we included the workshop instructions in the additional material. We further improved it in publication No. 7 *(Practical Guide)*, where we added a detailed TDM Guide outlining the TDM process and its implementation for practitioners.
This thesis also underscores the importance of not only evaluating predefined artifacts in practice or observing practitioners (e. g., through case studies [Run+12]) but also developing processes or artifacts in collaboration with practitioners (e. g., through action research [Sta20] or design science research [Pef+07]; [HC12]) to uncover these practical challenges.

6.1.9 Improving the Collaboration with Tool Vendors

Most IT teams organize their work in issue-tracking systems, i. e., they maintain a backlog of issues to be worked on in prioritized order. Accordingly, the team's existing issue-tracking systems seemed to be the natural tools for managing TD. The Dagstuhl Manifesto, resulting from the "Reframing Technical Debt" seminar, defined this as a principle: *"P5 - Build Seamless and Integrated Technical Debt Management Toolchains with Human Oversight"* [Avg+25]. However, in our action research studies reported in publications No. 6 *(Establishing TDM)* and No. 7 *(Practical Guide)*, we encountered multiple problems in adopting the issue-tracking tool.

First, some tools were not customizable at the team level, necessitating company-wide negotiations over changes. Second, it was difficult in these proprietary tools to populate an attribute based on a rule or add value calculations. For example, we implemented a *return on investment* calculation in Jira. However, this was extremely complex due to the rule-based nature of the calculations. Third, the standard visualizations in the issue-tracking tools were inadequate for our purpose: Across all teams, we had to use an additional tool for visualizations, which has several disadvantages. In addition to the effort required to implement and maintain this component, we observed that teams either avoided using it or simply forgot to use it in their regular work. Fourth, some requested visualizations were

infeasible. For instance, participants aimed to visualize the total *interest burden* (= *interest* * *interest probability*) for all open issues per month to evaluate the effectiveness of the TDM process. We successfully presented the *interest burden* of issues that were either opened or closed at a certain time. To establish a timeline, we had to iterate over all previous months and issues to determine whether an issue was opened or closed in this month, necessitating an additional program. We chose not to implement this because our teams could not commit to the necessary long-term maintenance effort.

There is considerable potential to improve professional issue-tracking tools, particularly in supporting calculations and visualizations. We suspect that the technical challenges presented may be one reason practitioners prematurely abandon TDM approaches. A solution would be for the issue-tracking tool vendors to integrate TD issue types by default. Close collaboration between TD researchers and tool vendors should be sought to optimize issue-tracking tools for TDM.

6.2 Practical Contributions

The theses yielded multiple practical contributions, which are explained in the following Sections as summarized in Table 6.2.

Section	Title	Contribution
Section 6.2.1	TAP Framework – A new Technical Debt Repayment Strategy	A repayment strategy suitable for time-critical situations as part of the TAP framework of four different maintenance tasks
Section 6.2.2	3-Minute Pitch based on the V4CTD Model	An application proposal for the V4CTD model resulting in one slide for a management pitch
Section 6.2.3	Communicating about Software Architecture Mistakes	Proposals on how to improve communication of software architecture mistakes, including factors that facilitate or hinder communication and suggestions to ensure communication opportunities
Section 6.2.4	TechDebt Game	The TechDebt Game to understand TD concepts and improve communication
Section 6.2.5	Workshop Concept	The workshop concept for establishing TD management, including an overview of how other contributions informed the workshops
Section 6.2.6	Technical Debt Management Guide	The TDM Guide, including TD Identification models, a TD prevention method, attributes relevant for a TD item issue in a backlog, calculations of KPIs, the use of a resubmission date, TD Backlog Visualizations, and finally, a step-by-step introduction on how to establish a TDM process

TABLE 6.2: Overview of practical contributions

6.2.1 TAP Framework – a new Technical Debt Repayment Strategy

Many strategies for repaying TD were discussed in research, e. g., repayment of TD that impedes a functional requirement [Fre+23], repayment if a cost calculation proves a positive effect [McC08]; [TAV13], or repayment based on the system's evolution [Sch13a]. Similarly, many TD prevention methods have been discussed [YHMS16]; [Apa+20]; [Tso+20]; [Bor+21b]; [DMR22]. However, these methods overlook the fact that oftentimes tight deadlines cause the incurrence of TD and also motivate stakeholders not to repay TD, which is one focal point of publications No. 1 *(TAP Framework)* and No. 2 *(TAP Framework – extended)*. In the case team's framework, the term *TD issue* refers specifically to intentional TD, in alignment with Cunningham's original concept of TD [Cun92]. Other related concerns are categorized as deconstruction issues, maintenance issues, or maintenance projects, each linked to its specific repayment approach. Accordingly, a structured approach for repaying intentional TD was determined.

When faced with a tight deadline and multiple solution options – one adhering to standards and others being sub-optimal but quicker to implement – the team records two issues. One issue captures the implemented sub-optimal solution that incurs TD, while the other outlines the repayment approach for the optimal solution. These options are typically identified during refinement meetings.

The TAP framework includes the repayment of these intentionally incurred and, as such, recorded TD issues in the project plan. It ensures that no project concludes until these TD issues are addressed as presented in Figure 6.3. This shifts responsibility for TD repayment to project management, fostering a culture in which project completion aligns with TD management.

The benefits of this approach include heightened awareness of intentional TD, reduced unintentional TD accumulation, and the accountability of project managers for TD. This creates a feedback loop (see Figure 6.3) that encourages behavioral changes that lead to TD prevention through conscious decision-making. Additionally, both project and IT managers can monitor TD accumulation in real time, enabling timely interventions and adjustments.

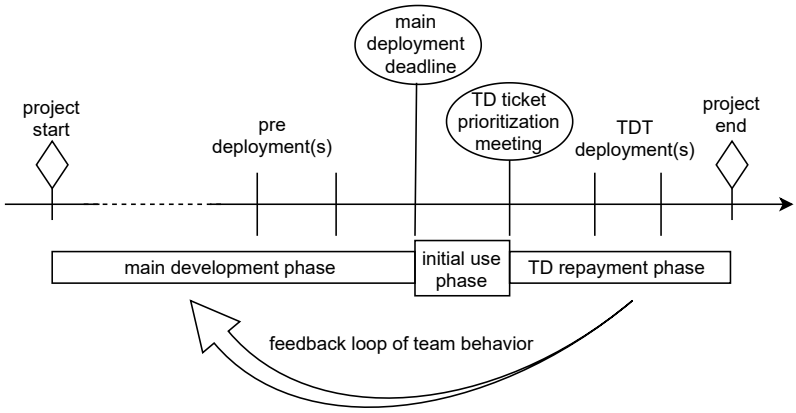


FIGURE 6.3: Project Plan with TD Repayment Phase and Feedback loop [Wie+22]

The introduced repayment approach can be adopted by practitioners facing tight deadlines. The approach can be adapted, as the deadline need not originate from a project but may, for example, stem from the deployment deadline for a feature. We introduced this method as part of the workshop concept during the workshop on TD repayment.

6.2.2 3-Minute Pitch based on the V4CTD Model

As explained in the introduction, communicating the issues emerging from unrepaid TD is crucial to negotiate funding for its repayment with IT and business managers [GSS16]; [RK19]. Even the origin of the TD metaphor stems from the need to communicate these issues to business stakeholders [Cun92]. In publication No. 3 *(IT Managers' Perspective)*, the IT managers similarly complained about the difficulties of identifying and subsequently explaining those issues. Guided by the interviewees' requests and based on the V4CTD model, we developed a guideline to create a *3-Minute Pitch* (also known as *elevator pitch* [MW21]; [Dan+25]) for IT managers to explain the TD-related issues within their teams or companies.

IT managers, supported by their developers, may use the model and the categories of causes and consequences (Tables 2.2 to 2.3) to examine and pinpoint their team's chains and vicious cycles of causes and consequences. Then, they can transfer these to the V4CTD model shown in Figure 2.8 and create an overview slide to illustrate and debate their TD-related issues with business stakeholders.

Figure 6.4 illustrates one possible instance of such a slide. In this example, time pressure and a shortage of experienced developers lead to unstable systems, business disruption, and developer attrition. Workarounds that lead to unreadable code create an inner vicious cycle in which the unreadable code leads to more faulty decisions (i. e., developer mistakes) and an increased tendency to create further workarounds. The lack of developers (cause) and

developer attrition (consequence), as well as time constraints (cause) and time to fix bugs and repay TD (consequence), create outer vicious cycles.

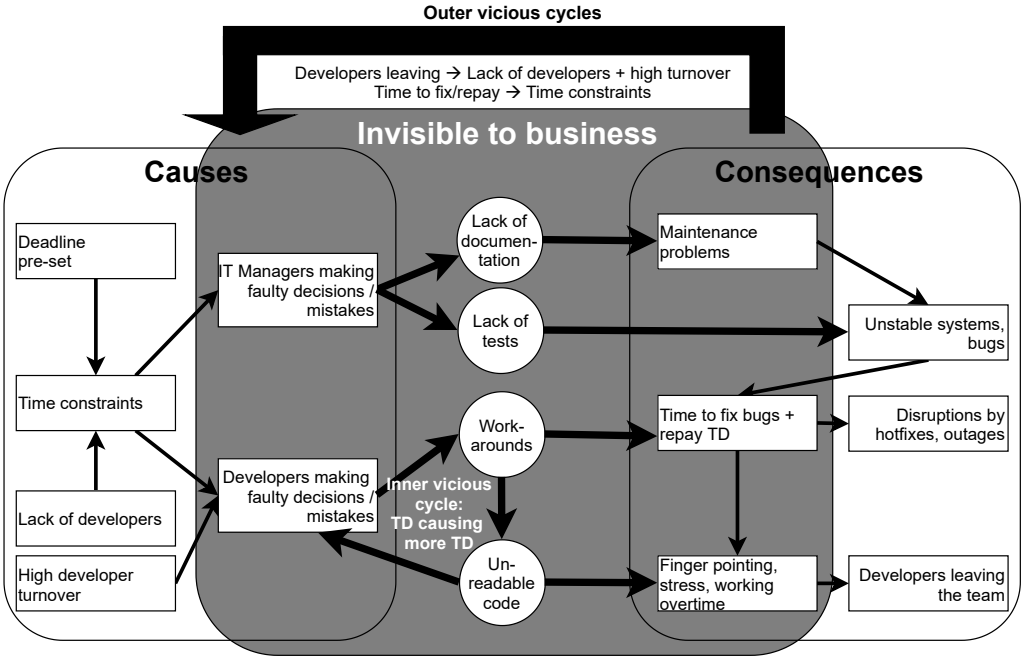


FIGURE 6.4: Example of a 3-Minute Slide [WB23]
(thin arrows = chains of causes/consequences;
thick arrows = cause-effect relation)

This contribution provided a method to communicate tangible TD-related issues to non-technical stakeholders. Practitioners can use it to create an overview of their team's chains and vicious cycles of causes and consequences. The overview can support the negotiations with business stakeholders as well as the prioritization of repayment efforts (e. g., by first breaking up vicious cycles).

6.2.3 Communication about Software Architecture Mistakes

Guided by the same communication questions that motivated the previously described contribution, i. e., the difficulties to convey issues invisible to business stakeholders [KNO12], we analyzed how architects communicate their mistakes, i. e., decisions that proved suboptimal. From this study reported in publication No. 4 *(Learning from each other)*, we derived factors that facilitate or hinder communication about a delicate topic, such as architectural mistakes. Moreover, the study reported concrete ideas for improving the management of software architecture mistakes in practice: *Increasing Detection Triggers* and *Ensuring Communication Opportunities*. The factors and the explanation of ensuring communication opportunities are relevant to this thesis and are explained in the paragraphs below.

Communication-Facilitating and -Hindering Factors The factors that facilitate or hinder communication about architectural mistakes are listed below and further explained in the original publication No. 4 *(Learning from each other)*.
Facilitating factors for SAM communication are:

- Access to information, i. e., transparent communication
- Diverse teams in terms of cultures and expertise
- Supportive communication structures

Hindering factors are:

- Potential negative consequences

- Missing psychological safety
- Discrepancies in reputation, status, or hierarchy of participants
- Missing professional status and self-confidence
- Emotional biases between participants
- Conflicts of interest, e. g., regarding project- and business goals

Practitioners can use the results to create an environment that facilitates open communication on a sensitive topic such as TD, e. g., by discussing the topics within a team structure to counteract discrepancies in status and hierarchy, utilizing external moderation to counteract monopolization caused by discrepancies in reputation, disclosing and openly discussing conflicts of interest, supporting diversity, when hiring, and providing all participants with the same information through sharing protocols, slide decks, or recordings of critical meetings.
We drew on these inspirations for our moderation approach during the workshops, particularly the first workshop, and shared all workshop slides, including the measures and responsibilities slide.

Ensure Communication Opportunities WE found that companies that plan to enhance communication should enhance communication opportunities among their employees, for example, by creating company-wide communities of practice or cultivating regular cross-team communication. Furthermore, we identified a need for greater openness about software architecture mistakes within and between companies. For example, governments or chambers of commerce could support this information exchange across companies.

We provided insights that organizations can adopt directly to improve their communication about (A)TD, e. g., by installing a CoP on software architecture mistakes or TD to foster communication and, thus, learning opportunities. We further suggested measures that are more difficult to adopt but might be worth pursuing, e. g., changing personal mindsets about software architecture mistakes or cross-company exchanges about them. This part inspired the creation of a CoP on TD to scale the resulting TDM process across a company's teams as explained as future work in Section 9.2.

6.2.4 TechDebt Game

In his backlog color scheme, Kruchten et al. [KNO12] identified that TD items are typically invisible to business stakeholders. Similarly, the V4CTD model of publication No. 3 *(IT Managers' Perspective)* depicts this visibility issue.

To overcome this issue and provide all stakeholders with a safe space to experience TD and discuss its effects, we developed the TechDebt Game, which we evaluated for publication No. 5 *(TD Game)*. The board game includes a PDF file for printing the board and cards, which must be supplemented with game pieces and dice. The online version is based on Tabletopia, a platform for transferring board games to the online environment [Tab]. The game is played in two teams of at least two players each. The goal is to develop a system consisting of three modules, with one architecture element and three features each. Finalized features lead to new customers, and a high number of customers is the final goal of the system and, thereby, the game. Players are allowed to incur TD to speed up the development. However, after speeding up one feature, the TD hinders the development of subsequent features. Additionally, there is a mechanism to incur TD unintentionally. Action and event cards add extra thrill to the game and give players the opportunity to gain further TD insights. Figure 6.5 shows an overview of the board game.

A detailed instruction is available with the game or in publication No. 5 *(TD Game)*. Both the online and offline games are available in German and English. The board game is available on GitHub [Tecc] and in an online version on Tabletopia [Teca]; [Tecb].

The evaluation showed that business stakeholders and junior developers gained the most new insight, while senior developers identified the game as a realistic emulation. Junior developers accepted that ignoring TD might be an option and were better aware of the issues arising from TD. Accordingly, they planned to be more careful not to incur TD and showed

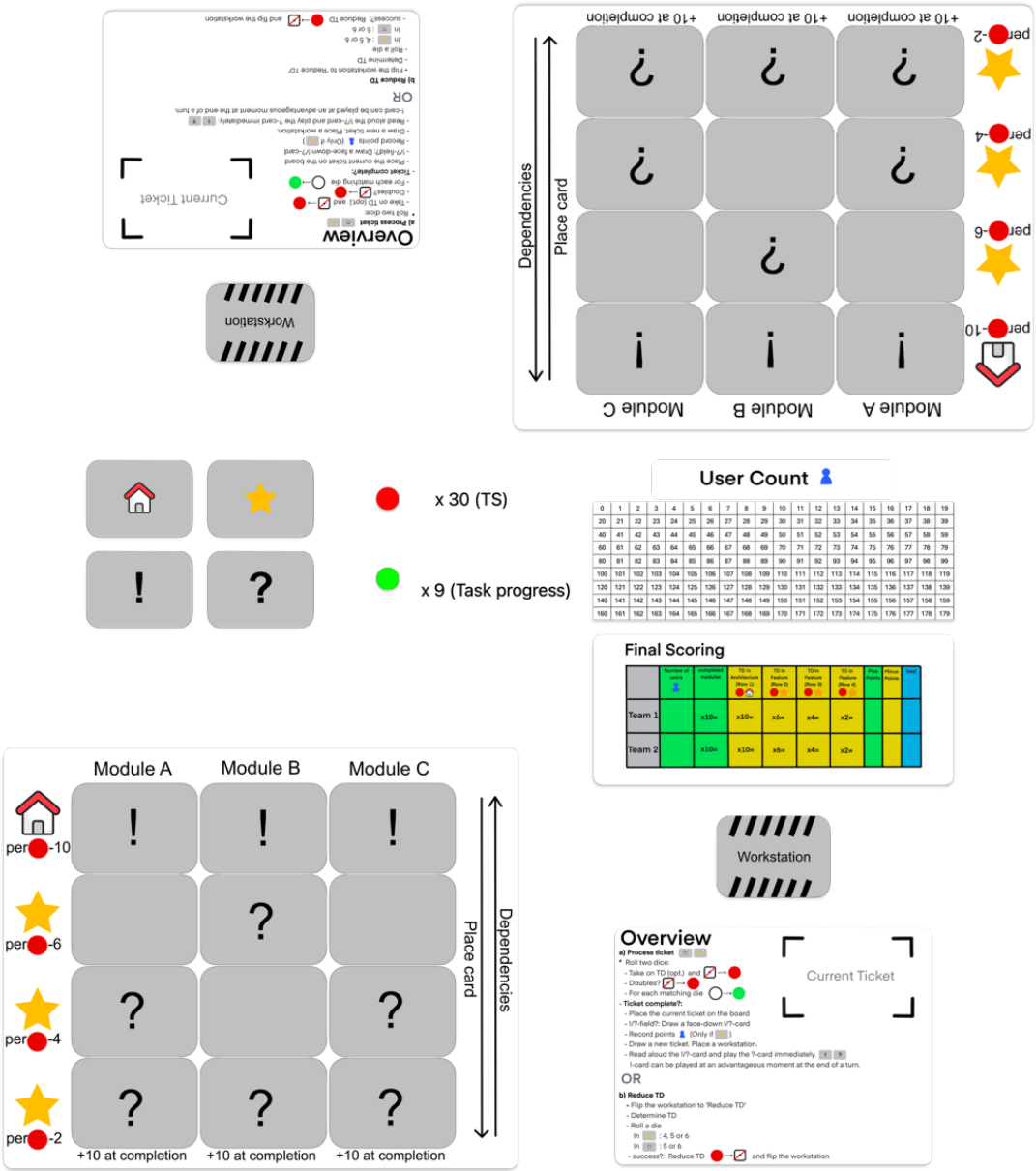


FIGURE 6.5: Overview of the Game Board [Wie+25]

a better understanding of senior developers after playing the game. Business stakeholders learned more about the TD concepts and acknowledged that there may be issues invisible to them. Thus, they were more inclined to trust the developer's assessment. All participants were open to changing their attitudes and behaviors regarding TD, making the game a good starting point for establishing TDM.

Practitioners can use the game to convey the problems TD creates to non-technical stakeholders and junior developers. The game shifts the discussion from real life to an emulated environment, thus avoiding the harshness of discussing real-life problems and providing a psychologically safe space. Accordingly, we played the TechDebt Game as an initial element of the workshop concept.

6.2.5 Workshop Concept

The overall goal of this thesis was to develop a method to bridge the gap between research and practice. While many research approaches have been developed, their adoption in practice is still rare [MBB18]. Furthermore, the researched approaches commonly focus on specific TD activities or TD types, missing a holistic view [RMS18]; [JT22]. The recently developed Dagstuhl Manifesto for Technical Debt [Avg+25] identifies overcoming these issues and fostering a close collaboration between researchers and practitioners as a key goal.

The workshop concept, which we developed as an artifact of Step II of this thesis, was designed to bridge this gap. Section 4.2.2.1 already detailed the workshop concept as an element of this thesis's research strategy. The additional materials to both publications No. 6 *(Establishing TDM)* and No. 7 *(Practical Guide)* include more details on this workshop idea.

Below, we detail how contributions from other thesis publications informed the workshops. Figure 6.6 shows an overview of how publications are related to the workshop concept.

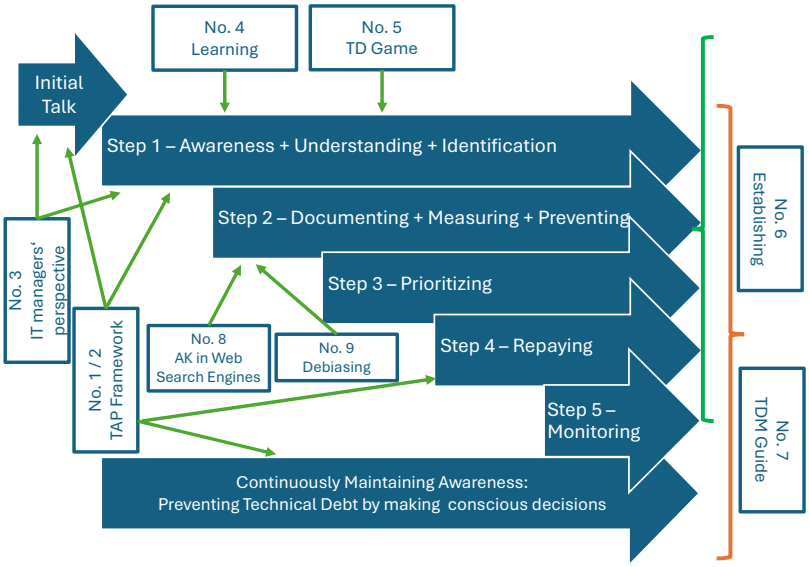


FIGURE 6.6: Publications Contributing to the Workshop Concept
(green – contributions and inspirations; orange – results)

Initial Talk The initial talk given prior to the workshops is designed to establish common ground on TD knowledge and to introduce TD-specific terminology to facilitate future discussion.

This talk includes the results of publication No. 3 *(IT Managers' Perspective)*, more specifically, the V4CTD model, because it provides a comprehensive overview of the concepts, categories, and intricacies of TD. The publications No. 1 *(TAP Framework)* and No. 2 *(TAP Framework – extended)* highlight the importance of discussing the various cost factors of TD items, which is why information on cost factors is provided. The section on the backlog organization changed over the execution of the workshops. For the first action research, the talk introduced the TAP framework from the publications No. 1 *(TAP Framework)* and No. 2 *(TAP Framework – extended)*. As this is only one of many TD repayment methods from the participants' perspective, the two subsequent action research studies provided the results of the first study, i. e., an example of a TD issue type, described in publication No. 6 *(Establishing TDM)* instead.

Workshop 1 – Kickoff – Awareness & Understanding The first workshop focuses on creating a sociable, psychologically safe environment as a foundation for future workshop discussions. This workshop was consequently inspired by the publication No. 4 *(Learning from*

each other), which underscores the relevance of creating a safe environment when discussing potential mistakes made in earlier days.

Publication No. 3 *(IT Managers' Perspective)* introduced the communication concern across multiple stakeholders, which is why the first workshop started with the TD Game from publication No. 5 *(TD Game)*, which was specifically designed for this purpose. The game provides stakeholders with a means to deepen their understanding of TD and enhance communication. Adhering to the workshop's purpose of creating a psychologically safe environment, the game bypasses heated real-life discussions and enables more objective discussion in a less emotionally charged environment. For this part of the workshop, we specifically invited all interested stakeholders, including those outside the team, to participate to optimize communication across teams and organizational levels.

After the game, we asked the participants to discuss their recent TD items, the TDM approach they have used so far, and their goals for the process and the upcoming collaboration. This follows the suggestion of the publication No. 4 *(Learning from each other)* to create opportunities for communication, and it further fosters a safe and open communication environment. At the same time, the discussions provided the researchers with valuable information and defined the starting point for evaluating success.

Workshop 2 – Prevention & Documentation The second workshop introduces research on TD prevention, i. e., debiasing decision-making and using ISO-250101 quality attributes to guide decisions. This workshop encloses the debiasing workshop slides from publication No. 9 *(Debiasing)* to introduce the topic of biases and debiasing, highlighting the importance of comparing solution alternatives, including their drawbacks and risks. Publications No. 1 *(TAP Framework)* and No. 2 *(TAP Framework – extended)* further underline the relevance of comparing solution alternatives. Additionally, we note the difficulty of obtaining information on drawbacks and risks from the internet, which stems from publication No. 8 *(AK in Web Search Engines)* .

The second workshop additionally aims to provide an initial idea for organizing the backlog, e. g., by introducing a TD issue type. In this regard, the workshop revisits the information on TD cost factors from the initial talk and encourages discussion of backlog organization. This information on cost factors is partially based on the results of the publications No. 1 *(TAP Framework)* and No. 2 *(TAP Framework – extended)* regarding the inclusion of principal and interest factors. During the last two studies, we also presented the resulting TD issue type from publication No. 6 *(Establishing TDM)* as inspiration.

Workshop 3 – Prioritization The focus of the third workshop is on prioritizing TD. This workshop is less influenced by the publications underlying this thesis. However, we included the results from publication No. 6 *(Establishing TDM)* in the subsequent studies published in No. 7 *(Practical Guide)*. Specifically, we introduced the idea of the resubmission date and the approaches of prioritization by an educated guess or an ROI calculation developed during this first action research study.

Workshop 4 – Repayment In this workshop, participants get information on seven different repayment approaches. In publications No. 1 *(TAP Framework)* and No. 2 *(TAP Framework – extended)*, we introduce a repayment approach that repays TD after a fixed deadline for delivering a feature, but still as part of the project or feature implementation. This was one of the seven approaches.

Workshop 5 – Visualization The literature review revealed no transferable visualization approaches for TDM. Consequently, we focused on developing more general visualization ideas based on visual storytelling for publication No. 6 *(Establishing TDM)*. We used the results of publication No. 6 *(Establishing TDM)* as inspiration in the fifth workshop of the subsequent action research studies.

Scaling the TDM Process on Company Level After establishing the TDM process within a team at a company, the task arose to scale it to other teams within the same company. The results of publication No. 4 *(Learning from each other)* inspired us to use a CoP for this purpose

in one of the collaborating companies. More ideas for scaling the TDM process, not directly building on the previous publications, are presented in Section 9.2 as future work.

Practitioners can employ the workshop concept, using the provided slides, agendas, and explanations in these publications for developing their adapted TDM process. However, we found that external support, e. g., from a researcher or consultant, may be necessary for the establishment's success.

6.2.6 Technical Debt Management Guide

Similar to the previous section, the TDM Guide was designed to bridge the gap between research and practice, as, e. g., presented in [MBB18]; [Avg+25], by providing a holistic, easy-to-understand, and easy-to-adopt framework for managing TD.

To support practitioners beyond the workshop concept with condensed information on how to start a TDM process, we developed the TDM Guide. The TDM Guide has been published as additional material for the thesis's final publication No. 7 *(Practical Guide)*. The guide has not been evaluated as part of this thesis, but it is the result of the findings, and evaluation might be part of future work as introduced in Section 9.2.

The suggestions in the TDM Guide emerged from comparing the adopted approaches and the challenges we observed across all teams. If an approach was adopted similarly by all three companies, it was included in the TDM Guide as *best practices*. Approaches adopted by only one or two companies were included as *nice-to-have*, i. e., the adoption seems to be context-dependent. The Merriam-Webster dictionary defines *best practices* as *a procedure that has been shown by research and experience to produce optimal results and that is established or proposed as a standard suitable for widespread adoption* [Bes]. Using the term *best practices* is controversial in research, as some researchers argue best practices should emerge from practice or be proven by multiple adoptions. However, the definition states that best practices can stem from research and (practical) experience, and may be a proposal, both of which align with the presented approach. Moreover, practitioners are familiar with the term and often have a different perspective: needing guidance, they are pleased to receive clear instructions grounded in prior experience. Specifically, they recognize that best practices are merely advice and can be adapted to their needs. Accordingly, we used this term in the TDM Guide.

The list below presents a short overview of the ***best practices*** derived from our studies.

1. Every team designated a TD champion or manager to guarantee compliance with the process and to promote communication regarding TD matters.
2. The teams identified TD primarily by evaluating who"suffers" from the TD item and who would be responsible for its repayment (see Section 2.6.1). The teams employed automatic TD identification only as an auxiliary approach.
3. The teams used their issue-tracking tool (backlog) to label TD items. They developed a distinct TD issue type or template that captures various attributes of their TD items. Typical attributes for a TD item include the TD's *interest*, *interest probability*, *principal*, and *contagiousness*. To facilitate ongoing and systematic (re-)prioritization, they established a *resubmission date*. Although the teams assessed these attributes in different ways (e. g., 0 to 5, very low to very high), they offered similar explanations for categorizing genuine values.
4. To prevent TD, the teams incorporated attributes into all types of backlog issues to prompt discussions about alternative solutions, their drawbacks, and risks. Good practices aimed at avoiding TD (e. g., code reviews) were already integrated into the team's standard work procedures. Some teams enhanced these procedures by adjusting meeting formats or employing "Scrum Poker" [MH12].
5. To assess the TD items and determine their prioritization, each team employed a unique method: An *educated guess*, a *return on investment (ROI)*, and a *mean attribute priority*. (We were not able to derive a best practice for this activity.)
6. The teams chose to prioritize TD relative to one another, avoiding competition with functional requirements. They also decided to continuously (re)assess the prioritization of TD items based on the system's evolution, using the *resubmission date*.

7. To repay TD items, teams used various strategies based on the circumstances. Common methods included paying interest (high effort, low priority), addressing "low-hanging-fruits" (low effort, high priority), repaying TD items linked to upcoming functional changes, or combining TD repayment with functional requirements.
8. None of the issue-tracking systems provided adequate support for visualizing data, forcing us to use external tools for creating visualizations of backlog issues. Despite this limitation, the resulting visualizations were similar.

The subsequent paragraphs provide additional details on intriguing new contributions from the TDM Guide. These details have been published in a condensed version in publications No 6 *(Establishing TDM)* and No. 7 *(Practical Guide)*. However, this section will include more detailed descriptions from the TDM Guide (Chapter A, [Wie26]) published as additional material to publication No. 7 *(Practical Guide)*.

Chapter A presents the complete Whitepaper *"A Practical Guide to Establishing Technical Debt Management"*, written specifically for practitioners and reviewed by six practitioners from our collaborating companies, who were not part of the team that established the TDM processes. Practitioners can use this guide to directly derive a TDM process and to identify how to establish it incrementally. The inclusion of *nice-to-have* approaches allows them to adapt the final process to their contexts.

6.2.6.1 Technical Debt Identification

Sections 2.6.1 and 6.1.7 already introduced that practitioners in the action research studies had difficulty identifying whether an issue is a TD item. These sections also introduced the matrices of Figures 2.9 and 6.2, which we provided to practitioners to support their decision-making and to outline the problem across different stakeholder levels. In response to the practitioner's request, we further developed a decision flowchart and provided a list of indicators for TD, i. e., spoken phrases or situations that suggested a TD issue. Figure 6.7 presents the flow chart.
The list of TD indicators presents situations or phrases in a discussion that may suggest TD. However, they do not result in fixed classifications, but rather clues to consider whether the issue could be TD. The list is open for extension and comprises the following so far:

- The use of certain terms in the title of the issue. The following terms (and their linguistic variations) are indicators:
  - *Optimizing, improving, revising, testing, documenting, performance, upgrading, updating, refactoring, cleaning up, converting*
- Using certain phrases when discussing the issue:
  - "We did it quickly back then."
  - "We shouldn't have done it that way."
  - "Now we want to do it right."
  - "We implemented it hard-coded at first."
  - "Now we're doing it right."
  - "I forgot to expand it back then."
  - "At the time, we thought that would be good."
  - "It turned out that this is not optimal."
  - "Special cases would be eliminated if we implement this."
  - "I don't understand what you're talking about, but it doesn't matter." (from the product owner or project manager)
- The issue is not presented by the product owner, but by a developer or architect.
- Adapting cross-cutting concerns (e. g., testing procedures, logging mechanisms, monitoring)
- Using new or changing work tools (e. g., for monitoring, documentation, issue-tracking, error-tracking)
- Fixing an error or simplifying a task for which a workaround already exists
- Moving a functionality/method to another area (class/package/...)

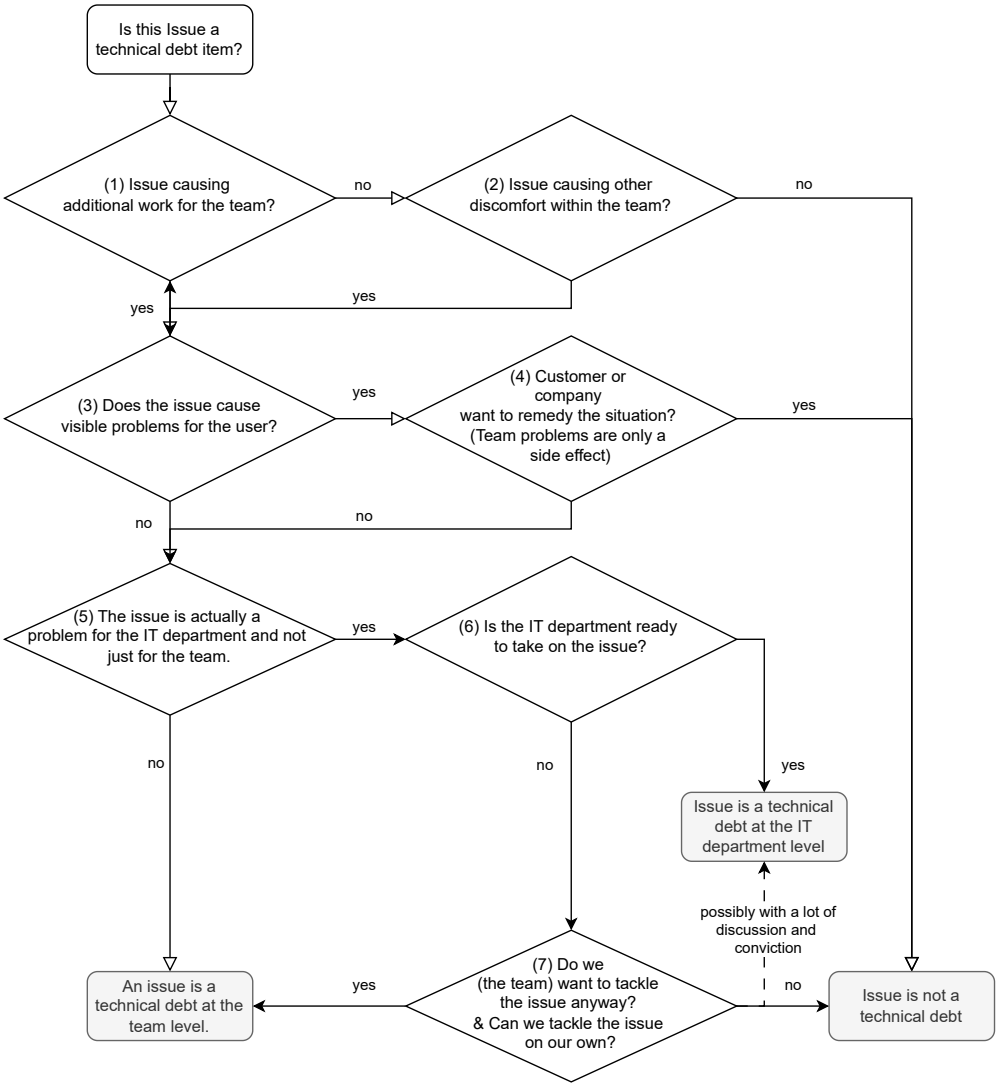


FIGURE 6.7: Flowchart for Identifying TD Items [WSB26], Chapter A

- Improvements to resilience (e. g., adjusting timeout)
- Product owner does not understand the issue.
- The solution and time frame have been communicated to the user before consulting with the developers

We developed various tools (decision matrix, decision flowchart, and list of indicators) to support practitioners in determining whether an issue is a TD item. However, the final decision remains an issue-specific matter and must be made with practitioners' knowledge. The complex version of the decision matrix Figure 6.2, which incorporates multiple stakeholder levels, can be used by practitioners to understand the issue at hand and determine a tailored solution.

6.2.6.2 Reminders for Technical Debt Prevention

In 2015, Li et al. [LAL15] found that TD prevention was understudied. Some papers on prevention strategies for code debt, including respective tooling, have been published [YHMS16]; [Apa+20]; [Tso+20]. Still, there are only a few papers on TD prevention on a more general

level [Bor+21b]; [DMR22]. Publication No. 9 *(Debiasing)* demonstrated the benefits of comparing alternatives, identifying drawbacks, and risks of a solution on architectural decision-making. Consequently, all teams adopted this approach and either added attributes or incorporated these factors into their definition of ready (DoR). Publication No. 8 *(AK in Web Search Engines)* cautions that these important attributes are underrepresented when searching the web, which we provided to the practitioners as a warning.

Practitioners can adopt the practice of adding reminders to discuss the alternatives, drawbacks, and risks of a potential solution in their backlog issue template to promote better decision-making.

6.2.6.3 Technical Debt Attributes

We identified the attributes documented by the teams in their TD issue types. We supplied a summary of them and their content in the TDM Guide. We presented attributes that were adopted by all teams as *best practices* and the ones that one or two teams documented as *nice-to-have* to enable practitioners to customize their approaches. Table 6.3 briefly presents the *best practice* attributes, including their types and contents, which are important for practitioners to implement in their issue-tracking tools.

Practitioners can use the list of attributes to customize their own issue-tracking tool to incorporate a basic TDM process. The TDM Guide in Chapter A comprises further details on *nice-to-have*-attributes.

6.2.6.4 Priority / Key Performance Indicators

As a goal of this study, practitioners noted that they would like to have *Key Performance Indicator* (KPI) for TD. The Cambridge Dictionary defines a KPI as *a way of measuring a company's progress towards the goals it is trying to achieve* [Kpi]. In terms of TD, the goal is to reduce the disturbances through TD to a reasonable level while focusing on the most relevant TD issues for repayment. To achieve this purpose, the KPI must capture the disturbance of each TD issue in a value to enable comparison, i. e., it must support prioritization. Furthermore, it would be beneficial if the value were aggregatable to provide an overview of the overall TD in the system. All three teams used different approaches for this purpose:

Educated guess. The first team decided to utilize an *educated guess*, meaning an estimate informed by an understanding of the ticket's attributes. This approach should be superior to the simple *gut feeling* they previously relied on. The results show the relevance of each TD item and can be used to determine the number of TD items per priority. One goal of a company could be to eliminate all TD items with the highest priority.

Return on Investment (ROI). The second team used an ROI as the primary KPI. To calculate the ROI, we specified the following values for the combo box fields from Table 6.3:

- Interest in minutes: 15; 60; 240; 480; 960
- Interest probability in "per month": 30; 4.5; 1; 0.0027; 0.0013

For the ROI calculation, this team further estimated the effort required to repay the TD in person-days. We calculated the monthly interest burden as presented in Equation (6.1) and determined the ROI in months as shown in Equation (6.2).

$$\textit{interest burden}\ \ \text{minutes/month} = \textit{interest}\ \ \text{minutes} * \textit{interest probability}\ \ \frac{1}{\text{month}} \tag{6.1}$$

$$\textit{ROI}\ \ \text{months} = \frac{\textit{effort}\ \ \text{minutes}}{\textit{interest burden}\ \ \text{minutes/month}} \tag{6.2}$$

To align the ROI with a priority on a one-to-five scale, its values can be categorized into equidistant value areas depending on the resulting value, e.g, as exemplified in Table 6.4. For management, the interest burden values of the TD items could be aggregated to determine the overall interest burden of the system.

Field/attribute	Explanation	Example	Field/attribute Type
Risks of non-repayment (benefits of repayment)	What can happen if this TD item is not repaid? There can be several consequences for TD at the same time. It is essential to list all consequences. You can use the 5-Whys-method to determine the business consequences [Ser17].	Additional work for developers, risk of errors in the system, performance losses, and security risks.	Free text; content: bullet point list
Repayment effort (SP) For sprint planning, an effort estimate should be made using the same unit as for other issues.	Story points or T-shirt sizes	Depending on content: Integer / Text / Combo box	
Contagiousness	Does the cost of repayment increase over time?	An API with TD initially has only one consumer that needs to be changed, but over time, additional consumers are added, all of which then also need to be changed.	Combo box with "will decrease," "will remain the same," and "will increase" as options
Interest	For the most serious *risk of non-repayment*: How much development effort will be required if the case occurs and interest has to be paid?	1 day of additional effort to understand the code without documentation. 4 hours of effort to find and fix the bug and deploy the hotfix. The workaround requires 15 minutes of a developer's working time.	Combo box: 15 min.; 1 hr.; 4 hrs.; 1 day; >= 2 days
Interest probability or frequency	How likely is it that interest will actually be incurred? In principle, this assessment should be made for all *risks of non-repayment*. A pragmatic approach is to assess the most serious *risk of non-repayment*.	(Similar to the above:) How likely is it that someone will need the missing documentation? (1x/year) How likely is it that the TD will result in a production error that causes additional work? (1x/quarter) How often does a workaround have to be performed (manually) by the developers? (daily)	Combo box: 1x/day; 1x/week; 1x/month; 1x/quarter; >= 1x/year(half year)
Priority	A priority can stem from a *gut feeling*, an *educated guess* based on the previous attributes, a calculation of a *mean value* based on the previous attributes, or a calculation of a *return on investment* (ROI), which includes the interest, interest probability, and principal and thus a cost-benefit-ratio.	A code issue that might cause system failures needs immediate action; the priority is *very high*; e. g., the ROI is reached in less than a month. Code duplication in a rarely changed area might be delayed; the priority is *very low*; e. g., the ROI is reached in more than 24 months.	Combo box with 5-point scale, e.g., very low, low, medium, high, very high (potentially calculated in the background as a mean or based on a ROI)
Resubmission date	Each TD issue is assigned a resubmission date. In each refinement, the list of all TD items is sorted by resubmission date, and all issues with expired resubmission dates are reevaluated to update their attributes in response to contextual changes.	A TD item is not relevant now, but is located in a component where we expect changes in half a year. We assign a resubmission date in 5 months to see if the expected change is taking place, and thus the relevance (and thus priority) of the TD issue changes.	Date field
Component	The (architectural) component (i. e., system part, program part) in which the TD is located or has an impact.	TD items in the checkout process of the online shop, in the tax calculation of the booking system, in the front-end, or during database access. (If changes are to be made in the component, it is now possible to identify the related TD issues to assess the risk they pose to the change.)	Hierarchy (usually already included in the standard of an issue-tracking system)

TABLE 6.3: *Best practice*-attributes for a TD issue (Chapter A, [WSB26])

ROI	Prio ID	Priority
$<$ 1 month	5	very high
$<$ 2 months	4	high
$<$ 1 year	3	medium
$<$ 3 years	2	low
$\geq$ 3 years	1	very low

TABLE 6.4: Example of determining priority based on return on investment (ROI).

Mean Priority. The third team used a mean value, i. e., they scaled all relevant attributes from 1 to 5 and computed the mean to determine the priority. This team uses GitLab [Git] as an issue-tracking tool, which does not support calculated values. Thus, the choice of mean calculation may be influenced by this limitation, as computing the mean is cognitively easy. This KPI provides a straightforward method for calculating a priority score, which can be adopted by companies using simple issue-tracking tools. Similar to the educated guess, the resulting priority can be used to determine the number of TD items per priority.

We identified three approaches that differ in complexity and benefits. The approaches are: an *educated guess*, the *ROI*, or the *mean priority*. Practitioners can directly use these approaches for their TD issues, even if they are not captured in an issue-tracking tool, but, for example, in an Excel sheet [Mic25b].

6.2.6.5 Resubmission Date

All teams recognized that TD issues were lost in the backlog, thereby "cluttering" it, which is a finding we similarly identified from publication No. 3 *(IT Managers' Perspective)*. TD tickets with low priority remained at the end of the backlog until they were forgotten. To address this issue, the first team introduced a *resubmission date* to all ticket types, which supports an iterative prioritization approach [Len+20]. In each refinement meeting, the list of all TD issues is sorted by resubmission date, and all issues with expired resubmission dates are reevaluated.

This provides the following advantages:

- Issues are not permanently overlooked at the end of the backlog (as explained above).
- If the priority relies on an undecided business decision, the team can set the resubmission date to the estimated decision date to reevaluate the priority.
- If a planned change will soon render the issue obsolete (e. g., system migration), the team can set the date to the expected shutdown. A decision on closing the issue can be made on the resubmission date.
- If the issue's priority is low and paying interest is preferable to repaying the TD, a resubmission date can be set. On that date, the need for repayment will be reassessed based on in-between context changes. The issue stays open for risk assessment without requiring repayment.

Practitioners can adopt this idea of the resubmission date. All teams expressed enthusiasm for this approach, and it was also well received in discussions with other practitioners, for example, when presenting or discussing the results at practitioners' conferences.

6.2.6.6 Visualizations

Publication No. 6 *(Establishing TDM)* presents multiple approaches to visualizing the TD issues and their estimated attribute values (see Table 6.3). Through interviews and group discussions during the workshops, we identified that the visualization have two main objectives:

- Support for planning, i. e., prioritization and monitoring of TD
- Raising awareness of TD, especially among non-technical stakeholders such as customers, business management, project managers, etc.

For planning purposes, the following visualizations have become popular among the teams:

- *Low-hanging fruits* (Figure 6.8, upper left corner): In a coordinate system where "effort" is represented on the x-axis and "priority" on the y-axis, each point indicates the existence of TD issues. The size of the points corresponds to the number of issues at that location. When the user clicks a point, a list of the associated TD issues is displayed.
- *Components / Quality attributes* (Figure 6.8, upper right corner): Issues are presented for each component or quality attribute in a bar chart. The list shows the TD issues associated with the component or quality attribute the user selects.

These interactive visualizations have become particularly popular for planning purposes, as they filter an additional list of TD issues (Figure 6.8, bottom). This list displays attributes relevant for decision-making and is sortable by all attributes, making it particularly valuable for selecting the TD item that needs to be addressed next. The resubmission date is also relevant information, but has a different purpose than selecting a TD item. Consequently, it may be an element of another list that can be used to identify TD items requiring reevaluation.

The visualizations used to raise awareness of TD also serve to monitor the TDM process and are primarily based on time series, e.g.:

- Number of opened TD issues per month
- Number of closed TD issues per month
- Number of open TD issues per month
- Interest burden (see Equation (6.1)) of all open TD issues per month.

Unfortunately, as explained in the theoretical contributions, presenting the interest burden proved to be impossible with reasonable effort.

More details on other visualizations are presented in publication No. 6 *(Establishing TDM)* and the additional material of publication No. 7 *(Practical Guide)*.

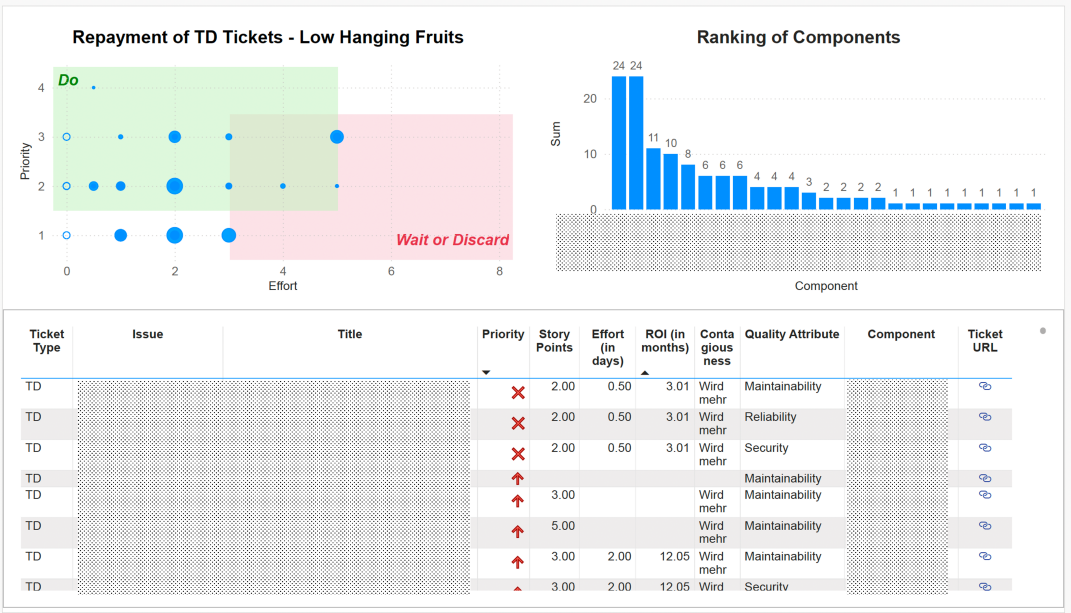


FIGURE 6.8: Example Visualization of TD issues for Planning Purposes [WSB26] Chapter A

We identified visualizations that are useful and favored in practice. Beyond creating filterable and sortable lists, these visualizations are *low-hanging fruit*, bar charts for components and quality attributes, and timelines to observe and control TD development. Practitioners can use these visualizations as templates to develop their own. Moreover, we encourage issue-tracking tool vendors to include a TD issue type and these associated visualizations in their standard.

6.2.6.7 Establishing a Technical Debt Management Process

As part of the TDM Guide (Chapter 16), we introduced a step-by-step procedure for establishing the TDM process, i. e., starting the TDM process. The single steps are listed below in chronological order:

1. The "technical debt manager" (TD manager) is specified in a joint meeting. From then on, the TD manager coordinates the subsequent instructions.
2. A label for TD (e. g., "TD") and issues that have already been searched but not identified as TD (e. g., "Non-TD") is defined.
3. All existing issues are reviewed, it is determined whether or not they constitute TD, and the issues are tagged accordingly ("TD" / "Non-TD"). ("Non-TD" helps analyze smaller blocks and track considered issues.)
4. Team members record all TD they are aware of but have not yet documented, and tag it as TD.
5. The Team discusses which attributes from Table 6.3 are useful for them/their company.
6. The TD manager creates a new issue type for TD with the selected attributes.
7. The TD manager transfers all issues tagged as TD to the new issue type.
8. All TD issues are evaluated. There are two options for this:
   - The TD manager schedules specific meetings for all team members to refine TD issues. Issues are discussed and evaluated one by one, meaning the attributes are filled in. This approach is recommended early in evaluations to help the team develop a shared understanding of the attributes.
   - Each team member handles 3-4 TD issues weekly and evaluates them individually in advance, setting a resubmission date that indicates urgency. If uncertain, they can opt for a short-term date, while for less relevant issues, a longer date can be chosen. During refinement, expired issues are reviewed together by the team, distributing evaluation over time and avoiding separate meetings.
9. The Team discusses which repayment procedure should be used (in which situations).
10. The TD manager ensures that the repayment agreements are adhered to.
11. The TD manager clarifies which visualization tools can be used, links the issue tracker to the visualization tool, and creates initial visualizations.
12. The visualizations are used by the team and adjusted by the TD manager as needed.
13. The TD manager adds attributes for TD prevention, i. e., quality attributes, benefits, drawbacks, and solution alternatives in all issue types, e. g., as attribute fields or part of a DoR.

Participants estimated the time required for this establishment and reported it in a survey. Publication No. 6 *(Establishing TDM)* noted that the participants invested less than 3% of their time in establishing and maintaining the process, respectively, while the TD manager invested less than 5% of their time. However, the participants' feedback suggests that initial support from an external entity (e. g., researchers or consultants) is necessary to provide guidance and feedback, as reported in publication No. 7 *(Practical Guide)*.

The step-by-step procedure for starting TDM helps practitioners overcome the status quo bias. Practitioners can carry out these steps at their discretion, making this part particularly important for the TDM Guide. The time participants recorded provides IT managers with important data to assess whether and when to establish a TDM process.

Chapter 7

Discussion of Contributions to Research Questions and Gaps

This chapter summarizes and discusses the contributions with respect to the research questions and the research gaps presented in Section 1.2 and Chapter 3. Figure 7.1 illustrates the contributions of the publications to each respective RQ.

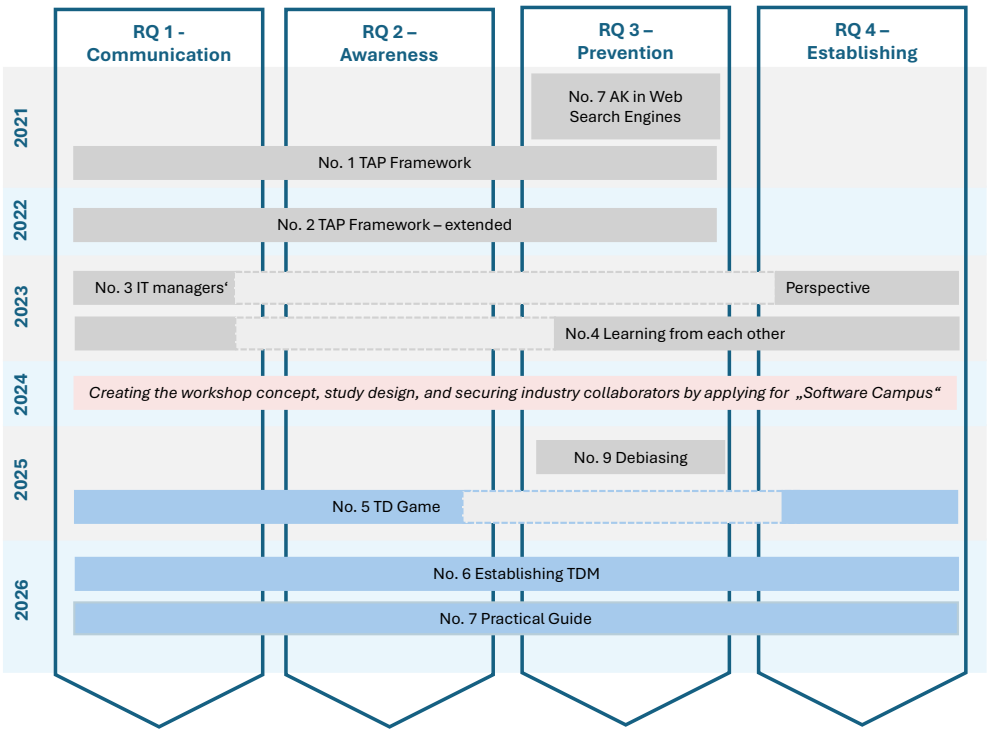


FIGURE 7.1: Publications per Year with their Contribution to the Research Questions
Grey – Step I, Red – Step II, Blue – Step III of the Thesis

Table 7.1 provides an overview of all RQs and aligns them with the previously introduced contributions (by section title), the respective publication (by number), and the research gaps they address (by number from Chapter 3).

The following sections discuss how the publications address the RQs and how the contributions address the respective research gaps.

7.1 RQ1 – How can we optimize the communication about TD?

The research in publication No. 1 *(TAP Framework)* shows that the TDM process led to more rational discussions and decisions within the observed team than within the comparison

RQ	Type	Publ.	Contribution (Title of the Section)	Research Gaps
RQ1 (*Communication*)	R	1, 2	Technical Debt Metaphor and Communication Issues	C1, C3, C5
	R	3	V4CTD Model	C3, C4
	R	4	Model for Managing Architecture Mistakes	C2, C8
	R	1, 2	Effects of a Technical Debt Management Process	C2, C4
	R	7	Improving the Collaboration with Tool Vendors	C2, C4
	P	3	3-Minute Pitch based on the V4CTD Model	C4
	P	4	(Communication about Software Architecture Mistakes) Communication-Facilitating and -Hindering Factors	C5
	P	4	(Communication about Software Architecture Mistakes) Ensure Communication Opportunities	C2, C4
	P	5	TechDebt Game	C2, C4
	P	6,7	(TDM Guide) Reminders for Technical Debt Prevention	C2, C5,
	P	6,7	(TDM Guide) Technical Debt Attributes	C2, C4, C6
	P	6,7	(TDM Guide) Priority / Key Performance Indicators	C2, C4
	P	6,7	(TDM Guide) Visualizations	C2, C4, C6
	P	6,7	(TDM Guide) Establishing a Technical Debt Management Process	C2, C4
RQ2 (*Awareness*)	R	1, 2	Effects of a Technical Debt Management Process	A3
	R	6,7	Measuring Technical Debt Awareness with TD-SAGAT	A1, A2
	R	6,7	Awareness Effects of the Technical Debt Management Establishment	A3
	P	5	TechDebt Game	A3
	P	6,7	(TDM Guide) Reminders for Technical Debt Prevention	A3
	P	6,7	(TDM Guide) Establishing a Technical Debt Management Process	A3
RQ3 (*Prevention*)	R	1,2	Effects of a Technical Debt Management Process	P1, P5
	R	6,7	Awareness Effects of the Technical Debt Management Establishment	P1, P2, P5
	R	7	(Uncovered new Research Areas) Debiasing Decisions on a Design Level	P1, P2, P5
	P	1, 2	TAP Framework – a new TD Repayment Strategy	P1, P2, P3, P4
	P	4	(Communicating about Software Architecture Mistakes) Ensure Communication Opportunities	P1
	P	5	TechDebt Game	P1, P2, P5
	P	6,7	(TDM Guide) Reminders for Technical Debt Prevention	P1, P2, P5
RQ4 (*Establishment*)	R	1,2	Effects of a Technical Debt Management Process	M1
	R	6,7	Awareness Effects of the Technical Debt Management Establishment	M8
	R	3	(Uncovered new Research Areas) Status-Quo Bias in Technical Debt Management	M1
	R	7	(Uncovered new Research Areas) Technical Debt Identification across Stakeholder Levels	M1
	R	7	(Uncovered new Research Areas) Debiasing Decisions on a Design Level	M1
	R	7	(Uncovered new Research Areas) Dealing with Multiple Interests	M1
	R	7	(Uncovered new Research Areas) Scaling the Workshop Concept	M1, M8
	R	6, 7	Improving the Collaboration with Tool Vendors	M1, M2, M5, M8, M9
	P	1,2	TAP Framework – a new TD Repayment Strategy	M7
	P	4	(Communication about Software Architecture Mistakes) Communication-Facilitating and -Hindering Factors	M1, M6
	P	4	(Communication about Software Architecture Mistakes) Ensure Communication Opportunities	M1, M6
	P	5	TechDebt Game	M6
	P	6	Workshop Concept	M2, M3, M4, M6, M8, M10
	P	7	Technical Debt Management Guide	M5, M8, M9
	P	7	(TDM Guide) Technical Debt Identification	M8
	P	6,7	(TDM Guide) Reminders for Technical Debt Prevention	M5, M8
	P	6,7	(TDM Guide) Technical Debt Attributes	M5, M8
	P	6,7	(TDM Guide) Priority / Key Performance Indicators	M1, M8,
	P	6,7	(TDM Guide) Resubmission Date	M1, M8
	P	6,7	(TDM Guide) Visualizations	M1, M5, M8, M9
	P	6,7	(TDM Guide) Establishing a Technical Debt Management Process	M1, M2, M3, M8
R. Goal			Proposals for Improving Research Applicability in Practice	

TABLE 7.1: Contributions per RQ with relations to publications and research gaps
(Type: R – Research, P – Practical; Research Gaps: reference Chapter 3)

team. We showed that the estimation of implementation and principal costs might have caused this improvement. In the extended article of this paper in publication No. 2 *(TAP Framework – extended)*, we demonstrated that the TDM process similarly improved communication between IT managers and business stakeholders. The subsequent question arose whether other forms of TDM establishment would improve communication in a similar way. This question and the study's results motivated the incorporation of cost factors into subsequent action research studies on the feasibility of the workshop concept.

First of all, in publication No. 3 *(IT Managers' Perspective)*, we identified levels of visibility that provide a model to better understand the concepts underlying the communication issues. The V4CTD model provides a comprehensive framework for explaining the causes and consequences of TD and their visibility. This model might be used to bridge visibility gaps when adopted in real-world scenarios to create a *3-minute pitch*, as described in the article. The identified communication issues with developer personalities reveal a new challenge for TD communication that warrants further research.

In publication No. 4 *(Learning from each other)*, we identified communication challenges when discussing a critical aspect of software engineering, like mistakes in software architecture, i. e., ATD, and proposed improvements. In particular, the aspect of ensuring communication opportunities can be applied when establishing a TDM process, e. g., by creating a CoP for TD. Additionally, this work identifies facilitating and hindering factors of communication that can serve as guidelines for leading discussions on such critical aspects as TD.

To provide a more concrete solution to the communication issues, we developed the TD Game and showed its effects in the publication No. 5 *(TD Game)*. The game provides practitioners with a means to enhance understanding of TD among various stakeholders and, in doing so, improve communication between them. Furthermore, the game is designed to foster discussion and negotiation about TD during gameplay. This replaces heated real-life discussions and enables more objective discussion during the less emotionally charged emulation.

During the action research of publications No. 6 *(Establishing TDM)* and No. 7 *(Practical Guide)*, we established customized TDM processes in three further IT teams. This research supported the previous findings from publication No. 1 *(TAP Framework)* that communication improves by estimating and evaluating the attributes of TD issues. Developers can now understand and identify TD items that are not worth repaying. Furthermore, similarly to publication No. 2 *(TAP Framework – extended)*, the clear ranking of TD items supported the management in prioritizing and planning TD repayment.

C1. Metaphor: ***Is the metaphor used for communication in practice? Can the metaphor or its usage be improved?*** We identified that IT managers rarely use the TD metaphor to explain the TD subject to business stakeholders. Instead, they often use their own metaphors. However, the term TD is widely known and accepted in the IT community. Thus, there is no need to change the terminology. However, other means of communicating TD-related issues must be identified, as the metaphor does not seem suitable for this purpose.

C2. Information: ***Which information related to TD is suitable and relevant to communicate about?*** First, we found that consciously discussing and comparing solution options and their related costs leads to more rational discussions and decisions. Second, the TD Game provided a list of learning objectives, i. e., factors relevant to TDM, and fostered the discussion about those factors. Practitioners identified the game as a realistic emulation, proving the relevance of these factors. Third, to uphold TD awareness, the teams included reminders in their backlog issues to discuss alternative solutions, their drawbacks, and the risks of each issue type. With this, they used debiasing techniques that have been shown to successfully improve decision-making. The participants were warned that when searching the web for architectural information, they would encounter fewer drawbacks than benefits. Fourth, for each TD item, the teams discussed the principal, interest, interest probability, and contagiousness. From this information, they derived priorities in various ways, e. g., by making an educated guess in a group effort or by calculating it. Fifth, the resubmission date was used to regularly reassess and thus discuss the TD item. Sixth, visualizations of the recorded

TD items supported the communication with business stakeholders. Finally, the estimated time effort of 3% is essential information for negotiations with IT managers and business stakeholders.

C3. Communication Partners: ***Which (other) communication partners and directions may be relevant?*** We provided information on which TD concepts are visible to which stakeholders and showed that business stakeholders are often only aware of some causes and consequences of TD, missing the whole picture. Developers are often too limited in their technical view to gather the beginning of a chain of causes or the end of a chain of consequences. IT managers need to bridge this communication gap. Additionally, we identified that IT managers face a duality when discussing TD with IT developers: some developers are extremely quality-oriented and do not distinguish between relevant and irrelevant TD, while others do not consider quality at all. Thus, communication partners should not only be divided by stakeholder type, but also by personality.

C4. Transferability: ***Why are the identified methods not transferable? For example, are they too context-specific, or too ambiguous to be applicable? How can we change this?*** First, we provided the V4CTD model and the 3-minute pitch slide derived from it to help IT stakeholders uncover and explain their own TD items. Second, the TD Game provides concrete support for initiating TD communication by making the situations feasible for all stakeholders in an emulated environment. This provides new insights for business stakeholders and junior developers, improves understanding between junior and senior developers, and increases the trust business stakeholders place in their developers. Third, answering C1, we provided insights into which TD-related attributes are relevant discussion topics. With this, we provide concrete information on what to communicate about. Fourth, these results are presented in a TDM Guide for practitioners, including a step-by-step approach to establishing and communicating about the TD topic. Fifth, using the gathered data, we showed which visualizations of these data are relevant. These visualizations guide the discussions, e. g., on which TD item to repay next, but also in terms of management communication, e. g., to compare the amount of TD between teams. Unfortunately, not all required visualizations could be implemented, thereby hindering informed discussions. This calls for more interaction and collaboration with vendors of issue-tracking systems.

C5. Psychology: ***What are the potential psychological effects associated with a TDM process? How can we address those?*** First, we showed that the TAP framework and its TD repayment, as part of a project, create a feedback loop in team and project managers' behavior that prevents TD items, which is a form of psychological trickery. Second, concerns about the duality of developers' communication indicate that it is important not to focus solely on communication with different stakeholder types, but to consider personality differences in future research. Third, the conceptual model for handling software architecture mistakes centers on communication and its psychological effects, resulting in a list of factors that facilitate and hinder communication. Fourth, we showed that cognitive biases can be reduced by using debiasing techniques, which we included in the TDM Guide as reminders in the backlog issues to discuss alternatives, drawbacks, and risks. Finally, the TD Game provides a psychologically safe space to encounter TD-related issues and discuss them with various stakeholders.

C6. TDM Process and Communication: ***What are the interchanges between a TDM process and communication? For which activities of a TDM process is communication relevant, e. g., needed or possible? Which elements of a TDM process trigger communication?*** We identified several standard situations for communicating about TD in our TDM process. First, reminders across all issues initiate discussion of the optimal solution, supporting TD prevention. Second, when identifying TD items in the backlog, the question must be discussed whether a given issue is a TD item. Third, the evaluation of a TD item's attributes usually triggers an intensive discussion during TD documentation. Fourth, the discussion of a TD item's priority, whether when making an educated guess or deciding which TD item to repay next, triggers discussion. Fifth, the resubmission date ensures that TD items are

rediscussed and reassessed regularly in terms of TD prioritization. Finally, examining visualizations of the backlog together might facilitate communication, e. g., with IT or business management.

7.2 RQ2 – How can practitioners increase long-term awareness of TD?

We developed the idea of the TD Game described in publication No. 5 *(TD Game)* to raise initial awareness of the TD topic by introducing TD concepts and providing insight into the emotions associated with it.

To maintain long-term awareness, publication No. 1 *(TAP Framework)* provided initial evidence that the presented TDM process increased the observed team's awareness of comparing solutions and TD cost factors relative to the comparison team. Furthermore, in publication No. 2 *(TAP Framework – extended)*, the IT managers specifically mentioned that awareness of the TD topic was raised among them as a result of the established TDM process. This prompted further research to evaluate whether any TDM process can yield these awareness benefits.

As the literature review did not disclose any methods for measuring awareness in software engineering, we searched for such methods in other domains and found the SAGAT method in psychology. Together with research from this domain, we adopted this method to TD awareness and employed and explained it in publications No. 6 *(Establishing TDM)* and No. 7 *(Practical Guide)*. Consequently, in these publications, we supported the finding that actively managing TD, including adding reminders to backlog issues (e. g., attributes as text-, combo-, or checkboxes), not only improves awareness of TD but also sustains it during developers' daily work. However, based on participant feedback, we also found that, initially, continuous support from external sources is valuable and should be considered when establishing TDM.

A1. Definition: ***How could and should TD awareness be defined? Which definition is helpful in research?*** While defining TD awareness as knowing the concept of TD or the TD items in one's own system is valid, it is often insufficient for use. Stakeholders who know about the TD items in their system might still inadvertently incur TD because they do not think about TD when making decisions. Therefore, we introduced a definition of TD awareness in decision-making situations that can be used to identify means of avoiding inadvertently incurred TD: *TD awareness is the (1) perception of the prerequisites (e. g., alternatives, drawbacks, interest) of a decision (e. g., incurring TD) during decision-making situations (e. g., refinement meetings), (2) the comprehension of these prerequisites' meaning, and (3) the projection of their status in the future (i.e.., TD's consequences).*

A2. Measuring Effects: ***How can TD awareness be measured, given a previously clarified definition? What are valid metrics, and how can those be gathered?*** Based on the previously introduced definition, we identified nine prerequisites that need to be considered in decision-making situations, i. e., being aware not to inadvertently incur TD and only to repay relevant TD issues: (1) solution alternatives, (2) solution's benefits, (3) solution's drawbacks, (4) solution's risks, (5) effort for implementing a solution, (6) TD item's principal, (7) TD item's interest, (8) component affected by TD, and (9) quality attributes affected by TD. These nine prerequisites can now be used as metrics in questionnaires, observations, or other data-collection methods. We used the prerequisites as metrics in the workshop survey and in our meeting observations. Moreover, we introduced and applied the SAGAT approach of interrupting participants in their work to obtain direct, less biased input.

A3. TDM Process and Awareness: ***How much and through which measures does a TDM process influence the TD awareness? Can we support these claims with metrics?*** First, in our first case study, we found that the TAP framework increases awareness, for example, by comparing solution options and discussing their cost factors. Moreover, the overview generated from the listed TD items improved their IT managers' awareness. Second, the TD

Game is a means of enhancing immediate awareness, as all players reported that the game changed their attitudes and that they planned to change their behavior. Third, the resubmission date helps developers remember all TD items and serves as a constant reminder to keep TD levels in check. Finally, we applied the previously introduced TD-SAGAT method in our action research studies and discovered that only adding reminders about prerequisites in the team's backlog issues improved TD awareness over the long term. However, initially, reminders from an external person are necessary to ensure consistent use of the reminders.

7.3 RQ3 – How can practitioners prevent TD incurrence?

During the study on publication No. 1 *(TAP Framework)*, we presented a repayment approach that reduced TD incurrence through the feedback loop in team behavior. Additionally, we found that the significance of the correlation between comparing solutions and TD prevention is only slightly above the 5% level. This indicates that a correlation may exist, particularly given our small sample size. Nevertheless, the study showed that TD prevention in the observed team was much better than in the comparison team, and we concluded that comparing solution options is essential to prevent TD. Similarly, in our publication No. 8 *(AK in Web Search Engines)*, we identified that the drawbacks of architectural solutions, together with the use cases that describe them, are highly relevant to decision-making and, by extension, to the prevention of TD items. Moreover, the workshop on debiasing decisions in publication No. 9 *(Debiasing)* introduces comparing solution alternatives, their drawbacks, and risks as methods for debiasing. The study shows that providing these debiasing techniques significantly alters participants' attitudes in decision-making situations and reduces biased decisions.

Unfortunately, in publication No. 8 *(AK in Web Search Engines)*, we found that the drawbacks of (and use cases for) architectural solutions are often not presented equally when searching for architectural knowledge on web search engines. This demonstrates that it is essential to consciously search for the drawbacks of solution alternatives. We included this information in the workshop concept, and as presented in publications No. 6 *(Establishing TDM)* and No. 7 *(Practical Guide)*, all teams decided to add reminders to their backlog issues to consciously search for and discuss the drawbacks and risks of an intended solution.

Finally, publication No. 4 *(Learning from each other)* demonstrates that open communication in a psychologically safe environment can help prevent mistakes and, by extension, the occurrence of ATD. When architects openly discuss their mishaps, other architects may learn from them and avoid repeating the same mistake, thereby preventing related TD items in the future. From this publication, we derived the idea of creating a CoP on TDM, in which participants can openly discuss the causes, consequences, and metrics of TD.

P1. Strategies: ***How can we prevent inadvertently incurred TD items? Are there other strategies for TD prevention beyond those already adopted by practitioners?*** First, the TAP framework provides a repayment approach that creates a feedback loop in team behavior, making the team less prone to incur TD. In particular, the comparison of alternatives appears to promote the avoidance of inadequate solutions. Second, regarding software architecture mistakes, we identified that ensuring opportunities for communication might foster architects' learning from one another and help them avoid mistakes that other architects have already made. Third, the TD Game made junior developers aware of the dangers of incurring TD, leading them to claim that they plan to be more careful in the future. Fourth, the backlog reminders prompted participants to debias their decisions by considering various solution alternatives, particularly their drawbacks and risks. This leads to better decisions and, thus, avoids inadvertently incurred TD. However, we also identified that developers sometimes struggle to identify alternatives. Thus, we encourage further research on debiasing design decisions on the lower level than architectural decisions. Additionally, we uncovered that the drawbacks and risks of potential solutions are underrepresented on the World Wide Web. We cautioned the participants about this, but it also emphasizes the aforementioned research gap in successfully debiasing design decisions.

P2. Operationalization: ***What are concrete measures that can be utilized by practitioners?*** We introduced various methods for preventing TD, as outlined above. Most of them provide concrete measures. First, using the repayment method from the TAP framework can prevent TD, given a similar situation, i. e., when a deadline is set, and work is under time pressure. Second, the TD Game might educate junior developers to be more careful and change attitudes toward TD among all stakeholders. Finally, applying debiasing techniques and adding relevant attributes to backlog issues to ensure continuous reminders might help prevent unintentionally incurred TD.

P3. Time Pressure: ***How can TD be prevented under time pressure from (project) deadlines when there is not enough time to implement the debt-free solution?*** We introduced the repayment method from the TAP framework, where TD incurred under time pressure has to be recorded and repaid after reaching the deadline, but before finalizing the project. This can prevent TD through the presented feedback loop in team behavior.

P4. Measuring Effects: ***How can we assess the success of prevention strategies? How can we measure how many TD items have been prevented?*** We were unable to adequately address this research gap and encourage further research. However, we systematically observed meetings in which potentially TD-incurring decisions are made. We marked all situations that posed a risk of unintentionally incurring TD in an observation protocol and discussed them with the participant to increase their awareness of these risks. We observed that the number of such situations decreased over the course of each study. While this observation is notable, the number of data points is low, and the researcher's biases in recording limit the validity of the results and, as such, provide only anecdotal evidence.

P5. TD communication: ***Which kind of communication can be effective in preventing TD? Which stakeholders have to communicate with each other in which situations?*** Regarding communication, it is essential to discuss the attributes introduced above to debias decision-making during refinement meetings, when the product owner and developers discuss upcoming issues. Additionally, the TD Game fosters stakeholder discussions in a safe environment that may help prevent TD by understanding its consequences.

7.4 RQ4 – How can practitioners establish a TDM process?

One main concern of IT managers that we identified in the publication No. 3 *(IT Managers' Perspective)* is how to overcome the status-quo bias and establish a TDM process. This concern motivated this RQ, the development of the workshop concept, and the subsequent action research studies presented in publications No. 6 *(Establishing TDM)* and No. 7 *(Practical Guide)*.

The workshop concept incorporates results from other researchers and research from this thesis. Results from the following thesis publications were included in the workshop concept, which have been detailed Section 6.2.5: No. 1 *(TAP Framework)*, No. 4 *(Learning from each other)*, No. 5 *(TD Game)*, No. 8 *(AK in Web Search Engines)*, and No. 9 *(Debiasing)* . Additionally, we incorporated newly developed approaches from publication No. 6 *(Establishing TDM)* (e. g., ROI, resubmission date, and visualizations) into the workshops of the subsequent studies.

The results of the action research and its replications answer this RQ by providing a list of best practices, a step-by-step guide for establishment, and a list of common challenges with corresponding solutions. Practitioners can choose to employ the workshop concept described in detail in publication No. 6 *(Establishing TDM)* to create a tailored process, or to directly employ the TDM Guide published as additional material along with publication No. 7 *(Practical Guide)*.

M1. Problem Space: ***Did we already comprehensively study the problem space? Did we overlook practical issues in establishing a TDM process?*** During the studies contributing

to this thesis, we identified various challenges in managing TD that are relevant to practitioners but have not been, or have not been sufficiently, researched. We addressed some of these problems in this thesis and left others for future research.

- We found that project portfolio planning affects TD by specifying project deadlines, which is one cause of TD. IT managers were better able to plan their portfolios after adopting the TAP framework and gaining an overview of a project's intentionally incurred TD items.
- We identified the status quo bias, i. e., the tendency for IT managers to recognize the need to manage TD but to refrain from doing so due to an unconscious preference to maintain the existing approach. We specifically addressed this issue by developing the workshop concept and including the step-by-step section of the TDM Guide.
- We identified factors that facilitate or hinder communication about critical issues, i. e., mistakes in software architecture, which may be known in psychology but have not been considered in communication related to TDM to date.
- We recognized the need to actively create opportunities for communication to discuss such critical issues, e. g., by providing dedicated communication spaces in workshops and by implementing a CoP.
- Establishing a resubmission date and the corresponding process addresses the challenge of changing priorities for TD items. The issue has been described previously, but research did not provide a means of ensuring iterative reassessment.
- While visualizations have been researched, the research focused on proprietary tooling. To the best of the authors' knowledge, no research on general visualizations of TD backlog issues has been conducted.
- We provided estimates of the effort required to establish and maintain a TDM process, which is valuable information when negotiating such an initiative with business and IT managers, but has not been investigated before.
- We pinpointed the need for supporting the establishment by an external person, which has not been discussed in previous research.
- We uncovered that TDM must be assessed at multiple levels: research on how to scale a team approach within a company is necessary, as is the identification of TD at the team, IT organization, or company level. The thesis provided initial ideas and insights, but we encourage further research on this complex topic.
- We determined the relevance of using companies' issue-tracking tools to integrate a TDM process with existing processes and the challenges associated with their use. These challenges may be why teams abandon TDM approaches prematurely. We encourage closer collaboration with issue-tracking tool vendors to overcome those challenges.
- We identified that practitioners need much more concrete support than currently provided in research, e. g., by providing not only the concepts for interest and interest probability but also their manifestations, i. e., potential values. Future research should provide concrete guidelines for adopting the approach in practice if it claims practical contributions.
- We determined multiple new research directions (Section 6.1.7): helping practitioners identify alternative design approaches, addressing multiple interests, and determining which KPIs to use in which situations.

M2. Preconditions: ***What are the preconditions for establishing a TDM process, e. g., in terms of authority, support, time, tools, and commitment?*** Similar to previous research, we found that establishing TDM requires management support and authorization to design the process and dedicate time to it. We determined that establishing and maintaining the TDM process each consumes approximately 3% of each developer's time. This time must be invested in addition to the time the developers need to repay TD items. Second, we found that practitioners require more concrete support than research provides, as explained above. Another precondition is an appropriate introduction not only to the TDM process but also to its establishment. Third, we found that the use of an appropriate issue-tracking tool is critical to an optimal TDM process. While we were able to establish processes for all three tools

(Azure DevOps, Jira, GitLab), GitLab proved less adaptable, and none of the tools provided sufficient visualization support. For the GitLab solution, an Excel export was identified as the best workaround. However, this contradicts the notion of optimal integration. Finally, the iterative approach proved suitable for focusing on different TDM elements separately without cognitively overloading the participants.

M3. Required Time: ***How can we establish a TDM process without overburdening the practitioners with the time needed for establishing and upholding the process?*** As mentioned above, an iterative approach appears most suitable for avoiding cognitive or time overload. Participants can test the new process and adapt it, e. g., remove unnecessary attributes from the issue type, if the time investment does not justify the outcome. With this approach, a TDM process can be tailored to a company's or team's willingness regarding invested time. We further showed that 3% of a developer's time must be freed for this purpose, thereby making time management for IT managers tangible.

M4. Process Integration – Establishment: ***How can a TDM establishment be integrated into the predominant agile, iterative, and self-determined mindset in the IT industry?*** A predefined process forced on a team by their superiors is counterproductive to an agile mindset. We found that the iterative workshop approach not only avoids cognitive overload but also aligns with the agile mindset, in which processes are adapted to the team's preferences and evaluated after a period of use, a principle known as "inspect & adapt" in agile methods.

M5. Process Integration – Resulting Process: ***How can we ensure that the TDM process is optimally integrated into existing mindsets, workflows, and tooling?*** Within the workshop concept, teams could adopt the approaches most suitable for their contexts, resulting in customized processes. The teams unanimously chose to use their existing issue-tracking tools for recording TD items and to create a TD type for this purpose. They added reminders for debiasing (alternatives, drawbacks, risks) to all issue types. For TD items, they added the four main TD attributes (principal, interest, interest probability, and contagiousness) and a resubmission date. All process steps could be seamlessly integrated into the existing process, i. e., without requiring additional meetings. We introduced these attributes as *best practices* and approaches adopted by some teams as *nice-to-haves*, thereby still allowing for a customized process when using the TDM Guide. Moreover, the teams identified similar ideas for visualizing the data. Unfortunately, all issue-tracking tools fell short of supporting the desired visualizations, making a seamless integration difficult for this specific topic.

M6. Culture Change and Psychological Effects: ***How can such a change in culture be initiated? Which psychological effects are relevant and must be taken into consideration when establishing a TDM process?*** First, the conceptual model for handling software architecture mistakes offered insight into communicating about critical topics, such as architecture mistakes or TD items. The study identified facilitating and hindering factors for such communication, which should be taken into account when initiating discussions on TDM. Second, this study identified that creating opportunities for communication is critical to foster communication on such topics, e. g., by providing time and space for such discussions as part of the workshop concept or by initiating respective CoPs. Third, the TD Game might open discussion of these critical topics in an emulated, and therefore uncritical, setting. Finally, the iterative approach of the workshop concept aligns with research on change management in business, which has demonstrated success in changing mindsets incrementally.

M7. Budget Split: ***How can we overcome the split of the budget in the project budget and the maintenance budget that boosts the accumulation of debt?*** We showed that the feedback loop inherent in the TAP framework's repayment method addresses this problem. Team behavior and, specifically, that of the project managers, changes if they are aware of having to repay the TD items they incurred during their project.

M8. Holistic Process: ***How can a holistic TDM process that includes all TD activities and is not limited to specific TD types look like?*** We identified such a holistic process by establishing a TDM process across three teams from different companies, allowing them to decide which approaches to adopt, and observing their challenges. Comparing those processes, we identified similarities that we defined as a basic TDM process (*best practices*), which can be augmented by other approaches selected by only one or two teams (*nice-to-have*). The elements of the basic process have been described above (M5). This information was compiled into a TDM Guide to be used by subsequent teams to adopt the basic process and augment it with *nice-to-have* elements. In the TDM Guide, we provided guidance on calculating a priority, e. g., as a mean or an ROI. However, no *best practice* emerged. We further specified desirable visualizations based on this data. Finally, we pinpointed some challenges that still need to be overcome and thus pose further research opportunities (see M1).

M9. Transferability: ***How can we ensure this transferability between companies?*** The interactive workshop approach fosters transferability, as it does not present a complete TDM process but rather a framework for designing a context-sensitive TDM process. Providing participants with the choice to select their own process elements makes this approach sufficiently flexible to be applied across various contexts. This thesis demonstrated this transferability by applying the workshop concept in three companies with different domains, organizational structures, team tasks, role structures, and tools. The cost elements adopted by all teams suggest that we may derive a generally valid basic TDM process (*best practices*) transferable to many contexts. The variation in prioritization methods indicates that the workshop concept remains sufficiently flexible to accommodate diverse organizational styles.

M10. Measuring Effects: ***How can the effects of establishing a TDM process be measured, particularly with regard to other TD types?*** We used the approaches the teams adopted as one first measurement, as the adoption rate provides insights into the usability and feasibility of the presented approaches. To assess effects on awareness, we gathered the TD-SAGAT prerequisites from a workshop survey, a TD-SAGAT survey, and meeting observations. Comparing the results yielded further insights, including the effect of adding reminders to issue types. The presented visualizations show changes in TD items, including counts, priorities, interest amounts, and related metrics, which can be used to assess the effect of a process. However, this should be interpreted cautiously, as the number of TD items may be affected by incomplete TD item recording. To augment these methods, we used meeting observations. While those may still be observer-biased, they add another perspective to the data in terms of triangulation and provide information about the adoption and use of process steps. Finally, we used questionnaires to gather the participants' satisfaction with the process, which may also be highly biased, but it adds another method to triangulate. Overall, measuring the effect of a TDM process remains a topic for further research. As of today, there is no metric that objectively and reliably measures a system's TD. Accordingly, there is no such measure to assess the success of a TDM process.

Chapter 8

Threats to Validity

This chapter presents threats to validity based on the guidelines provided by Wohlin et al. [Woh+12], i. e., structured by:

- Construct validity, i. e., the suitability of the selected methods
- Internal Validity, i. e., the risks of drawing incorrect conclusions regarding the causal relationships between treatment and outcomes
- External validity, i. e., the ability to generalize the result
- Conclusion validity, i. e., the trustworthiness of the conclusions drawn from the results.

This section on the threats to the validity focuses on the action research outcomes, as these are the final and foremost results of this thesis and cover PART II and III. For the initial studies of PART I of this thesis, each publication discusses the respective threats to validity, which we have taken into account when designing the workshop concept in PART II.

In his book on action research in software engineering, Staron [Sta20] presents several common threats to validity in action research studies, using Wohlin et al.'s guidelines as a reference [Woh+12]. This section presents typical threats to validity in action research, based on the description and terminology provided in the book [Sta20].

Furthermore, we reflect on the *ACM Special Interest Group on Software Engineering*'s empirical standards for action research studies. These standards indicate that *"example criteria include reflexivity, credibility, resonance, usefulness, and transferability,"* are paramount for action research studies [SIG25]. Regarding *reflexivity*, the sections of this chapter systematically examine potential biases by evaluating common threats to validity discussed in Staron's book with regard to this thesis [Sta20]. To enhance *credibility*, we have supplied comprehensive additional materials to all publications, particularly to publications No. 6 *(Establishing TDM)* and No. 7 *(Practical Guide)*, which report on the action research studies. *Resonance* is addressed through the reflective segments of each workshop, i. e., the retrospectives (see Section 4.2.2.1). To ensure *usefulness*, we provided sufficient information to replicate the study and detailed descriptions of the workshops in the publications. This thesis, further, presents the TDM Guide in Chapter 16 aimed at practitioners, who plan to adopt the basic TDM process. *Transferability* is an important element of qualitative research, characterizing the extent to which research findings can be applied across different contexts. In our scenario, transferability is facilitated by replicating the workshops to establish a TDM process in three companies. Unlike merely providing a fixed TDM process, our workshop concept's adaptability allows participants to select strategies that best suit their contexts, thereby further supporting its transferability. In contrast, *generalizability* is the extent to which conclusions from quantitative research with a representative sample can be extrapolated to the broader population, e. g., by surveying 1,000 randomly selected participants and generalizing the results to a larger population. In this regard, the empirical standards for action research mention that *"positivist quality criteria such as internal validity, construct validity, generalizability, and reliability generally do not apply"* [SIG25]. However, we clarify and discuss these criteria to the extent possible in alignment with the reflexivity criterion.

In the following, we present an overview of all threats that apply to this thesis, and how we mitigated them or the limitations they impose.

8.1 Construct Validity

First, we might succumb to the "mono-operation bias." Even though we replicated the study three times in different contexts, new contexts might yield additional insights or different results, particularly in other countries and cultures. The workshop concept and the TDM Guide, including the specified *best practices*, will prove their usefulness over time and with increasing use. Nonetheless, we presented approaches that proved useful in practice multiple times and provided a first step in the right direction.

Second, we could not completely eliminate "interactions of different treatments" or the "multiple treatment interference". The iterative approach was designed to focus on one TD activity at a time, which not only reduced practitioners' cognitive load but also allowed us to analyze each workshop's outcomes separately. However, changes in general working processes might have influenced both the research results and the TDM establishment. For example, the first team was newly assembled and still figuring out their overall processes, and in the second team, the chosen TD manager left the company during the study. These events may have impacted the results. We were able to mitigate this threat to some extent by replicating this study in three contexts and comparing the results.

Third, the "interaction of testing and treatment" may have impacted our study. Given the nature of action research studies, this interaction is integral to their basic design. Recording the meetings and the researchers' presence in some meetings might have influenced the participants' awareness of the TD topic, making this particularly a threat to RQ3 *(Prevention)*. We mitigated this issue to some extent by attending meetings intermittently rather than attending every meeting, and by recording them instead.

Finally, our study may be affected by "hypothesis guessing" and "evaluation apprehension." This means that the practitioners are aware of the study's purpose and the fact that they are being observed, which could lead them to modify their behavior. Following Staron's guidelines [Sta20], we mitigated these threats by separating the team into the action and reference teams and we reiterated the importance of the reference team and open criticism at the beginning of each workshop in order to offset some of the effects.

8.2 Internal Validity

First, inquiring about problems before the study and its interventions start may create the impression that a problem exists, which Staron referred to as a "testing threat." Our initial survey included questions regarding the existing TDM process, with the expected result that there is none. Though this brought the problems to light, the choice to take action and implement a TDM process was determined prior to the first survey, as the pressure from the consequences of TD was already significant.

Third, a "biased selection of subject" impacts this thesis. Typically, only industry teams that are already facing a particular issue will collaborate with researchers. This suggests that our findings may benefit other teams ready to invest in a TDM process, but may not be helpful for teams that are not ready to commit. Therefore, our conclusions are limited to teams that are prepared to invest in a TDM process.

Finally, we might fall for the "John Henry effect", i. e., that participants compete with a predetermined baseline. However, the baseline, i. e., the current TDM process, was especially low, i. e., a TDM process was almost non-existent. In our cases, participants recorded TD as tickets, but there was no TDM process established, and the tickets got lost in the backlog. Therefore, we assume that this effect is of little relevance to our results.

8.3 External Validity

First, we analyzed a team that actively sought to address the issue of TDM and that received approval from the team managers, which represents a form of "selection bias" similar to what is discussed regarding internal validity. This limits the applicability of this research to teams that are equally inclined to implement a TDM process and to invest the necessary time. For this purpose, they need to have at least some backing from their managers. Nonetheless,

the goal of this research was not to create a TDM process in teams that are resistant to it, but to support interested teams.

Second, Staron describes the "reactive effects of experimental arrangements", which take place when researchers heavily influence the context. We intervened in a controlled manner by sharing specific research findings, but we were unable to present all the research outcomes related to TD. Nonetheless, the practitioners had the freedom to select the methods they wished to implement, and the primary objective of the workshop approach is to empower participants to decide what aligns best with their environment without restricting the scope to a singular process. Furthermore, we attended the team's meetings and were occasionally consulted on the nuances of the process for more direct feedback, or, on rare occasions, we intervened deliberately to remind the team of process-related commitments. We see this as the most significant threat to validity and limitation, as the results can only be achieved with external support from persons knowledgeable in the TD topic, as also indicated by the participants in their final survey.

Finally, Staron points out "constructs, methods, and confounding factors", which means that the measurements and methodologies may not permit us to generalize beyond the specific context of the study. In our research, the approach is confined to agile working teams, as we embrace the concept of iterations and team-based decision-making. This strategy would be less effective in contexts where the team lacks the authority to make such decisions, for instance, in cases where a central group like architects or team managers directs these process decisions.

8.4 Conclusion Validity

First, the action research studies may have been impacted by the risk of overlooking crucial observations, i. e., "missing the needle in the haystack," and "seeing things that aren't there", i. e., noting observations irrelevant to our intervention. These risks were mitigated by having two researchers synchronize their observations. Although not every observation was conducted by both researchers, they frequently reviewed each other's findings to confirm validity and ensure that important information was captured. Nevertheless, it is possible that some observations were overlooked. However, we assume that the overall tendency remains valid.

Second, the "low reliability of measures" may have impacted the TD-SAGAT findings. We conducted the TD-SAGAT survey once each month per team and collected approximately three responses for each participant and meeting. This resulted in a small number of data points each month. Nevertheless, aggregating the results across all teams and merging the results gathered between two workshops somewhat mitigates this concern.

Finally, in certain instances, we observed matters related to the "random heterogeneity of respondents." Sometimes the interpretations of the questionnaires varied at first. Nevertheless, researchers were available during the completion of the questionnaires and could address any ambiguities, thereby mitigating this threat. For example, we were able to determine the backlog issue for which the TD-SAGAT questionnaire needed to be filled out.

8.5 Summary of Limitations

To summarize, the workshop concept as well as the use of the TDM guide are limited to teams that are committed to change, have the time to invest in the establishment and the support of – at least – their direct manager. Furthermore, the workshop concept is limited to agile-organized teams that are empowered by their leader to change their processes. The biggest limitation is the presence of an external support knowledgeable in the TD topic for the workshops. Similarly, as some participants noted, initial support during meetings appears to be an essential factor in the success of establishing TDM.

Future research might show whether teams can develop a TDM process based on the TDM guide without external support. Future adoption of the workshop concept, the TDM Guide, and the defined *best practices* in practice will further demonstrate their usefulness and limitations.

Chapter 9

Conclusion and Future Work

9.1 Conclusion

This thesis aimed to examine why TDM research concepts are not commonly used in practice and what is missing to address this gap. The focus of this thesis is on four (socio-technical) aspects of TD, i. e., TD communication, TD awareness, TD prevention, and, finally, establishing a TDM process. The thesis comprised three research steps: the analysis of the research area to identify the requirements for or hindrances to a TDM process, the design of the five-step workshop concept to establish TDM, and the evaluation of this concept in the form of three action research studies and the deduction of a TDM guide for practitioners from the results. The thesis contributes:

- Various methods to improve TD communication, e. g., a model for handling and communicating architecture mistakes, the TD Game, or an introduction on how to create a 3-minute pitch slide,
- A method to measure TD awareness based on its definition as awareness in decision-making situations, and the results of this evaluation, which were used to improve the TDM process.
- A repayment method that leads to TD prevention, as well as techniques to debias decisions and thus avoid TD-incurring decisions
- A workshop concept to establish a customized TDM process and the results of the conducted workshops in the form of a TDM Guide with the commonly adopted approaches as a basic *best practices* TDM process.

We further identified multiple research gaps, including the identification and management of TD across organizational levels, the integration of multiple interests, and the debiasing of design-level decisions, including the support for practitioners in identifying alternative solutions.

9.2 Future Work

For future work, we identified the aforementioned research gaps, outlined in Section 6.1.7, that should be addressed within the TDM process. In addition, this section outlines several ideas for future work that directly build on this thesis, as explained below.

9.2.1 Evaluating the TDM Guide

We developed the TDM guide based on the research findings from workshops and action research. The TDM Guide, as presented in Chapter A, includes:

- Foundational information on TD,
- The so-called *best practices*, which are the TDM approaches all teams adopted to create a basic TDM process,
- The *nice-to-have* approaches, which some teams adopted to augment this basic TDM process,
- An explanation of how to establish a TDM process step by step

- A list of common questions with respective answers
- A list of common challenges with respective suggestions on how to overcome those
- An outlook on how this process can be scaled company-wide.

Although this information builds on the research findings, we were unable to evaluate the effectiveness of using this guide to establish a TDM process as part of this thesis. Furthermore, participants indicated that the external support from researchers was essential for the success of the workshops.

The next research step would be to have teams – first from the collaborating companies, then from other companies – use this TDM Guide to establish a TDM process. Due to the lack of reliable metrics to assess a TDM process, success could be measured by team and company satisfaction with the resulting process, as assessed through surveys administered to team members and companies' IT managers. Additionally, observations of team meetings conducted after the process was successfully established might support the survey findings. If teams are unable to establish a TDM process in accordance with the TDM Guide, the survey may request reasons to refine the guide further. In this way, the study would also be able to evaluate whether external support was indeed crucial to success.

9.2.2 Streamlining and Reevaluating the Workshop Concept

We noted that the content complexity varied across the five workshops. The second workshop included two distinct TD activities (documentation & prevention), whereas the third and fourth workshops focused on a single topic each (prioritization and repayment). Since TD prioritization and repayment are closely related, it may be beneficial to merge these topics into a single workshop to minimize the number of person-days required to set up the process.

Additionally, the workshops were heavily influenced by elements of action research. In particular, the introduction of evaluation results during the workshop mornings was deemed superfluous by some participants. Additionally, these were among the workshop elements in which the researchers interacted most with participants. These elements might be removed to streamline the workshops. In this case, the workshops would focus on three elements: the retrospective, i. e., the "inspect&adapt" part; the introduction of new information on TD activities; and the discussion of newly introduced approaches, including the identification of measurements to change the existing TDM process accordingly. A shortened agenda suitable for practitioners is presented in Table 9.1. For the first workshop, no element can be removed. Consequently, the first workshop should take one day, while subsequent workshops would require only half a day.

Start Time	Agenda Topic	Format	Material	Duration
09:00 a.m.	Recap Study: What is the workshop's general procedure? What have we achieved so far? What is still missing?		PPT	15 min.
09:15 a.m.	Retrospective	Love it – Change it – Leave it	Flipchart	60 min.
10:15 a.m.	Break			15 min.
10:30 a.m.	Introduction of a new TD activity and its possible approaches		PPT	30 min.
11:00 a.m.	Discussion of the new topic & identification of measurements	1 – 4 – all	Flipchart	60 min.
12:00 a.m.	Gathering todos & assigning responsible persons		PPT	15-30 min.
12:30 a.m.	End			

TABLE 9.1: Shortened workshop agenda without action research elements

The reevaluation of the success of the streamlined workshop concept could follow the approach introduced above for the TDM Guide evaluation.

9.2.3 Online Course for TD Management

To make the TDM Guide more interactive, we have begun developing an online course on establishing TDM that builds on the workshop material and the TDM Guide. Both approaches

rely heavily on the team's decisions and interactions. Yet, online courses are usually performed by individual learners. Accordingly, an online course on establishing TDM should be structured differently, incorporating asynchronous learning of the workshop content and synchronous discussions to determine an approach feasible for the team. Developing a concept for this purpose and evaluating the success of this approach would allow for a lower-level approach than the workshops and a more interactive approach than using the TDM Guide. Additionally, all industry partners welcomed the proposal to use an online course for this purpose. We suggest intensifying the research on this topic for future work.

9.2.4 Scaling TD Management

The establishment of a TDM process, as outlined in the TDM Guide or developed through the workshop model, addresses individual teams and yields a customized TDM process for each team. Similarly, the additional ideas of using a streamlined workshop approach or an online course to establish a TDM process offer a customized approach for a single team. In terms of maturity models, this results in a "systematic" level as defined by Martini et al. [MBB18] or a "level 4" as defined by Kruchten et al. [KNO19]. At these levels, each team can gain advantages in its everyday operations:

- Overview of TD within the team
- Estimation of risks arising from TD
- Prioritization of TD issues against one another
- Defined and uniform repayment of TD
- Capacity to provide details and arguments to management for negotiating investments

However, in a corporate setting, it is worthwhile to explore whether and how a resulting process can be applied broadly, meaning to various or all teams within the organization. This would align with reaching the "institutionalized" [MBB18] level or "level 5" [KNO19]. The business and IT management would gain additional benefits when the data generated by multiple teams is presented in an aggregated format:

- Overview of TD per team and per architectural component, allowing to identify "hot spots"
- Estimation of the risks originating from TD in cross-team projects, components, products, or supporting tools
- Prioritizing support for overburdened teams, e. g., by increasing team size or reducing the number of customer requests to allow time to repay the TD.
- Defined and therefore predictable procedure for repaying TD

Unfortunately, the workshop concept has not yet reached a level of scalability suitable for large organizations. Hosting workshops for every team in a company could become expensive. Conducting interactive online courses with all teams is not much less time-consuming. Furthermore, the introduced methods would enable each team to develop its own process, thereby hindering uniform evaluation across the organization, as explained above. Hence, it is crucial to discover methods to improve the scalability of this approach.

One approach could involve conducting workshops with one or a few representative teams, and then qualifying other teams to adopt the developed process, using the issue type and visual representations created. The TD manager(s) of the representative team(s) could introduce the process. We identified two probable ways to motivate other teams to use the developed TDM process:

1. Distribution of the process "from the top down," such as through management or architectural committees, i. e., the team architects.
2. Distribution of the process "from bottom up", i. e., from within the company through a CoP focused on the subject of TD. This strategy may increase intrinsic motivation for change, leading to greater engagement and greater teams' assimilation of the process.

Together with one industry partner, we already established a CoP and evaluated the first meetings. However, because a CoP relies on self-organization, it cannot be guaranteed that

the process developed by one team will be adopted. Nonetheless, elements of this process could be used to develop an alternative process with broader acceptance. Consequently, the development of such a CoP, the TDM process that results from it, and the differences from the initial process may warrant future research. With another industry partner, we are currently pursuing the alternative approach of engaging with the company's management to implement a guideline on TDM. Similarly to the CoP approach, the development of this approach, the TDM process resulting from this, and the differences to the initial process might be worth researching in the future. Additionally, it may be useful to compare the two approaches with respect to the suitability of the resulting process and the rate of adoption within the company. For the adoption rate, it is relevant to assess actual adoption, i. e., whether teams engage in the process or merely fulfill the minimum requirements. The path chosen probably depends on corporate culture. Thus, it might be a final research endeavor to analyze which approach fits best in which context, i. e., company culture.

Chapter 10

Preventing Technical Debt by Technical Debt Aware Project Management

1 *(TAP Framework)*	
Title	Preventing Technical Debt by Technical Debt Aware Project Management
Authors	Wiese M., Riebisch M., Schwarze J.
Year	2021
Conference	Conference IEEE/ACM International Conference on Technical Debt (TechDebt)
Reference	[WRS21]

TABLE 10.1: Publication No. 1 *(TAP Framework)*

Preventing Technical Debt by Technical Debt Aware Project Management

Marion Wiese
Universität Hamburg
Hamburg, Germany
wiese@informatik.uni-hamburg.de

Matthias Riebisch
Universität Hamburg
Hamburg, Germany
riebisch@informatik.uni-hamburg.de

Julian Schwarze
Gruner + Jahr GmbH
Hamburg, Germany
schwarze.julian@guj.de

***Abstract*—Technical Debts (TD) are problems of the internal software quality. They are often contracted due to tight project deadlines, for example quick fixes and workarounds, and can make future changes more costly or impossible. TD prevention should be more important than TD repayment, because subsequent refactoring and reengineering is usually more expensive than building the right solution from the beginning. While there are numerous works on TD repayment, solutions for TD prevention are understudied. This paper evaluates a framework that focuses on both TD prevention and TD repayment. It was developed by and applied in an IT unit of a publishing house. The unique contribution of this framework is the integration of TD management into project management. The evaluation was carried out by a study based on ticket statistics and a structured survey with participants from the observed IT unit and a comparison unit. The evaluation shows that the adoption of this framework leads to a raised awareness for the contraction of TD. This results in benefits like more rational discussions and decisions, TD prevention and timelier repayment of TD tickets.**

***Index Terms*—Technical Debt, Technical Debt Awareness, Technical Debt Aware Project Management, Technical Debt Repayment, Technical Debt Prevention**

I. INTRODUCTION

In their definition of Technical Debts (TD), Avgeriou et al. [1] describe TD as "a collection of design or implementation constructs that are expedient in the short term, but set up a technical context that can make future changes more costly or impossible." In a technical metaphor to financial debt a sub-optimal implementation or design is interpreted as debt. The resulting problems are interpreted as interest rates, the refactoring as repayment of the debt and the refactoring cost as principal.

Many papers focus on TD repayment while the topic of TD prevention is understudied ([2]). However, in a long-term view, it can be expected that it is cheaper to implement the optimal solution right from the beginning especially when considering possible interest payments ([3], [4]). Therefore, TD prevention is worth more consideration ([4]).

The originally named cause for TD contraction is the problem of tight project deadlines as described in the paper of W. Cunningham ([5]). Other causes have been identified by different research papers ([6], [2], [1], [7]), e.g. bad design decisions, unavailability of a key person, neglected technical improvements, lack of education or parallel development.

Nevertheless, tight timelines are the TD cause that was identified by many researches as the most pressing cause ([8], [4], [7], [6], [1]). However, to the best of the authors notice no studies provide a solution for the problem of tight timelines that is feasible in an industry environment.

A good example of the timeline problems can be taken by the German government's reduction of value-added taxes from 07/01/2020 to 12/31/2020 as part of the management of the corona crisis. The corresponding law was only passed two days before it came into force. To reflect this change in the software systems, adjustments to these systems were necessary in many German IT units, which were therefore subject to a very tight schedule. Well-structured project planning or adoption of good development practices were mostly not possible.

In many cases, the constant time pressure leads to a phenomenon in developer behavior. In preemptive obedience the developers search for the fastest and easiest solution assuming a tight schedule, even when there is no time pressure. In these cases, it is not common practice to step back and think about or discuss different solution options. There may be frequent situations where the sub-optimal but faster to implement solution must be chosen due to project timelines. However, it should be a goal to make it a conscious decision of the whole team to choose the optimal or sub-optimal solution.

At this point, TD prevention has a good chance for success by introducing a special form of tickets. These tickets that we call TD tickets make the contraction of TD explicit. The awareness for the contraction of TD is raised among all team members including business analysts and managers. By this, a basis for TD prevention is created.

Furthermore, TD tickets integrate TD management into project management. Project managers become responsible for the repayment of the TD they contracted during their project and therefore get an extrinsic motivation to reduce the contraction of TD. By this, this paper presents a solution to the problem that is presented by Martini et al. ([7]) as "*Split of budget in Project budget and Maintenance budget boosts the accumulation of debt*".

The evaluation in this paper shows that TD tickets prevent TD and achieve a timely repayment of TD items.

These TD tickets are part of a framework developed in industry and a new contribution to the State of the Art. Further parts of the framework are continuous TD repayment and

978-1-6654-1405-0/21/$31.00 ©2021 IEEE
DOI 10.1109/TechDebt52882.2021.00018

maintenance projects which are already described in some research papers, e.g. by Steve McConnell ([9]).

The contribution of this paper is therefore the presentation and evaluation of the TD tickets as a means of TD prevention. The evaluation consists of a survey and ticket statistics. It will provide evidence that TD prevention and timely TD repayment can be reached by adopting this framework. Other benefits like more rational discussions and decision-making are highly appreciated side-effects.

In the following section we will introduce the framework. Section III will present the research questions (RQs) regarding the effectiveness of the framework that will be answered by this paper. The methodology of the evaluation will be explained in Section IV. The results of the evaluation will be presented in Section V and comprise ticket statistics and the results of a survey carried out in the observed IT unit and a comparison unit. The results will be discussed in Section VI. In this section the RQs will be answered, threats to validity will be discussed and the paper will be embedded in related work. The paper ends with the conclusion and future work in Section VII.

II. FRAMEWORK PRESENTATION

A framework for managing TD has been developed and established in an IT unit in the beginning of 2018. It has been utilized ever since. The IT unit resides in a German publishing house with a size of more than 9000 employees worldwide.

In this IT unit, TD accumulation leads to increasing cycle times, unnecessary errors in the systems, and discontent of the developers. The biggest problems of this IT unit are tight timelines. There is a high market pressure in this industry sector and contracts or laws need to be fulfilled in time. The tight timelines often mean that "*well-structured project planning*" or "*adopting good development practices*" as proposed by [8] are not sufficient.

These problems are recognized by the management and sufficient priority to handle TD is given. The IT unit including the management is willing to change its processes. The main goal for this change is predefined by the manager of the IT unit: TD have to be prevented in any case where this is possible without impairing the deadlines. Whenever a set deadline leads to TD, this TD shall be repaid timely.

On this basis, a framework to help manage TD was created. Especially, two factors of this framework shall support TD prevention: the overall raised awareness for TD contraction and the integration of TD management in the project management process. The resulting framework consists of four management categories for TD management which are presented below.

A. Management Categories

The framework comprises four management categories for handling different kinds of TD related tasks. There is a fifth management category for handling all functional requirements which will not be presented in detail as it does not deal with TD.

The different tasks are recorded as tickets in a comprehensive project backlog and tagged and handled depending on their category. The framework includes guidelines for the recording and processing of these tickets depending on their management category. It was decided that the person that has the most interest in the completion of the tasks of one category should be the one that is responsible for this category. These may be architects or business analysts. Table I shows an overview of the management categories, their responsible persons and repayment terms.

1) Maintenance: In this framework, tickets of the maintenance category are technically driven tickets that do not have a direct impact on the user's perception of the system. [1]

Recording: Every maintenance ticket must contain a one-sentence information that describes the impact of the ticket. The description must be written in a way that is understandable for business analysts and managers. *Processing:* Ten percent of the planned capacity of every sprint can be invested for maintenance tickets according to the architect's decision. *Goal:* The goal of the maintenance tickets is to repay the unintentionally contracted TD continually and to allocate time for other maintenance tasks, e.g. technical adaption.

Examples for this category are the upgrade of a third-party-library or the refactoring of bad structured code.

2) Maintenance Project: All tasks that belong to the maintenance category as stated above but that require more than five days development time are handled as a maintenance project. This is based on the practice that is also applied for functional requirements and business projects.

Recording: A maintenance project roadmap is produced and prioritized by the architects. *Processing:* The roadmap is part of the overall project roadmap as defined and prioritized by the unit managers. *Goal:* The goal of maintenance projects is to allocate enough time to conduct maintenance tasks with a long duration or maintenance tasks requiring more organization, e.g. when more than one team is involved.

An example for this category is the version upgrade of a central database that is used by more than one application. Another example would be the change of the underlying architecture, e.g. from a SOA to a microservice architecture.

3) Technical Debt: Tickets of this category, so-called TD tickets, describe tasks that are necessary for internal software quality during the implementation of a functional requirement, e.g. time for clean coding, good design or to comply with a specified architecture. These tasks are not mandatory to implement the required functionality and can therefore be implemented after a reached deadline if this is necessary for the project.

Often two or more solutions are discussed, where one follows standards and the specified architecture and the others more or less deviate from the optimal solution but may

[1]We use the term "maintenance" as it is often used in agile organizations. However, implementation of new or changing functional requirements is called adaptive maintenance in research, standardization, and sometimes in classical organizations.

TABLE I
TICKET CATEGORIES, RESPONSIBLE PERSONS AND TIME

Management Category	Responsible	Recorded By	Payback Time	Contingent
Maintenance	architect	developer	continually / architect decision	maintenance (10%)
Maintenance Project	architect	architect	management decision	project
Technical Debt	business analyst	all	after deadline, part of project	project
Deconstruction	architect	all	as soon as possible	project / functional req.
Functional Requirement	business analyst	business analyst	business analyst decision	project / functional req.

therefore be faster to implement. This idea of TD follows its original introduction by Cunningham ([5]).

Recording: In such cases, it is consciously decided to record two tickets. One ticket, a functional requirement ticket, describes the sub-optimal solution to be implemented before the deadline. By this ticket, TD is contracted. The other ticket, the TD ticket, describes the optimal solution and the tasks to repay the contracted TD. These TD tickets are usually identified during estimation meetings when the details of a functional requirement are discussed in the team and the effort is estimated. *Processing:* The uniqueness of this framework is that the repayment of intentionally contracted TD is part of the project plan. Thus, no project can be finished before the TD tickets are processed. After reaching the planned milestone for the main deployment and keeping to the tight schedule, the project plan does not end. Instead, TD tickets are prioritized and repaid in a subsequent project phase. This can be seen in Figure 1. This transfers the responsibility for the TD repayment to project management and thereby integrates TD management into project management. *Goal:* Evaluating different solution options in advance raises the overall awareness to only take on TD intentionally and prevents unnecessary TD. Furthermore, by making the project manager responsible for the TD accumulated during the project, his willingness to contract TD decreases. In other words, a feedback loop is generated and the behavior of the team and the project manager related to TD contraction changes. Additionally, the project and unit managers get an overview of the accumulating technical debt of a given project while it is still running. Thereby, the managers are given an opportunity to intervene or adjust the project plan and the following projects early on. Finally, by this approach TD are repaid timely.

For the example from Section I this may mean that, to adhere to the schedule, hard coded value-added taxes are adjusted in the code and are not refactored to a flexible solution. The corresponding TD ticket to introduce a centralized tax variable, is recorded during the estimation meeting. This TD will then be repaid in an orderly manner after the law came into force and in the "TD repayment phase" of the "tax adjustment" project. On the other hand, as different options for these changes are discussed during the estimation meeting, this can also mean that due to the low effort the centralization may be the better option and TD are prevented.

4) Deconstruction: A ticket of the deconstruction category is a special kind of TD ticket that cannot be processed during the project. After the implementation of a new solution there

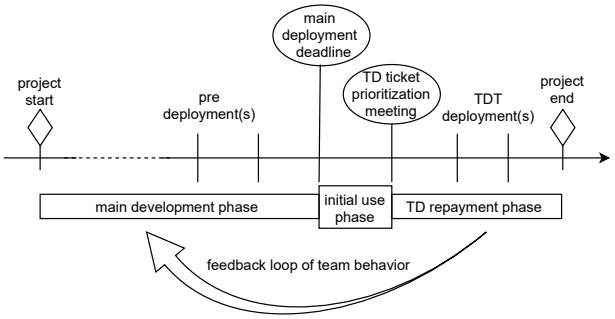


Fig. 1. Project plan with included TD repayment phase

is sometimes the need to keep legacy code in parallel for the time being. These tickets then comprise the deconstruction of this legacy code when it is no longer needed. They can also comprise the deletion of deprecated code parts after all consumers of this parts are detached.

Recording: These tickets have to contain an additional information describing the time when the deconstruction can take place. *Processing:* The deconstruction ticket has to be repaid as soon as possible as part of the contingent for functional requirements. *Goal:* The main goal of this tickets is to avoid cluttering of code and by this support code comprehension.

An example of this kind of tickets is the need for a new business rule for the next business year. The application should not be deployed on New Year's Eve to minimize deployment risks. Therefore, the application will be deployed in advance and must contain a date-driven switch to choose between the old and new business rule. When the old business rule is no longer needed it can be deconstructed.

III. RESEARCH QUESTIONS

As described in Section I, two main problems concerning TD are the questions on how to prevent TD and how to deal with TD despite having a tight schedule and fixed deadlines. For this question, the described framework provides a possible solution found and developed in practice. With subsequent research questions (RQ) this paper provides an evaluation of this framework. Especially its feasibility and its effectiveness regarding the aforementioned problems is evaluated.

*(**RQ1**) Framework application*

- *(**RQ1.1**) Do practitioners find the framework reasonable?*
- *(**RQ1.2**) Are the processes of the framework feasible in practice?*

This will be the basis for all other evaluations as it does not make sense to further analyze the impact of the framework should it not be feasible in practice. These questions will be evaluated by a survey of the members of the observed IT unit.

*(**RQ2**) Framework effectiveness*

- *(**RQ2.1**) Does the framework lead to raised awareness for the contraction of TD?*
- *(**RQ2.2**) Are the TD items taken on more consciously when using the framework?*
- *(**RQ2.3**) Are the TD items paid back timely?*

These aspects are the goals of the framework as presented in Section II. The evaluation of this question shall show whether these goals were reached. The evaluation is conducted by a survey and the tickets statistics (for the last part).

*(**RQ3**) Framework benefits*

- *(**RQ3.1**) Can TD be prevented by the adoption of the framework?*
- *(**RQ3.2**) Are there other benefits arising from the adoption of the framework?*
- *(**RQ3.3**) Do these benefits justify the additional effort?*

The purpose of these questions is to gather the impact of the framework. The main goal of TD prevention is evaluated, as well as secondary benefits concerning team discussion and decision-making process. The correlation between the survey variables will be used to answer this RQ. Finally, we ask for the justification of the additional effort in relation to the benefits to substantiate the feasibility of the framework.

IV. RESEARCH METHOD

The presentation and evaluation of the aforementioned framework is the contribution of this paper. The presentation took place in Section II. To answer RQ 1.2 and RQ 2.2, we evaluated the recording and processing statistics for TD tickets and maintenance tickets. All other RQs are answered by the results of a survey. The survey targets participants from the IT unit that uses the framework (observed IT unit) and a comparison unit that does not use the framework.

A. Ticket Statistics

To evaluate RQ 1.2 and RQ 2.2 the number of tickets are presented in terms of categories, priorities, and time as descriptive statistics. As the backlog includes some old and invalid tickets the extracted data had to be cleaned up manually first. This was carried out in consultation with the architects of the unit. The statistics for maintenance projects and deconstruction tickets are not evaluated because there is still insufficient data for this.

B. Survey

The method of a survey was chosen to benefit from the input of all participants that are willing to share their experience. For all RQ's corresponding questions were asked in this questionnaire.

TABLE II
COMPOSITION OF UNITS AND SURVEY PARTICIPANTS

	observed unit		comparison unit	
	members	particip.	members	particip.
manager	2	n/a	2	n/a
architect	2	n/a	1[a]	n/a
business analyst	8	6	1[a]	1
developer	15	9	5	4
operations	5	2	0	0
sum	32	17	8	5

[a]architect and business analyst are the same person

1) Survey Participants: All members of the observed IT unit are asked to fill out the questionnaire. Additionally, a comparison unit is asked to complete the questionnaire. The comparison unit is led by the same unit manager which increases the comparability.

The results of the comparison unit gave us the opportunity to validate the descriptive statistics. The participants were asked to which team they belong, but due to works council regulations not which role (e.g. architect, manager) they inhibited. The composition of the observed unit and the comparison unit are presented in Table II. The response to the survey is also shown in this table. All questions and evaluations of the survey can be accessed in the additional material[2].

2) Questionnaire Construction: To avoid misunderstandings the questionnaire starts with a short information about the term TD. For statistical reasons, background information, e.g. the team membership of the participant, are queried. The questionnaire is further divided into two main parts: (I) assessment of the framework and (II) effects and benefits of the framework.

Part (I) shows the practical feasibility of the framework and provides the answers to RQ1.1 and RQ1.2. For every management category of the framework the same set questions regarding the reasonableness and feasibility are asked. Each part starts with a short information about the management category as it was also documented in the IT unit's wiki. Part (I) was not filled out by the comparison unit as these participants did not use the framework.

The second part comprises four subsections: TD Awareness (RQ2.1), Comparison of optimal and sub-optimal solutions (RQ 2.2), Observed benefits of comparison (RQ3.1 and RQ3.2), and Justification of the additional effort (RQ 3.3).

All parts comprised a set of assertions that the participants were asked to validate using the following Likert scale ([10]): applies - rather applies - rather does not apply - does not apply - cannot answer. The *cannot answer* option was given as some questions could not be answered by all members of the unit. The participants were explicitly asked to only use this option in these cases.

Finally, two more open questions in the end gave the participants an opportunity to point out especially good parts of the framework and parts that need improvement.

[2]https://doi.org/10.5281/zenodo.4616485

A first questionnaire was filled by two developers that knew about the framework as a pilot test. Based on this, the questions were reduced and focused. Furthermore, one manager gave feedback on the construction of the optimized questionnaire which was also incorporated.

3) Survey Evaluation: The questionnaire is evaluated using the statistics software SPSS[3] for closed questions and the qualitative research software MaxQDA[4] for open questions.

First, descriptive statistics for all closed questions are created for an exploratory data analysis. For simplification and analysis purposes the answers for the questions of part (II) are dichotomized. This means the answers *applies* and *rather applies* as well as *rather does not apply* and *does not apply* are summarized. All values are presented as percentage to align the output of the observed unit and the smaller comparison unit. For the sake of clarity, only the *applies* answers are shown in the following figures.

As a second step, hypothesis for RQ2 are developed on the basis of this exploration and the significances of the hypotheses are evaluated. The Mann-Whitney U-Test ([11]) is used because it can be applied for not normally distributed data on an ordinal scale. Only significant findings are presented.

Regarding the comparison unit, the benefits (RQ 3.2 and RQ3.3.) cannot be assessed but must be interpreted as expected benefits. As a result, no differences between the units can be evaluated regarding the benefits. Hence, it is interesting to analyze correlations between all survey participants who take on TD consciously and the benefits they assessed. These correlations are evaluated using the ϕ-coefficient of Pearson's $\tilde{\chi}^2$-correlation ([12], [13]) for dichotomous variables. Significant and strong correlations are deduced and presented.

Lastly, the open questions were evaluated using open coding.

V. RESULTS

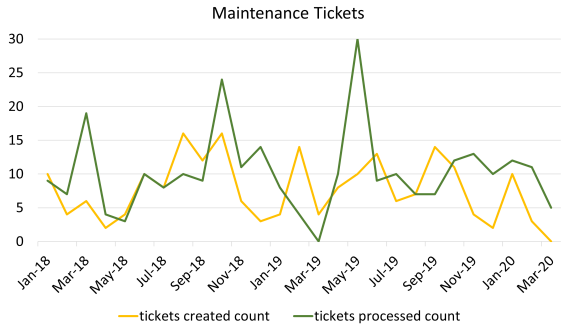
A. Ticket Statistics

The timelines in Figure 2(a) and Figure 2(b) show the development of ticket counts over time to answer RQ2.3. The maintenance tickets are created and processed more or less continually.

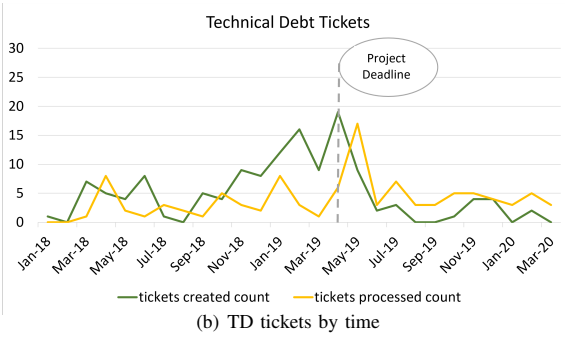
The timeline for TD tickets show a peak of contracted TD in spring 2019 due to a project with a tight timeline. No maintenance or TD tickets were processed in March 2019. Accordingly, the TD tickets as well as the maintenance tickets show a processing peak after the reached deadline at the end of April 2019. In summer and autumn 2019 the TD tickets with a lower priority are processed which means that this project is still not finished (see Figure 1). Other projects start in parallel to this project and development capacities must be divided between the projects.

B. Survey

1) Assessment of the Framework: Most of the survey participants agree that it is generally reasonable to record and process all four types of tickets (Fig. 3(a)/ 3(d)). Most of them also agree that the procedures of the framework for recording and processing are reasonable (Fig. 3(b)/ 3(e)). When asked if the framework is working as intended, the agreement decreases, but still, most of the survey participants state that the procedures work well and are therefore helpful (Fig. 3(c)/ 3(f)). For RQ1.1 and RQ1.2 this means that the framework is reasonable and feasible in practice.



(a) Maintenance tickets by time



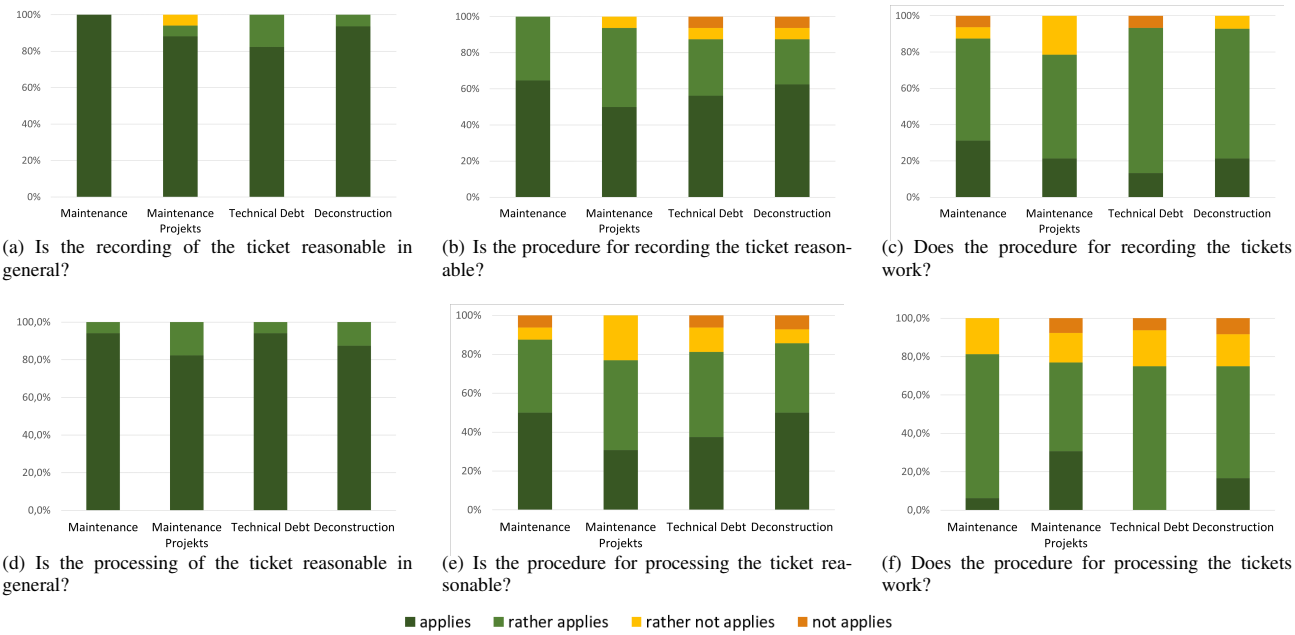
(b) TD tickets by time

Fig. 2. Ticket statistics with project deadline at the end of April 2019

2) Effects and benefits of the Framework: First, the survey participants are asked if they and their team usually recognize that TD items are or were contracted before or after the point in time when they were contracted. The goal of this question is to identify whether the participants are aware of the contraction of TD items to answer RQ2.1. As can be seen in Figure 4(a) there is no significant difference between the units. Both units state with more than 75.0% agreement that they usually recognize it when TD items are contracted on.

An obvious difference can be seen in Figure 4(b). The units are asked whether they compare different solution alternatives and whether they estimate respective implementation costs, principal, and interest rates for taking on TD. To answer RQ2.2, we assume conscious decisions to take on TD, if these comparisons are made. 75.0% of the participants of the observed unit agree to compare quick and optimum (or more) solutions and even calculate implementation cost for this comparison. For the comparison unit 80.0% survey participants do not agree to this which means they usually do not compare different solutions and estimate implementation costs.

[3]https://www.ibm.com/dede/analytics/spss-statistics-software
[4]https://www.maxqda.de/



(a) Is the recording of the ticket reasonable in general?

(b) Is the procedure for recording the ticket reasonable?

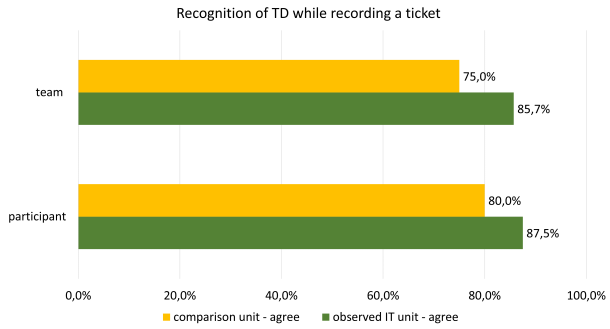
(c) Does the procedure for recording the tickets work?

(d) Is the processing of the ticket reasonable in general?

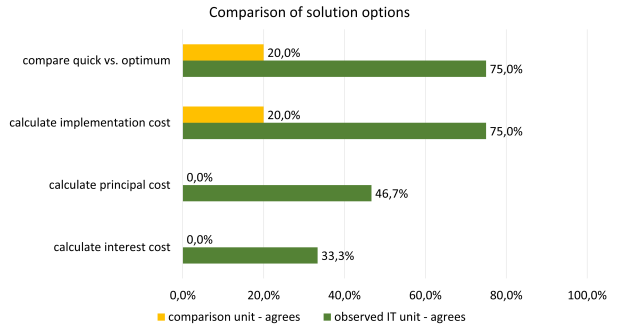
(e) Is the procedure for processing the ticket reasonable?

(f) Does the procedure for processing the tickets work?

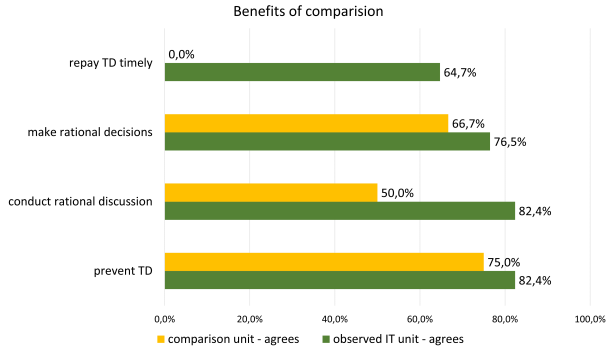
Fig. 3. Appropriateness and usability of the framework as assessed by the observed IT unit



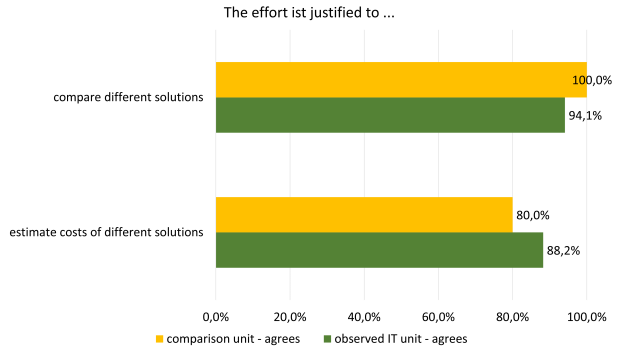
(a) Do the survey participants recognize that TDs are taken on while contracting them?



(b) To what extend are the quick/sub-optimal and the optimal solution compared?



(c) What benefits of the comparison can be found?



(d) Do the survey participants think that the extra-effort for the comparison is justified?

Fig. 4. Effects and benefits of the Framework - Descriptive Statistics

TABLE III
SIGNIFICANCE OF H_R-HYPOTHESIS

H_R	variable	exact significance
$H_{R.1}$	comparison	0.058
$H_{R.2}$	implementation cost	0.048
$H_{R.3}$	principal cost	0.002
$H_{R.4}$	interest cost	0.015

Many participants of the observed unit agree that they also estimate principal (46.7%) and interest costs (33.3%), while no participants of the comparison unit do this.

These observations lead to the following research hypothesis (H_R) in comparison to respective null hypothesis (H_0):

$H_{R.1}$: The observed unit compares quick and optimum solutions more often than the comparison unit. ($H_{0.1}$: The observed unit does not show a different behavior than the comparison unit in terms of comparing quick and optimum solutions.)

$H_{R.2\text{-}4}$: The observed unit estimates the implementation (and principal and interest) costs for different solutions more often than the comparison unit. ($H_{0.2\text{-}4}$: The observed unit does not show a different behavior than the comparison unit in terms of estimating the implementation (and principal and interest) costs for different solutions.)

Table III shows the significances of these hypotheses that were measured with the Mann-Whitney U-Test. The exact significance for $H_{R.2\text{-}4}$ hypotheses is below 0.05 therefore these hypothesis can be accepted as they are significant to the 5% significance level. This means the possibility that these hypotheses are wrong is lower than 5%. The exact significance for $H_{R.1}$ hypothesis is 0.058 which means the possibility that this hypothesis is wrong is only a little higher than 5%. It can be accepted to the 10% significance level. In summary, this means that the adoption of the framework leads to the named effects of a more conscious comparison of different solutions options and their respective costs.

To answer RQ3.1 and 3.2, the participants are asked what benefits they expect (comparison unit) or assessed (observed unit) from the comparison of different solution option and therefore of the conscious contraction of TD. Figure 4(c) shows that both units mostly agreed that TD could be prevented by this comparison. Especially the observed unit assessed that discussions were led more rational (82.4% agreement) and more rational decisions were made (76.5% agreement). Timely repayment (RQ2.3) of TD was not expected by the comparison unit but assessed by many participants of the observed unit (64.7% agreement). This may partly be due to a misunderstanding that the observed unit was unconsciously assessing the benefits of the entire framework and not just the benefits of the comparison of solution options. There is no obvious difference between the units regarding these questions which means no hypotheses can be derived.

To answer RQ3.3, the participants are asked whether the effort for comparing and estimating the different solutions is justified by the gained or expected benefits. Both units mostly think that the effort is justified (>80.0% agreement). There is also no obvious difference between the units as can be seen in Figure 4(d). No hypotheses are derived.

TABLE IV
CORRELATIONS OF EFFECTS VS. BENEFITS

(a) Comparing solutions vs. assessing TD prevention and rational decisions as benefit

		comparing solutions: not applies	comparing solutions: applies	ϕ-coeff signif.
TD prevention	not applies	42.9%	7.7%	0.419
	applies	57.1%	**92.3%**	0.061
rat. decisions	not applies	66.7%	7.7%	0.623
	applies	33.3%	**92.3%**	0.007

(b) Estimating the implementation costs of different solutions vs. assessing rational decisions and rational discussions as benefit

		estim. impl. costs: not applies	estim. impl. costs: applies	ϕ-coeff signif.
rat. discussions	not applies	50.0%	8.3%	0.471
	applies	50.0%	**91.7%**	0.035
rat. decisions	not applies	57.1%	8.3%	0.535
	applies	42.9%	**91.7%**	0.020

(c) Estimating the principal costs vs. assessing rational decisions and rational discussions as benefit

		estim. princ. costs: not applies	estim. princ. costs: applies	ϕ-coeff signif.
rat. discussions	not applies	41.7%	0.0%	0.456
	applies	58.3%	**100.0%**	0.049
rat. decisions	not applies	45.5%	0.9%	0.495
	applies	54.5%	**100.0%**	0.037

3) Correlations regarding the benefits of the framework: As shown in the previous section, the most significant difference that arise by adopting the framework is the ongoing comparison between different solutions for every new ticket. As pointed out in Section IV-B3, the benefits of the framework cannot be assessed by the comparison unit. To identify the benefits of the comparison, we therefore analyze correlations between all survey participants who state that they compare different solutions and the benefits they therefore assessed.

Table IV shows all relevant and significant correlations that can be found.

It is important to point out that a correlation between two variables usually lead to two possible interpretations. One interpretation could be that variable A is the cause and B is the consequence and vice versa. In the following part only the most likely and reasonable interpretations are presented.

Table IV(a) shows the correlation between participants that compare sub-optimal and optimal (or more) solutions and participants that have assessed the benefits of TD prevention and rational decision making. The correlation for TD prevention is medium strong (0.419) but with an asymptotic significance of 0.061 only significant to the 10% significance level. The correlation for rational decision making is strong (0.623) and with an asymptotic significance of 0.007 significant even to

the 1% significance level. This can be interpreted to mean that comparing different solution options leads to the named benefits. The possibility that this correlation does not exist, and our assumptions are wrong is higher for TD prevention ($<$10%) than for rational decision making ($<$1%).

Table IV(b) shows the correlation between participants that are estimating the implementation costs of different solutions and participants that have assessed the benefits of rational discussions and rational decision making These correlations are both medium strong (0.471 and 0.535) and significant to the 5% significance level (0.035 and 0.02). This can be interpreted to mean that estimating the implementation costs of different solutions leads to the named benefits.

Table IV(c) shows the correlation between participants that are estimating the principal costs and participants that have assessed the benefits of rational discussions and rational decision making These correlations are both medium strong (0.456 and 0.495) and significant to the 5% significance level (0.049 and 0.037). This can be interpreted to mean that estimating the implementation costs of different solutions leads to the named benefits.

4) Open Questions: Most comments were just made one time and are therefore just single opinions. There were two points that were mentioned three times each.

First, "TD is a topic" expresses that it is already helpful to talk about the topic of TD and not to ignore it which is related to the goal to raise the of the overall TD awareness. This was pointed out by one participant: "*I actually see the main advantage of a conscious handling of the topic in the matter as such, because otherwise it is a topic that is almost ignored in everyday development*".

The second code is "Sometimes TD is OK" where the participants were considering situations in which cases it is reasonable to keep TD, e.g. in legacy systems. The following was stated by another participant: "*If a system is to be replaced, then I consider technical debts to be justifiable if they disappear after 1-2 years anyway*".

One other point was mentioned two times which is "Evaluation is difficult". In research this topic is known as TD prioritization problem. It was phrased by one participant as follows: "*Even if the "textbook" says that the current implementation does not correspond to the architectural specifications, it still seems to be a personal opinion that decides on the fundamental correctness of the specification*".

Two relevant drawbacks were mentioned by one participant, respectively. One participant stated that the distinction of the categories and the separation of their responsibilities may lead to a separation of the team. "*Although we keep saying that we are a team, we often stay separate and the different tickets in particular help make the differences between the team parts visible*". Another participant thought that the framework "*(slightly) contradicts the agile principle of providing a functioning solution with company value as quickly as possible. At the same time, however, it levels the stable ground so that we can continue to work on new requirements quickly and without production errors. The advantage outweighs*".

VI. DISCUSSION

A. Framework application

RQ1.1/RQ1.2: *Do practitioners find the framework reasonable? Are the processes of the framework feasible in practice?*

Figures 3(a) to 3(f) show that there is very little disagreement. Most survey participants find the framework reasonable and think that it does work as indented. Only few participants think that it must at least be optimized. The ticket statistics (see Section IV-A) also indicate that the tickets are recorded and processed as intended. Furthermore, the framework is already used in the IT unit for more than two years. It can therefore be concluded that in an overall view the framework is feasible in practice and that the following findings based on this framework are valid.

B. Framework effectiveness

RQ2.1: *Does the framework lead to raised awareness for the contraction of TD?*

Figure 4(a) shows that most participant are aware of it when they contract TD. Yet, there seems to be no difference between the observed and the comparison unit. The answers to the open questions support the finding that the awareness was raised considerably ("TD is a topic") and that the participants started to consider when to take on or keep TD and when not to ("Sometimes TD are OK").

RQ2.2: *Are the TD items taken on more consciously when using the framework?*

Figure 4(b) reveals that in comparison the observed unit shows a significantly different behavior than the comparison unit. The members of the unit using the framework are comparing different solutions, i.e. optimal and sub-optimal, and their costs before deciding for one of them. They may still take on TD by choosing the sub-optimal solution, but this is a conscious decision.

RQ2.3: *Are the TD items paid back timely?*

If TD has to be contracted it is mostly paid back timely which can be seen in Figure 2(b). Even though not all TD tickets may be paid back during their respective projects as intended by the framework. Figure 4(c) shows that two thirds of the participants of the observed IT unit perceive that TD items are paid back timelier. Nevertheless, not all TD tickets are paid back and a third of the participants are still dissatisfied with the timely repayment.

C. Framework benefits

RQ3.1/RQ3.2: *Can TD be prevented by the adoption of the framework? Are there other benefits arising from the adoption of the framework?*

The most obvious effect of the framework is the conscious contraction of TD by comparing different solution options. In Figure 4(c) the benefits that arise by these are evaluated. Both units assessed or expected benefits of this comparison.

Furthermore, the correlations between the benefits and participants that did this comparison are significant. This paper was able to show that the comparison led to more rational discussions and decisions. The error probability for

this assumption is lower than 5%. The prevention of TD was assessed with 6.1% only to the lower 10% significance level which means that it is only a little less likely that this observation is valid. These benefits can therefore be referred to the use of the framework.

RQ3.3: *Do these benefits justify the additional effort?*

Finally, Figure 4(d) shows that most participants think that the additional effort for comparison and cost estimation is justified and thus backs up the usefulness of this framework.

D. Threats to Validity

Construct Validity: To enhance construct validity, we questioned and compared two units one of which did adopt the framework and one that did not use any method for managing TD. To ensure comprehensibility and to optimize the questionnaire, we did pilot tests. It could be seen as a problem that the survey did not directly but only in open questions ask for drawbacks. The questions whether the effort for the benefits is justified should have remedied parts this problem. Additionally, we not only evaluated the questionnaire but also the ticket statistics which further improves the construct validity, e.g. when evaluating the timely repayment of TD.

Conclusion Validity: The paper was able to show statistically significant differences between the observed and the comparison unit. We used SPSS and standard statistic techniques for the evaluation of the significance of the findings. On the other hand, the sample was small. The comparison unit was considerably smaller than the observed unit and had a slightly different organization. By choosing a unit that was led by the same unit manager this threat was minimized. The survey response was only about 50% of the respective units which can lead to deviations in the evaluation.

Internal Validity: Some valuable effects were observed, but it is possible that the same effects could have been observed in other IT units. This threat was minimized by the comparison with a unit that did not use the framework. Nevertheless, just the management of TD could have led to a better developer morale like presented in [14] and therefore to good ratings. This threat can and should be minimized by replicating the study in other constellations. As correlations do not directly lead to causalities, the mentioned interpretations of the correlations should be validated, e.g. by follow up interviews.

External Validity: It is an important benefit of this paper that the framework was developed and tested in an industry environment and that the framework has been used for a long time. Nevertheless, this was just a case study and can be biased. The framework adoption and the study should be replicated in other units and companies, especially in other economic sectors and countries. By providing the framework information in this paper and the questionnaire and raw data[5] the study can be replicated to back up the findings.

[5]https://doi.org/10.5281/zenodo.4616485

E. Related Work

TD Prevention: In a systematic mapping study in 2015 Li et al. [2] identified eight TD activities one of them being TD prevention which was with seven studies between 1992 and 2013 also one of the obviously understudied activities. In 2016 Yli-Huumo et al. [3] find TD prevention to be "*one of the most influential activities of the eight TDM activities that a development team can conduct*". In 2018 Rios et al. [4] started to fill this gap by indicating that TD can be prevented, and that TD prevention is a worthwhile research topic. In their follow-up work [8], they identified actions and impediments to TD prevention by evaluating a questionnaire filled by practitioners. The work did not provide ideas how to implement these actions in practice or how to deal with the impediments.

Timeline Problem: As already mentioned in Section I one of the main root causes for TD to be found in all papers that deal with TD causes is the problem of tight timelines (e.g. [8], [4], [7], [6], [1]). To the best of the authors knowledge no papers contribute a solution to this problem. By using TD tickets, this framework fills this gap and provides a practical method that has impact on two of the actions listed in [8]: (I) "*Following a well-defined project planning*" by integrating TD management into project management, (II) "*Having an effective team*" with "*good communication on the team*" as shown by the impact the framework has on team discussions. Codabux et al. [15] also list different strategies to prevent TD, e.g. "*Education and Training*", ''*Pair Programming*", "*Conformance to process and standards*", but they also do not provide a solution on how to stick to the strategies under time pressure. This framework puts the responsibility for TD ticket repayment to the business analysts and therefore project management. Thereby, this paper proposes a solution to the problem of "*Split of budget in Project budget and Maintenance budget boosts the accumulation of debt*" ([7]).

TD Awareness: One important effect of this framework is the raised awareness for contracting TD. Kruchten [16] already mentioned the importance of awareness: "*The first step is awareness: identifying debt and its causes*". In a survey with more than 1,800 participants, Ernst et al. [17] find that that 79% of the participants strongly agree that "*Lack of awareness of TD is a problem*". Tonin et al. [18] researched the effects of TD awareness by conducting a classroom study. In contrast to that this paper presents a method to raise the awareness not only for developers but also for business analysts and managers. Finally, this paper presents an evaluation in industry, which provides a higher external validity.

TD Management Case Studies: Other case studies concerning TD management are for example [19] and [20]. They present research findings in the same area as this study as they try to tackle impact of TD in industry. Yet, in [19] Martini et al. follow a more organizational solution to TD management. Guo et al. [20] focus their framework on the TD identification, TD measurement and TD monitoring. Furthermore, Guo et al. provide an idea for TD prioritization. It could be interesting future work to evaluate whether the frameworks can be inte-

grated to a more holistic approach.

VII. CONCLUSION AND FUTURE WORK

With this paper we present and evaluate a framework that includes TD management in project management by so-called TD tickets. TD tickets hand over the responsibility for TD contracted during a project to the project management of the same project. This creates a feedback loop of team and management behavior that leads to TD prevention. As it was developed and extensively used in practice, the framework shows a high external validity and provides the research community with and insight into the practical handling of TD.

The adoption of the framework led to a higher TD awareness and conscious contraction of TD in the case company. This on the other hand leads to the prevention of TD, more rational discussions between the unit members and more rational decisions. Furthermore, the TD items taken on during a project are part of the project and will be paid back timelier. While the idea of maintenance tickets and projects is not new, the idea of TD ticket is a novel approach that could not be found in research literature to the best of the author's knowledge.

For industry, this paper presents a framework that can be adopted, refined, and adapted to their own needs. The idea of including TD management in project management can be an incentive to develop a similar approach. Also, the importance of raising TD awareness and contracting TD intentionally by taking the time to compare different solution options was presented in this paper. The advantages shown by this paper can be helpful when negotiating options to handle TD with the IT management.

For researchers, this paper provides insights to procedures for managing TD developed in industry. The paper indicates the importance to provide a solution for handling and preventing TD even under the pressure of a tight timeline. The paper also confirms the problem that the responsibility for TD needs to be transferred to the whole unit and cannot be solved by developer teams alone. The framework presented in this paper provides a solution for these problems.

For future work, the framework could be complemented by other TD activities, e.g. TD prioritization, TD monitoring and TD identification, and their impact could be evaluated. Especially ideas to TD Prioritization may help to optimize the repayment phase as this was mentioned as a problem. The management of deconstruction tickets in particular is an interesting and promising research topic and should be evaluated in future. Timely repayment of TD is sometimes not performed due to low priority or missing awareness of competing goals. Further research with respect to project management and business value estimation is necessary. Finally, the prioritization of maintenance tasks including TD should be assessed between different application domains and industries.

ACKNOWLEDGMENT

We would like to thank *Gruner+Jahr GmbH - Media Sales Services* and - *Smart Business Solutions* for their contributions to the framework, for completing the questionnaire and for their support of this paper.

REFERENCES

[1] P. Avgeriou, P. Kruchten, I. Ozkaya, and C. Seaman, "Managing Technical Debt in Software Engineering," *Dagstuhl Reports*, vol. 6, no. 4, pp. 110–138, 2016.

[2] Z. Li, P. Avgeriou, and P. Liang, "A systematic mapping study on technical debt and its management," *Journal of Systems and Software*, vol. 101, pp. 193–220, mar 2015.

[3] J. Yli-Huumo, A. Maglyas, and K. Smolander, "How do software development teams manage technical debt? – An empirical study," *Journal of Systems and Software*, vol. 120, pp. 195–218, 2016.

[4] N. Rios, R. O. Spínola, M. G. De Mendonça Neto, and C. Seaman, "A study of factors that lead development teams to incur technical debt in software projects," in *Proceedings - 44th Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2018*, 2018, pp. 429–436.

[5] W. Cunningham, "The WyCash portfolio management system," in *Proceedings of the Conference on Object-Oriented Programming Systems, Languages, and Applications, OOPSLA*, vol. Part F1296, no. 2, 1992, pp. 29–30.

[6] R. Verdeccia, P. Kruchten, P. Lago, R. Verdecchia, P. Kruchten, and P. Lago, "Architectural Technical Debt : A Grounded Theory," in *European Conference on Software Architecture (ECSA 2020)*, A. Jansen, I. Malavolta, H. Muccini, I. Ozkaya, and O. Zimmermann, Eds. Springer, Cham., 2020, pp. 202–219.

[7] A. Martini, J. Bosch, and M. Chaudron, "Architecture technical debt: Understanding causes and a qualitative model," in *Proceedings - 40th Euromicro Conference Series on Software Engineering and Advanced Applications, SEAA 2014*, 2014, pp. 85–92.

[8] S. Freire, N. Rios, M. Mendonça, D. Falessi, C. Seaman, C. Izurieta, and R. O. Spínola, "Actions and impediments for technical debt prevention: Results from a global family of industrial surveys," in *Proceedings of the ACM Symposium on Applied Computing*, 2020, pp. 1548–1555.

[9] S. McConnell, "Managing technical debt," *Construx Inc.*, 2008. [Online]. Available: http://www.construx.com/uploadedfiles/resources/whitepapers/ManagingTechnicalDebt.pdf

[10] A. Joshi, S. Kale, S. Chandel, and D. Pal, "Likert Scale: Explored and Explained," *British Journal of Applied Science & Technology*, vol. 7, no. 4, pp. 396–403, 2015.

[11] H. B. Mann and D. R. Whitney, "On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other," *The Annals of Mathematical Statistics*, vol. 18, no. 1, pp. 50–60, 1947.

[12] J. Cohen, *Statistical Power Analysis fot the Behavioral Sciences*. New York, New York, USA: Lawrene Erlbaum Associates, 1988.

[13] K. Pearson, "I. Mathematical contributions to the theory of evolution. —VII. On the correlation of characters not quantitatively measurable," *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, vol. 195, no. 262-273, pp. 1–47, 1901.

[14] T. Besker, H. Ghanbari, A. Martini, and J. Bosch, "The influence of Technical Debt on software developer morale," *Journal of Systems and Software*, vol. 167, 2020.

[15] Z. Codabux, B. J. Williams, and N. Niu, "A Quality Assurance Approach to Technical Debt," in *Proceedings of the International Conference on Software Engineering Research and Practice (SERP)*, 2014.

[16] P. Kruchten, R. L. Nord, and I. Ozkaya, "Technical debt: From metaphor to theory and practice," *IEEE Software*, vol. 29, no. 6, pp. 18–21, nov 2012.

[17] N. A. Ernst, S. Bellomo, I. Ozkaya, R. L. Nord, and I. Gorton, "Measure it? Manage it? Ignore it? Software practitioners and technical debt," *10th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, ESEC/FSE 2015 - Proceedings*, pp. 50–60, 2015.

[18] G. S. Tonin, A. Goldman, C. Seaman, and D. Pina, "Effects of Technical Debt Awareness: A Classroom Study," in *18th International Conference, XP 2017*, H. Baumeister, H. Lichter, and M. Riebisch, Eds., 2017, pp. 84–100.

[19] A. Martini, L. Pareto, and J. Bosch, "Towards introducing agile architecting in large companies: The CAFFEA framework," in *Lecture Notes in Business Information Processing*, vol. 212, 2015, pp. 218–223.

[20] Y. Guo, R. O. Spínola, and C. Seaman, "Exploring the costs of technical debt management – a case study," *Empirical Software Engineering*, vol. 21, no. 1, pp. 159–182, 2016.

Chapter 11

Preventing Technical Debt with the TAP framework for Technical Debt Aware Management

2 *(TAP Framework – extended)*	
Title	Preventing Technical Debt with the TAP framework for Technical Debt Aware Management
Authors	Wiese M., Rachow P., Riebisch M., Schwarze J.
Year	2022
Journal	Information and Systems Technology, Volume 148
Reference	[Wie+22]

TABLE 11.1: Publication No. 2 *(TAP Framework – extended)*

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

Preventing technical debt with the TAP framework for Technical Debt Aware Management

Marion Wiese [a,*], Paula Rachow [a], Matthias Riebisch [a], Julian Schwarze [b]

[a] Universität Hamburg, Department of Informatics, Vogt-Kölln-Str.30, 22527 Hamburg, Germany
[b] Gruner + Jahr GmbH, Information Technology, Baumwall 11, 22459 Hamburg, Germany

ARTICLE INFO

Keywords:
Technical Debt
Project Management
Technical Debt Awareness
Technical Debt repayment
Technical Debt prevention
Technical Debt backlog

ABSTRACT

Context: Technical Debt (TD) is a metaphor for technical problems that are not visible to users and customers but hinder developers in their work, making future changes more difficult. TD is often incurred due to tight project deadlines and can make future changes more costly or impossible. Project Management usually focuses on customer benefits and pays less attention to their IT systems' internal quality. TD prevention should be preferred over TD repayment because subsequent refactoring and re-engineering are expensive.
Objective: This paper evaluates a framework focusing on both TD prevention and TD repayment in the context of agile-managed projects. The framework was developed and applied in an IT unit of a publishing house. The unique contribution of this framework is the integration of TD management into project management.
Method: The evaluation was performed as a comparative case study based on ticket statistics and two structured surveys. The surveys were conducted in the observed IT unit using the framework and a comparison unit not using the framework. The first survey targeted team members, the second one IT managers.
Results: The evaluation shows that in this IT unit the TAP framework led to a raised awareness for the incurrence of TD. Decisions to incur TD are intentional, and TD is repaid timelier. Unintentional TD incurred by unconscious decisions is prevented. Furthermore, better communication and better planning of the project pipeline can be observed.
Conclusion: We provide an insight into practitioners' ways to identify, monitor, prevent and repay TD. The presented framework includes a feasible method for TD prevention despite tight timelines by making TD repayment part of project management.

1. Introduction

Software systems are subject to frequent changes due to new or changing requirements or changes in the environment. Hence, a central goal in the development of software systems should be to implement these systems in such a way that they are easily changeable over a long time. Many different causes limit this ease of changeability and should be addressed. Because these modifications usually provide no visible benefit to non-developers and are challenging to understand, they are often given a lower priority by business leaders. Cunningham introduced the metaphor of Technical Debt (TD) to make this problem better understandable to business managers [1].

Avgeriou et al. define TD as "*a collection of design or implementation constructs that are expedient in the short term, but set up a technical context that can make future changes more costly or impossible*" [2]. In a technical metaphor for financial debt, a sub-optimal implementation or design is interpreted as debt. The resulting problems are interpreted as interest rates, refactoring as repayment, and refactoring cost as principal.

In 2016, Avgeriou et al. claimed about TD: "*It's the next big thing, and it's messy*" [3]. TD is a serious problem in practice, often caused by tight deadlines due to fast-changing requirements in the age of digitalization [2,4,5].

McConnell and Fowler differentiate between intentional and unintentional incurred TD [6,7]. Intentional TDs result from a conscious decision, e.g., implementing a workaround to reach a set deadline. Unintentional TDs result from mistakes, e.g., poor design at the code level or architectural decisions that, over time, turn out to have been wrong.

TD prevention is stated as the preferable option for TD management by many practitioners [8–10]. In a long-term view, implementing the optimal solution right from the beginning is cheaper than incurring TD,

* Corresponding author.
E-mail addresses: marion.wiese@uni-hamburg.de (M. Wiese), paula.rachow@uni-hamburg.de (P. Rachow), matthias.riebisch@uni-hamburg.de (M. Riebisch), schwarze.julian@guj.de (J. Schwarze).

https://doi.org/10.1016/j.infsof.2022.106926
Received 14 September 2021; Received in revised form 12 April 2022; Accepted 15 April 2022

especially when considering possible interest payments [4,11]. TD prevention primarily targets unintentional TD. To enable TD prevention, two main aspects must be considered:

1. Stakeholders have to be aware of TD to make conscious decisions [12–14].
2. Stakeholders have to understand the causes of TD [4,15].

TD Awareness. Awareness is self-evident for self-admitted TD, i.e., ToDo-Statements in the code, and TD found by tools. In practice, TD prevention is often focused on these kinds of TD [10,16]. In research, the perception of design and architecture decisions and the TD incurred by these is less evident and less focused on.

TD backlogs and lists lead to more visibility, make TD explicit, and raise the perception of TD. This, in turn, raises the overall awareness for TD [17–20].

The awareness leads to better communication between developers and IT management, which is a problem often mentioned regarding TD management [3,15,21,22]. For example, in their work about architectural TD (ATD), Verdeccia et al. found "*difficulties in communicating the presence of ATD to the stakeholders of a software product*" [5].

Better communication and the overview of TD can, in turn, impact how IT managers plan the project pipeline, i.e., the timing of the project portfolio's various projects. This could provide a solution to the impediment mentioned by Besker et al. [23]: "*If the managers are not aware of the amount of software development time the developers waste because of TD, they are consequently not able to react and take appropriate action regarding the wasted time.*"

TD Causes. The initially named cause for TD incurrence is tight project deadlines, as described in the paper of Cunningham [1]. Other causes have been identified by different research papers, e.g., bad design decisions, unavailability of a key person, neglected technical improvements, lack of education, communication issues, or problems related to project planning [4,17,24,25]. Nevertheless, tight deadlines are identified by many research works as the most pressing cause [4,5,15,21,24,26].

An example of the problem of tight deadlines is the German government's VAT cut in the summer of 2020 as part of its response to the Corona crisis. The corresponding law was enacted only two days before it came into force. To reflect this change in the software systems, adjustments to these systems were necessary for many German IT units. These units were, therefore, subject to a tight schedule.

In many cases, this constant time pressure leads to a phenomenon in developer behavior. In preemptive obedience, the developers search for the fastest and easiest solution assuming a tight schedule, even when there is no time pressure. To step back and think about or discuss different solution options is not common practice. In a paper about cognitive biases and their antecedents, Borowa et al. call this phenomenon communicational antecedents, i.e., "*constructive criticism and challenging the ideas underlying the decisions of others is not standard*" [22]. This may lead to the anchoring bias, i.e., choosing the first available option. In turn, this bias causes decisions to incur TDs that are not made intentionally. Regarding this topic, Besker et al. found that "*practitioners rarely base their decisions on pre-defined procedures or guidelines*" [20], and Lenarduzzi et al. state that "*decisions related to TD issues were often informal and ad-hoc*" [27].

There may be frequent situations where the sub-optimal but faster to implement solution must be chosen due to project deadlines. However, it should be a goal to make it the whole team's conscious and intentional decision to choose the optimal or sub-optimal solution.

In this paper, we present and evaluate the TAP framework for **T**echnical debt **A**ware **P**roject management which was developed by practitioners. The TAP framework divides all tasks that are not visible to the customer into four categories. The developers create corresponding backlog tickets. One of these categories is the so-called TD ticket category. These tickets comprise only intentional TD and relate these tickets to the project incurring the TD. The repayment of these TD items is performed as part of the project after meeting the set deadline. Other parts of the TAP framework are a continuous time contingent for maintenance and maintenance projects similar to what McConnell described [6]. Maintenance in the TAP framework includes repayment of unintentional TD, adaption to new technologies, optimization of quality attributes. This way of handling TD is a new contribution to the State of the Art.

To the best of the authors' notice, no studies integrate TD management into the project management process focusing on the time limit and including a solution to prevent TD despite tight deadlines that is feasible in an industrial environment. By TD tickets and their effects on decision-making and the project pipeline, this framework presents such a solution.

The contribution of this paper is the presentation and evaluation of the TAP framework and especially the TD tickets. The evaluation consists of the analysis of ticket statistics and two surveys addressing team members and IT managers, respectively.

The evaluation will show the TAP framework's benefits as perceived by the IT unit using it:

- Communication between different stakeholders is optimized.
- Overall awareness for TD is raised.
- Decisions on TD incurrence are made consciously and intentionally.
- TD incurred by unintentional and unconscious decisions can be prevented.
- TD incurred by intentional and conscious decisions due to tight deadlines are repaid timely.

Additionally, a survey of the IT managers indicates that they can plan the project pipeline more realistically and make the customer understand delays due to TD repayment and prevention.

In the following subsection, we will give an overview of the related work. Subsequently, we will motivate and introduce the TAP framework in Section 3. Section 4 will present the case study design, including research questions (RQs) regarding the TAP framework's effects, benefits, and drawbacks. The evaluation comprises ticket statistics and results of two surveys conducted in the observed IT unit and a comparison unit. The evaluation's results will be presented in Section 5. In Section 6, we will discuss the results, answer the RQs, and describe the threats to validity. The paper ends with the conclusion and future work in Section 7.

2. Related work

In this section, we present the related work to our paper. For this purpose, we follow the paper's title and describe the state of the art of TD prevention, TD awareness, TD in project management, and case studies for TD management frameworks or methods.

2.1. TD prevention

TD Prevention is often mentioned in research but mostly as a **problem statement** [8,28,29]. In 2015, Li et al. [25] found TD prevention to be understudied. More research on how to prevent TD beyond raising TD awareness has been done since.

Many papers researching **prevention strategies** focus on inadvertent code debt and tools helping to find these debts, e.g., code smells, before incurrence [8,10,11]. Fewer papers focus on the prevention of TD on the design and architecture level, which are usually incurred during decision-making.

Morgenthaler et al. provided insights to the TD management of build debts at Google. They summarized their prevention strategies in three topics "Automation", "Make it easy to do the right thing", and "Make it hard to do the wrong thing" [12]. Yli-Huumo et al. found that minimum viable products can help prevent TD by focusing TD prevention on areas that are important for the customer and throwing away

parts that have shown no real business value [30]. Preventing documentation debt during requirements engineering was the research focus of Charalampidou et al. They integrated requirements specifications into the IDE (integrated development environment) [31]. Lenarduzzi et al. suggested optimizing requirements and receiving early feedback from customers. These approaches improved the effort estimation and enhanced the quality by optimizing tests and documentation [27]. Yang et al. suggested for COTS (commercial of the shelf) - intensive systems to train the COTS assessment team and explore contracting options with COTS vendors [32]. Another way to prevent TD is to reuse code which was mentioned in the work of Feitosa et al. [33]. In their comparison of encouraging, rewarding, forcing, and penalizing strategies for TD reduction, Besker et al. found some companies were using educational sessions on how to avoid TD [13]. Vogel-Heuser et al. advised using guidelines to prevent TD in embedded systems [29]. Borowa et al. presented some mechanisms to prevent cognitive biases that may lead to TD incurrence, such as "challenging all decisions", "explicitly gathering information about alternatives", "documenting and passing on knowledge", "explicitly registering all accounts of TD" and "defining and recording who is responsible" [22].

The InsighTD project[1] is an international survey study. The survey participants often stated that their TD could have been prevented [4]. Various papers of this study uncovered TD causes and prevention strategies used by practitioners. The most commonly found prevention strategies are "well-defined scope/requirements", "code evaluation/standardization", "following well-defined project processes", "well-defined architecture/design", "TD awareness/management", "adoption of good practices", "improving tests/coverage", "good communication", and "reviews" [4,9,14,28].

All these research results are valid aspects of TD prevention. However, compared to the **TAP framework** these aspects are mostly vague, i.e., details and information on how to achieve these aspects are missing. Furthermore, they do not provide a solution to the problem of tight deadlines.

2.2. TD awareness

In 2012, Kruchten et al. already mentioned the importance of TD awareness: "*The first step is awareness: identifying debt and its causes*" [17]. In a survey with more than 1800 participants, Ernst et al. found that 79% of the participants strongly agree that "*Lack of awareness of TD is a problem*" [26].

As with prevention, there are many papers mentioning TD awareness in the form of a **problem statement** [10,12,29,34].

Papers on **how to raise awareness** are less common. Many research papers found that visualization helps to raise awareness for TD. An obvious way to visualize TD is to create a list of all known TD items, which is also part of the TAP framework. For example, Kruchten et al. provided a color scheme for a backlog including colors for new features, architectural features, defects, and TD [17]. Kaiser et al. presented the Code Christmas Tree, which is a visual presentation of the code coverage of a system. This visualization was hung in the hallway and resulted in discussions. By this, the visualization raised the awareness of different stakeholders [35]. Gupta et al. also found that so-called information radiators, i.e., visualizations that are displayed where people can see it frequently, were a good way to raise awareness [36]. Baysal et al. as well as Treude et al. examined the positive effects dashboards have on developers' awareness of their projects' issues [37,38]. Guo et al. presented a TD management approach including a list of TD, which increased the awareness of TD. In this case study, the project manager stressed the TD list's positive effect [19]. Eliasson et al. proposed a method to visualize dependencies in the automotive domain to make TD visible. They found their method to be useful for stakeholders to be aware of architectural TD [18]. Martini et al. proposed a method called AnaConDebt to identify whether and when Architecture TD should be refactored. They found this method is useful to raise awareness "*about the importance and the urgency of proactively refactoring ATD*" [39]. Baars et al. used a gamification approach to make developers aware of harmful patterns in their codebase by turning the patterns into monsters in Minecraft[2] [40]. Al Mamun et al. researched TD in self-driving miniature cars. They focused on code debt and proposed that tools detecting rules violations would also increase awareness [41].

Another way to raise awareness is to optimize communication, e.g., workshops and training sessions. The communication aspect is important for the TAP framework, too. Shrama et al. found the awareness of the refactoring's benefits was lacking. They reported on the strategy to conduct workshops on the refactoring's benefits and refactoring tools [34]. Letouzey et al. also mentioned an awareness workshop for top managers to introduce TD and the SQUALE[3] method to them [42]. Tonin et al. [43] researched the effects of TD awareness by conducting a classroom study. The students were informed about the concept of TD and should use TD boards during a project. The effects of this were a changed attitude, more discussions, and more conscious incurrence of TD.

In addition to the presented ways to raise TD awareness, the **TAP framework** raises the awareness by assigning responsibilities for TD management which is an aspect not found in research to the best of the authors' notice.

2.3. TD management and project management

A project is characterized by a beginning and an end and thus has a deadline. This deadline is an often mentioned cause for TD, as shown in Section 1. One often mentioned **problem** for managing and preventing TD is related to project management [4,9,14,19]. "*Lack of process integration*" is an obstacle mentioned in the work of Guo et al. Their case study provided some insights into the obstacles of adapting research methods to a practical environment [44]. Martini et al. found that the "*split of budget in project budget and maintenance budget boosts the accumulation of debt*" [24].

To **solve these problems** Ramasubbu et al. integrated TD management into quality management steps [45]. They created a TD register to store TD items in relation to their requirements and resulting defects. Furthermore, they considered design moves to implement requirements or repay TD and the risks associated with them. They conducted a cost–benefit analysis on this data and proposed a way to control TD. While their approach is similar to the TAP framework, their focus was on companies that already use quality assurance processes, and their approach did not focus on TD prevention. Other research papers address the management of a TD backlog, e.g. [17,46,47].

In contrast to the **TAP framework**, these papers do not focus on projects and the resulting timeline problems. The TAP framework tackles the deadline problem by integrating TD management into project management, i.e., integrating the repayment of a project's TD as part of the project itself.

2.4. Case studies on TD management

Regarding TD management, Besker et al. found that it "*promotes a culture within organizations in which developers are supported and appreciated for managing their TD*" and that it "*leads to a virtuous cycle where the right culture and TD prevention mechanisms reinforce each other*" [48].

This paper presents a comparative case study of a TD management framework developed by practitioners. To the best of the authors' notice, no other case studies compare an IT unit using a framework

[1] http://www.td-survey.com/.

[2] https://www.minecraft.net/.

[3] http://www.squale.org/.

or method for managing TD with an IT unit not using a TD management strategy. For this reason, we compare our work to other case studies on **TD management frameworks and methods** but focus on confirmatory case studies and action researches.

In an action research, Yli-Huumo et al. developed a process for identification, documentation, and prioritization of TD. They focused on uncovering the risks and benefits related to these TD activities [49]. Guo et al. focused their case study on uncovering the costs of their TD management approach, which centers around a TD list and comprises five steps for measuring and prioritizing TD items. During the implementation of this approach in practice, they had to face various obstacles, which they presented in another paper [44]. Finally, they applied their approach not directly in practice but simulated it for past releases on actual company data. The Tracy framework for business-driven TD prioritization was developed by Reboucas de Almeida et al. They proposed this framework to prioritize TD repayment activities not only by technical necessities but also by the business value of the configuration item that holds the TD item [50]. The work of Ramasubbu et al. is similar to the TAP framework and was also validated by an industrial case study [45]. We already introduced it in the previous paragraph. Malakuti et al. described the TD management approach of a specific project and obstacles while introducing TD concepts to a company, but they were still missing a working solution [51].

All these TD management strategies focus on the general management of TD without a relation to a specific time frame. In contrast, the central aspect of the **TAP framework**'s TD tickets is the integration of TD management into project management which includes the deadline aspect.

3. Tap framework

3.1. Motivation

The TAP framework was developed and established in an IT unit at the beginning of 2018. It was utilized ever since and is the subject of this case study. The focus of the IT unit that developed this framework is on systems that support the marketing of advertisements in print, online and mobile media, a volatile and competitive market.

Accordingly, tight timelines and frequently changing systems are the main cause for the TD of this IT unit. The tight timelines often meant that "*well-structured project planning*" or "*adopting good development practices*" as proposed by Freire et al. [28] is not sufficient.

In this IT unit, TD accumulation led to increasing cycle times, unnecessary errors in the systems, and developers' discontent. The IT management was surprised by these problems because they were not aware of having incurred debts in any form. Ultimately, however, they recognized these problems and gave sufficient priority to dealing with TD. The IT unit including the management was willing to change its processes.

The main goals for this change were pre-defined by the IT unit's manager: First, TD must be prevented in any case if this is possible without affecting the deadlines. Second, whenever a set deadline leads to TD, this TD shall be repaid timely. Third, the IT managers should have the ability to track the TD in their systems.

On this basis, a framework to help manage TD was established. Two factors of the TAP framework, in particular, were intended to support TD prevention: the overall raised awareness for TD incurrence and the integration of TD management into the project management process. A third factor, the overview of TD resulting from the framework adoption, helped the management track TD. The resulting TAP framework for TD management is presented below.

Table 1
Hindering tasks and their assigned ticket categories (MT — Maintenance Ticket, MP — Maintenance Project, TDT — TD Tickets, DT — Deconstruction Tickets, FRT — Functional Requirement Ticket).

Task	Scope	Category
(1) Bug fixing	Not visible/	MT
	visible	FRT
(2) Refactoring	Small	MT
	Big	MP
(3) Workaround		TDT
(4) Documentation/Test		TDT
(5) Incomplete implementation		TDT
(6) Temporary switch		DT
(7) Architecture change	Small	MT
	Big	MP
(8) Infrastructure		MP
(9) POCs	Small	MT
	Big	MP
(10) Monitoring		MT
(11) Performance	Not visible	MT
	Visible	FRT

3.2. Development

The framework was developed in a working group of two managers and two solution architects of the IT unit. The development spanned five steps: (I) identification of the tasks that hinder the development of functional requirements, (II) identification of management and prevention processes for these tasks, i.e., ticket categories, (III) assignment of tasks to ticket categories, (IV) development and documentation of the overall framework, (V) establishment of the new processes in the IT unit.

Eleven hindering tasks were identified: (1) bug fixing, (2) regularly refactoring of code that may be in a bad condition due to causes that are not specified in detail, (3) workarounds which are made under the pressure of a deadline, (4) missing documentation or tests mostly due to time pressure, (5) implementation that stay incomplete because of changing business decisions, (6) temporary switches in the program for the distinction of new and old program parts, (7) adoption of new architecture approaches, (8) adapting new infrastructure like e.g. version upgrades, (9) innovation like proofs of concepts (POC) for new technologies, (10) implementation and refinement of technical monitoring tools, (11) performance optimization.

Four processing types were developed to handle these tasks, resulting in the corresponding ticket categories. In this framework, all other tasks are functional requirements including bugs and performance issues that are visible to the user. The assignment of the hindering tasks to these categories can be seen in Table 1.

3.3. Ticket categories

The TAP framework results in four ticket categories for handling different types of TD-related tasks. It is essential to notice that the distinction of these categories is not because of the causes or theoretical classification schemes. The distinction refers to the way the tickets are to be processed. A fifth ticket category handles all functional requirements. The latter category is not presented in detail as it is not in the scope of this framework.

All tasks are recorded as tickets in a comprehensive project backlog and tagged and handled depending on their category. The framework includes guidelines for the category-specific recording and processing of these tickets. The person who has the greatest interest in completing the tasks of a category is the one responsible for this category. This person may be an architect or a business analyst.

Table 2 shows an overview of the ticket categories, the responsible person, and the processing conditions.

Table 2
Ticket categories, responsible persons, and time.

Ticket category	Responsible person	Recorded by	Processing time	Contingent
Maintenance	Architect	Developer	Continually/architect decision	Maintenance (10%)
Maintenance project	Architect	Architect	Management decision	Project
Technical Debt	Business analyst	All	After deadline, part of project	Project
Deconstruction	Architect	All	As soon as possible	Project/functional req.
Functional Req.	Business analyst	Business analyst	Business analyst decision	Project/functional req.

3.3.1. Maintenance

In the TAP framework, maintenance tickets are technically driven tickets that have no direct impact on the user's perception of the system.[4] In particular, adapting systems technically, enhancing quality attributes, and repaying unintentional TD are parts of this category.

Recording: Every maintenance ticket must contain one sentence describing the ticket's impact. The description must be understandable to business analysts and managers.

Processing: Ten percent of every sprint's planned capacity is invested in maintenance tickets according to the architect's decision.

Goal: The goal of the maintenance tickets is to repay the unintentionally incurred TD continually and to allocate time for other maintenance tasks, e.g., technical adaption.

Example: Examples of this category are the upgrade of a third-party library or the refactoring of poorly structured code.

3.3.2. Maintenance project

All tasks belonging to the maintenance category but requiring more than five days of development time are treated as maintenance projects. This is based on the practice applied for functional requirements and business projects.

Recording: The architects prioritize maintenance projects and create a pipeline of these projects.

Processing: This pipeline is integrated into the overall project pipeline as defined and prioritized by the IT managers.

Goal: Maintenance projects aim to allocate enough time to perform maintenance tasks with a long duration or maintenance tasks requiring more organization, e.g., when more than one team is involved.

Example: An example is a version upgrade of a central database used by more than one application. Another example is the change of the system's architecture, e.g., from a SOA to a microservice architecture.

3.3.3. Technical debt

These so-called TD tickets only comprise intentional TD. These tickets describe tasks necessary to improve internal software quality during the implementation of a functional requirement, e.g., time for clean code, good design, or to comply with a specified architecture. These tasks are not mandatory to implement the required functionality and can be implemented after a given deadline.

The team often discusses two or more solution options, one of which follows the standards and the specified architecture, and the others more or less deviate from the optimal solution. The latter, however, may be faster to implement. Hence, this category only refers to conscious decisions to incur TD, i.e., intentional TD.

This idea of TD follows the original introduction of TD by Cunningham [1].

Recording: In such cases, the team consciously decides to record two tickets: The functional requirement ticket describes the sub-optimal solution implemented before the deadline. With this ticket, the team incurs TD. The TD ticket describes the optimal solution and the tasks to repay the incurred TD. TD tickets are usually identified during estimation meetings when the team discusses the details of a functional requirement and estimates the effort for this requirement.

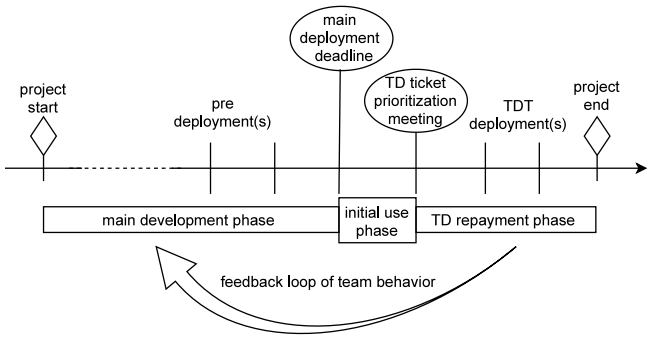


Fig. 1. Project plan with included TD repayment phase.

Processing: The uniqueness of the TAP framework is that the repayment of these intentionally incurred TD items is part of the project plan. No project can be finished before the TD tickets that have been incurred during this project are processed. The project plan does not end after reaching the deadline and keeping to the tight schedule. Instead, the development team and architects prioritize the TD tickets of this project in a prioritization meeting. The prioritization is discussed and decided by a majority vote and does not follow a strict procedure. In a subsequent project phase, the TD repayment phase, the developers process the project's TD tickets alternately with the project's remaining requirement tickets, as shown in Fig. 1. Thus, the responsibility for the TD repayment is transferred to project management because to finish their project is in the project manager's interest. In this way, the TAP framework integrates the management of intentional TD into the project management process.

Goal: First, evaluating different solution options in advance raises the overall awareness only to incur TD intentionally. Consequently, unintentionally incurred TD is prevented. Second, making the project managers responsible for the TD accumulated during their project decreases their willingness to incur TD. In other words, this creates a feedback loop, and both the team and the project manager change their behavior towards TD incurrence. Third, project managers and IT managers get an overview of accumulating TD of a particular project while still running. The IT managers can intervene or adjust the project plan and the following projects early on. Finally, by this approach, intentionally incurred TD is repaid timely.

Example: For the example from Section 1, this may mean that hard-coded VAT rates are adjusted in the code rather than refactored to a flexible solution to meet the schedule. As a result of this decision, the developers record the corresponding TD ticket to introduce a centralized tax variable. This TD will be repaid in an orderly manner in the "TD repayment phase" of the "VAT adjustment" project and after the law comes into force. In contrast, discussion of the various solution options may show that the low effort makes centralization the better option, and TD are prevented from the beginning.

3.3.4. Deconstruction

A ticket of the deconstruction category is a special TD ticket that cannot be processed during the project. After implementing a new

[4] We use the term "maintenance" as it is often used in agile organizations. However, in research, standardization, and sometimes in classical organizations, the implementation of new or changing functional requirements is also referred to as adaptive maintenance.

solution, the legacy code sometimes needs to be kept parallel for the time being. These tickets then comprise the deconstruction of this legacy code when it is no longer needed. They can also comprise the deletion of deprecated code parts after all consumers of these parts are detached. Therefore, this category only comprises intentional TD that cannot be repaid immediately, i.e., during the TD repayment phase of its project.

Recording: These tickets must contain additional information about when the deconstruction can take place.

Processing: The deconstruction ticket must be repaid as soon as possible as part of the contingent for functional requirements.

Goal: The primary purpose of these tickets is to avoid code cluttering and improve code comprehension.

Example: An example is a need for a new business rule for the following business year. The application should not be deployed on New Year's Eve to minimize deployment risks. Hence, the application will be deployed in advance and must contain a date-driven switch to choose between the old and new business rules. When the old business rule is no longer needed, it can be deconstructed.

3.4. Scope

The initial scope of this framework is the IT unit that developed the framework. Nevertheless, other units and companies may adopt the framework or at least relevant parts of it. Relevant parts are, in particular, the distinction in processing intentional TD as TD tickets and unintentional TD as part of maintenance tickets, the immediate recording of intentional TD, and the processing of intentional TD as part of the project that incurred them.

We think this framework will particularly help units that are subject to tremendous time pressure. The framework was used in an organizational setting of agile project management, including agile processes and project management elements, including project deadlines. Regardless, we expect the framework to help in all organizational environments where a timeline is relevant, i.e., project or release deadlines.

The framework may not be suitable for agile product development without timeline constraints. Additionally, the problem caused by accumulated TD must be large enough to make the initial effort to organize the backlog worthwhile.

4. Case study design

The goal of this research was the presentation and evaluation of the TAP framework. The framework's presentation took place in Section 3. The evaluation was based on the case study design, according to Runeson [52].

4.1. Research questions

As described in Section 1, the two main problems concerning TD are preventing TD and dealing with TD despite having a tight schedule and fixed deadlines. The TAP framework provides a possible solution found and developed in practice for these problems. With subsequent research questions (RQ), this paper provided an evaluation of the TAP framework. We evaluated especially the framework's feasibility and effectiveness regarding the problems mentioned above.

(RQ 1) *Framework Application*

> ***(RQ 1.1)*** *Do practitioners find the TAP framework reasonable?*
> ***(RQ 1.2)*** *Are the processes of the TAP framework feasible in practice?*

This is the basis for all other evaluations as it is not reasonable to analyze the framework's impact further should it not be feasible in practice. We used a team survey of the observed IT unit's members and ticket statistics to evaluate these questions.

(RQ 2) *TAP Framework's perceived Effects*

> ***(RQ 2.1)*** *Is the use of the TAP framework associated with a raised awareness for the incurrence of TD?*
> ***(RQ 2.2)*** *Is the use of the TAP framework associated with a more conscious incurrence of TD items?*
> ***(RQ 2.3)*** *Is the use of the TAP framework associated with a better overview of TD items?*
> ***(RQ 2.4)*** *Is the use of the TAP framework associated with a timelier repayment of TD items?*

These aspects are the TAP framework's goals, as presented in Section 3. The evaluation of these questions shall show whether these goals were reached. To evaluate these questions, we used ticket statistics and surveyed the members of the observed and a comparison unit.

(RQ 3) *TAP Framework's perceived Benefits and Drawbacks*

> ***(RQ 3.1)*** *Do practitioners observe that TD can be prevented by the adoption of the TAP framework?*
> ***(RQ 3.2)*** *Are there other benefits or drawbacks arising from the adoption of the TAP framework?*
> ***(RQ 3.3)*** *Do practitioners find these benefits justify the additional effort?*

The purpose of these questions is to gather the TAP framework's impact. The main goal of TD prevention is evaluated, as well as secondary benefits concerning team discussion, decision-making process, and generating an overview of TD and maintenance tasks. To answer these RQs, we used descriptive statistics, and correlations between the survey variables for effectiveness and benefits to support the descriptive statistics. Furthermore, we asked for the justification of the additional effort in relation to the benefits to substantiate the framework's feasibility.

(RQ 4) *Management Perspective*

> ***(RQ 4.1)*** *Do the managers have an overview of their IT systems' TD?*
> ***(RQ 4.2)*** *What are the benefits of the TD overview from a management perspective?*

The success of the TAP framework depends largely on the management's support as the additional effort must be approved by them. Therefore, we evaluated the effect the framework and especially the generated TD overview has on the management.

4.2. Case and units of analysis

Following the definitions of Runeson [52], we conducted a comparative case study with two units of analysis — the observed IT unit that developed and adopted the TAP framework and the comparison unit that did not. The composition of the observed unit and the comparison unit are presented in Table 3. The comparison unit is slightly smaller but is led by the same unit manager and follows a similar organizational structure.

The focus of this case study was the evaluation of the framework's effects on the observed IT unit. The case study protocol, as well as all data collection and evaluation details and the data itself can be accessed in the additional material.[5]

Context. The IT units are part of a German publishing house with more than 9000 employees worldwide. This is an established company publishing content and advertisements in magazines, websites, and mobile apps. Both IT units follow a Scrum approach and develop and maintain software for internal use. They are both dealing with a substantial amount of legacy code.

Unit of analysis. The observed unit develops and evolves systems responsible for the marketing of advertisements. The main competitors are Google and Facebook. This leads to often changing systems due to new advertising formats and the merging of companies to get a bigger market share. Additionally, the changes are usually time-sensitive to

[5] https://doi.org/10.5281/zenodo.5788222.

be first in the market or adhere to the business year when merging companies.

The unit develops about 5–10 medium–large systems and the interfaces between these systems. The development tasks are organized in agile-managed projects. The unit comprises three organizational teams with two cross-cutting Scrum teams. The three organizational teams comprise:

- Business Analysis: The members are responsible for the requirements engineering and project management.
- Development: The members focus solely on the systems' development.
- Operations and Support: The members are responsible for customer communication, quality assurance, and basic server support.

Each Scrum team is responsible for specific tasks depending on the required technology's expertise. The business analysts are acting as product owners in the Scrum processes.

The unit of analysis was not chosen but given, as this IT unit developed the framework for itself.

Comparison Unit of Analysis. The comparison unit develops and evolves systems responsible for internal company tasks, like support for human resources, janitor services, company intranet, or canteen.

The unit develops about 20 small- to medium-size systems and their interfaces. Like the observed unit, the comparison unit uses agile projects for their development and has to reach project deadlines. On the one hand, in some of their projects, the time pressure is less stringent than in the observed unit. On the other hand, some of their projects have to adhere to legal requirements, including hard-set deadlines.

Furthermore, the IT management particularly encourages this unit to explore new technologies, leading to additional risk of TD and legacy code. The comparison unit consists of exactly one team, which is both the organizational team and the Scrum team. This team has one business analyst who is also the architect and the product owner in the Scrum process.

The comparison unit of analysis was chosen due to their availability as the manager of both units supports this study. Furthermore, both units follow a similar organizational approach, i.e., agile-managed projects.

4.3. Data collection

To answer RQ 1.2 and RQ 2.4, we evaluated the recording and processing statistics for TD and maintenance tickets. RQs 1 to 3 were answered by the results of the team survey. Finally, a follow-up survey investigated the TAP framework's effects on the IT managers, which answered RQs 4. Both surveys targeted participants from the observed unit and the comparison unit.

4.3.1. Ticket statistics

We collected data from all project backlogs of the observed unit and identified the tickets under consideration by tags and epic affiliation. All tickets in the backlog were labeled according to their category at the time of data collection. 524 maintenance tickets and 141 TD tickets were identified.

In consultation with the teams' architects, we validated the raw data and removed old or invalid tickets. Furthermore, we narrowed down the tickets to the period from January 2018 to March 2020. 236 maintenance tickets and 102 TD tickets remained for the evaluation.

We did not evaluate the statistics for maintenance projects and deconstruction tickets due to insufficient data, e.g., only ten deconstruction tickets exist between 2018 and 2021.

Table 3
Composition of units and team survey participants.

	Observed unit		Comparison unit	
	Members	particip.	Members	particip.
Manager	2	n/a	2	n/a
Architect	2	n/a	1[a]	n/a
Business analyst	8	6	1[a]	1
Developer	15	9	5	4
Operations	5	2	0	0
Sum	32	17	8	5
Response rate		53%		62%

[a]Architect and business analyst are the same person.

4.3.2. Team survey

We chose the method of a survey to benefit from the input of all team members willing to share their experiences. For RQ's 1 to 3, we asked corresponding questions in this questionnaire.

Survey Participants. We asked all members of the observed and comparison IT unit to fill out the questionnaire. The comparison unit's results allowed us to validate the descriptive statistics. The participants were asked which team they belong to, but due to works council regulations, they were not asked which role (e.g., architect, manager) they inhibit. The response rate to the survey is shown in Table 3.

Questionnaire Construction. To avoid misunderstandings, the questionnaire started with short information about the term TD. For statistical reasons, background information was queried, e.g., the participant's team membership. The questionnaire was divided into two main parts: (I) the framework's assessment and (II) the framework's effects and benefits.

Part (I) showed the framework's practical feasibility and provided answers to RQ 1.1 and RQ 1.2. For every ticket category, the same set of questions regarding the reasonableness and feasibility was asked. Each part started with short information about the ticket category. Part (I) was not filled out by the comparison unit as these participants did not use the framework.

Part (II) comprised five subsections with questions regarding:

- TD awareness (RQ 2.1)
- comparison of optimal and sub-optimal solutions (RQ 2.2)
- overview of TD (RQ 2.3)
- observed benefits of the comparison and overview (RQ 2.4, RQ 3.1 and RQ 3.2)
- justification of the additional effort (RQ 3.3)

All parts included a set of assertions the participants were asked to validate using the following Likert scale [53]: applies – rather applies – rather does not apply – does not apply – cannot answer. The *cannot answer* option was given as some questions could not be answered by all members of the unit. The participants were explicitly asked only to use this option in these cases.

Finally, two more open questions at the end allowed the participants to point out good parts of the TAP framework and parts needing improvement.

Two developers familiar with the TAP framework filled out a first questionnaire as a pilot test. Based on this, we reduced and focused the questions. One IT manager gave additional feedback on the optimized questionnaire's construction, which we incorporated.

4.3.3. Follow-up management survey

During a first presentation of the evaluation results, the management mentioned that the framework provides a valuable overview of the TD and maintenance tasks. This perception contradicts what most of the survey participants assumed of their management (see Fig. 4(b)). Since the advantages of TD management for IT managers are not often researched, we investigated this in more detail.

Survey Participants. The survey participants were only the three involved line managers: the unit manager of both units, the team manager of two teams of the observed unit, and the team manager of the comparison unit. All managers have personnel responsibilities for the teams involved and have a good and broad overview of the business goals. Due to the small sample size, we could not generalize the results. However, the evaluation provided qualitative input exploitable in further research.

Questionnaire Construction. We asked the managers four open questions via email to gather qualitative data. The questions were related to

- the overview they have of their system's TD,
- the effects this has
  - on their project pipeline,
  - on TD repayment and prevention, and
  - on the communication with their customers.

4.4. Data analysis

4.4.1. Ticket statistics

To evaluate RQ 1.2 and RQ 2.4, we presented the number of tickets in terms of categories and time as descriptive statistics for the period from January 2018 to March 2020.

4.4.2. Team survey

For the questionnaire's evaluation, we used the statistics software SPSS[6] for closed questions and the qualitative research software MaxQDA[7] for open questions.

First, we created descriptive statistics of all closed questions for an exploratory data analysis. For simplification and analysis purposes, we dichotomized the answers for the questions of part (II). This means we summarized the answers *applies* and *rather applies* as well as *rather does not apply* and *does not apply*. We present all values as a percentage to align the output of the observed unit and the smaller comparison unit. For the sake of clarity, we only show the *applies* answers in the following figures.

Second, we developed hypotheses for RQ 2 based on this exploration and evaluated the significance of the hypotheses. We used the Mann–Whitney U-Test [54] because it can be applied to not normally distributed data on an ordinal scale. We present only significant findings.

Third, we analyzed correlations between survey participants who incur TD consciously and the benefits they assess. We used the ϕ-coefficient of Pearson's $\tilde{\chi}^2$-correlation [55,56] for dichotomous variables and assumed a significant correlation from a significance level of 0.05, i.e., 5%. The effect size is interpreted as medium if it is higher than at 0.3 and as strong if higher than 0.6.

Last, two researchers used open coding to evaluate the open questions.

4.4.3. Follow-up management survey

Two researchers evaluated the management questionnaire's results using open coding with MaxQDA. Due to the limited number of participants, we did not count codes. Instead, we provided some qualitative insights of the line managers of the observed and comparison unit.

[6] https://www.ibm.com/dede/analytics/spss-statistics-software.
[7] https://www.maxqda.de/.

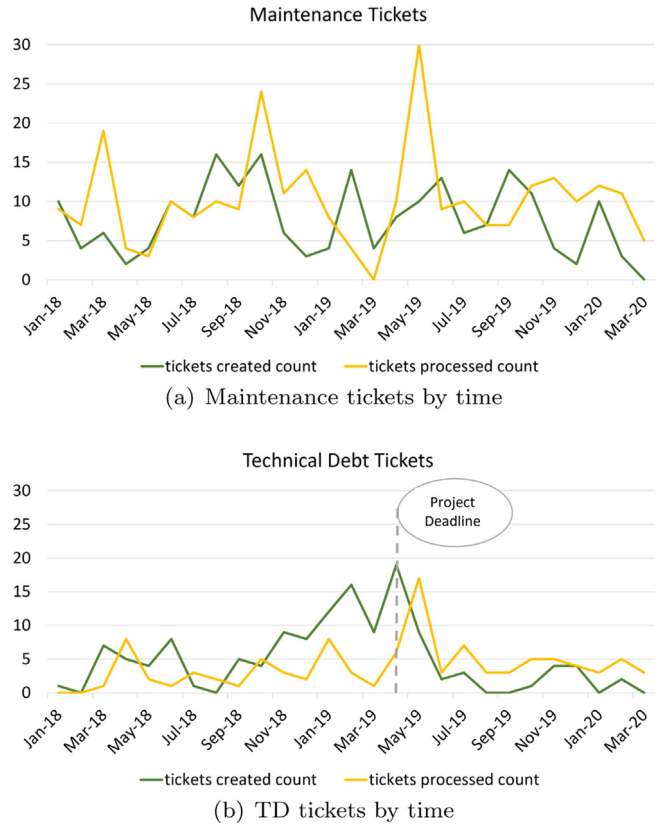


Fig. 2. Ticket statistics with a project deadline at the end of April 2019.

5. Results

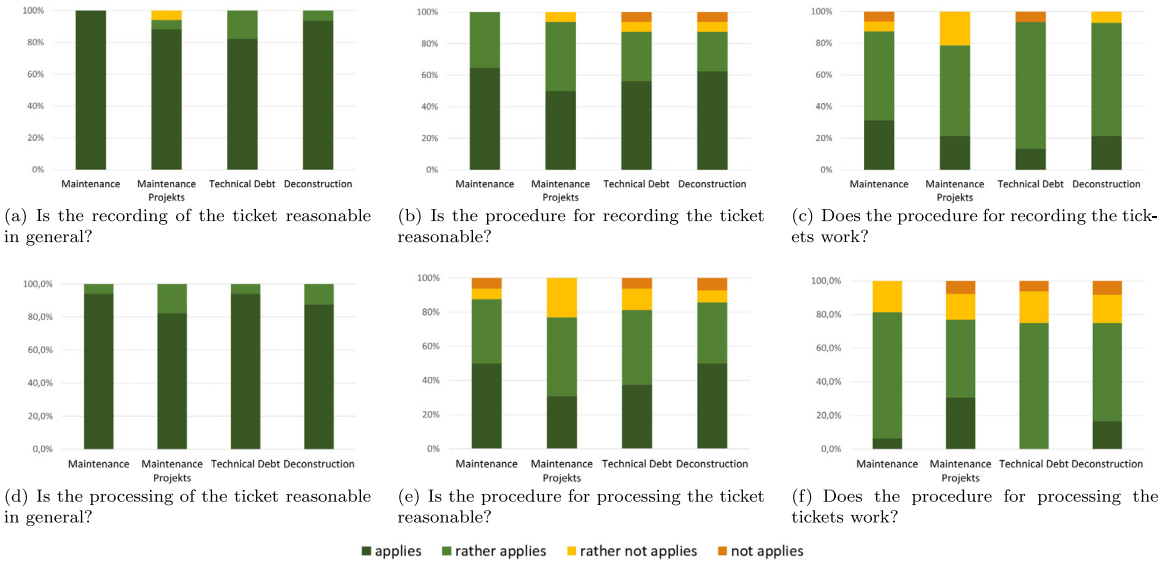
5.1. Ticket statistics

The timelines in Figs. 2(a) and 2(b) show the development of ticket counts over time to answer RQ 1.2. (feasibility) and RQ 2.4 (timely repayment). The maintenance tickets were created and processed continually. The timeline for TD tickets shows a peak in incurred TD in spring 2019 due to a project with a tight deadline. No maintenance or TD tickets were processed in March 2019. Accordingly, the TD tickets as well as the maintenance tickets show a processing peak after the reached deadline at the end of April 2019. Initially, only the TD tickets with the highest priority were processed. From July to November 2019, the TD tickets with a lower priority were processed, meaning that this project was still not completed (see Fig. 1). Other projects that started in parallel caused this long duration because development capacities had to be divided between the projects.

5.2. Team survey

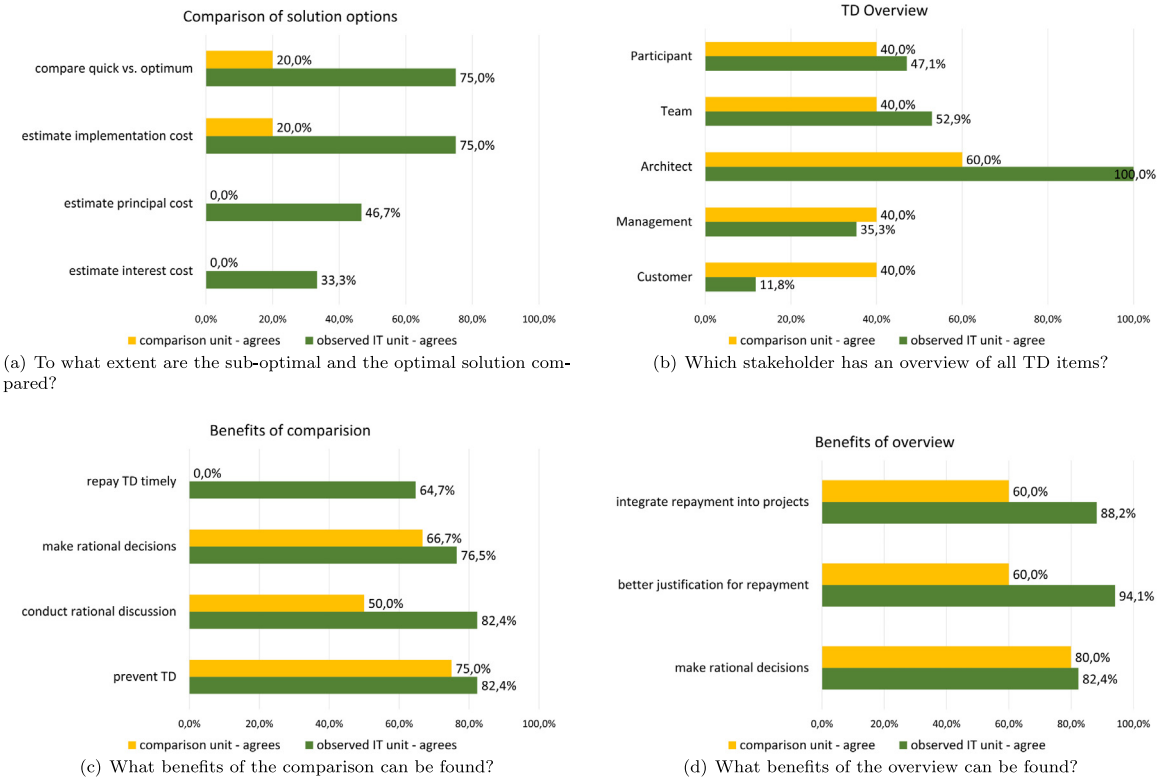
5.2.1. Assessment of the TAP framework

Most survey participants agreed that recording and processing all four types of tickets are generally reasonable (Figs. 3(a) and 3(d)). Most of them also agreed that the framework's procedures for recording and processing are reasonable (Figs. 3(b) and 3(e)). When asked whether the framework worked as intended, the agreement decreased, but most of the survey participants still stated that the procedures work well and are helpful (Figs. 3(c) and 3(f)). For RQ 1.1 and RQ 1.2, these results mean the TAP framework is reasonable and feasible in practice.



(a) Is the recording of the ticket reasonable in general?
(b) Is the procedure for recording the ticket reasonable?
(c) Does the procedure for recording the tickets work?
(d) Is the processing of the ticket reasonable in general?
(e) Is the procedure for processing the ticket reasonable?
(f) Does the procedure for processing the tickets work?

Fig. 3. Appropriateness of the TAP Framework — Descriptive Statistics.



(a) To what extent are the sub-optimal and the optimal solution compared?
(b) Which stakeholder has an overview of all TD items?
(c) What benefits of the comparison can be found?
(d) What benefits of the overview can be found?

Fig. 4. Usability, effects, and benefits of the TAP Framework — Descriptive Statistics.

5.2.2. Perceived effects of the TAP framework

First, we asked survey participants whether they and their team typically recognize that they incurred TD items before or after their incurrence. The goal of this question was to identify whether the participants were aware of the incurrence to answer RQ 2.1. With more than 75.0% agreement, both units stated that they usually recognized it when TD items were incurred. The results are not significantly different.

Second, the participants were asked whether they compare different solution options and estimate respective implementation costs, principal, and interest rates for incurring TD. To answer RQ 2.2, we assumed a conscious and intentional decision to incur TD if the units compared different solution options. Regarding this question, an apparent difference between the units could be seen in Fig. 4(a). 75.0% of the observed unit's participants agreed to compare sub-optimal and optimal (or more) solutions and to calculate implementation cost for this comparison. Only 20.0% of the participants agreed to compare options and estimate implementation costs for the comparison unit. Furthermore, many participants of the observed unit agreed that they estimate principal (46.7%) and interest costs (33.3%), while this was not the case for the comparison unit.

These observations led to the following research hypothesis (H_R) in comparison to the respective null hypothesis (H_0):

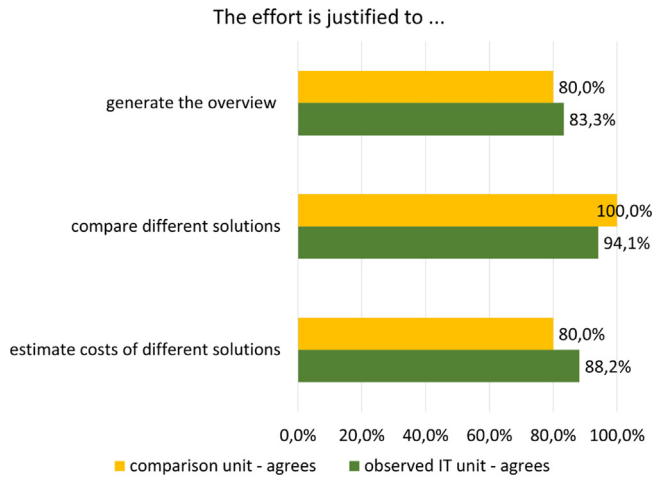


Fig. 5. Do the survey participants think the extra effort for this is justified?.

Table 4
Significance of H_R-hypothesis.

H_R	Variable	Exact significance
$H_{R.1}$	Comparison	0.058
$H_{R.2}$	Implementation cost	0.048
$H_{R.3}$	Principal cost	0.002
$H_{R.4}$	Interest cost	0.015

$H_{R.1}$: The observed unit compares sub-optimal and optimal solutions more frequently than the comparison unit. ($H_{0.1}$: The observed unit does not show different behavior than the comparison unit in terms of comparing sub-optimal and optimal solutions.)

$H_{R.2\text{-}4}$: The observed unit estimates the implementation (and principal and interest) costs for different solutions more frequently than the comparison unit. ($H_{0.2\text{-}4}$: The observed unit does not show different behavior than the comparison unit in terms of estimating the implementation (and principal and interest) costs for different solutions.)

Table 4 shows the significance values of these hypotheses measured with the Mann–Whitney U-Test. The exact significance for $H_{R.2\text{-}4}$ hypotheses is below 0.05. Consequently, these hypotheses can be accepted as they are significant to the 5% significance level. This means the possibility these hypotheses are wrong is lower than 5%. The exact significance for $H_{R.1}$ hypothesis is 0.058 and a little higher than 5%. $H_{R.1}$ cannot be accepted. While these statistics give an additional indication on the difference between the units, we have to keep the low number of data points in mind.

In summary, this means the framework's adoption is associated with a more conscious comparison of different solutions options. In particular, the estimate of the respective costs of the solution options differs significantly between these two units.

Third, we asked whether the framework's adoption generated an overview of TD items to answer RQ 2.3. To evaluate this, we asked the participants which stakeholder has an overview of all TD. Fig. 4(b) shows that in the observed IT unit, the architects had an overview of all TD (100% agreement). The customer was not expected to know the system's TD (11.8% agreement). In the case of the comparison unit, it was unclear which stakeholder had an overview of TD.

5.2.3. Perceived benefits of the TAP framework

To answer RQ 3.1 and 3.2, we asked the participants what benefits they expected (comparison unit) or assessed (observed unit) of the comparison of different solution options and, thus, the conscious incurrence of TD. Fig. 4(c) shows that both units mostly agreed that this comparison could prevent TD. Especially the observed unit assessed that discussions were led more rational (82.4% agreement) and more rational decisions were made (76.5% agreement). Timely repayment (RQ 2.3) of TD was not expected by the comparison unit but assessed by many participants of the observed unit (64.7% agreement). This difference may partly be due to a misunderstanding that the observed unit was unconsciously assessing the framework's benefits and not just the comparison's benefits. The units show no relevant differences in benefits, which means no hypotheses can be derived.

To answer RQ 3.2, the participants were asked what benefits they expected or observed from the generated overview of TD items. The observed unit's participants noted that integrating the TD items into existing or upcoming projects is easier with an overview (88.2%). Furthermore, the overview gives a better justification for the repayment (94.1%) (see Fig. 4(d)). Only 60% of the comparison unit expected the overview's benefits, which is a noticeable smaller but still not significant amount. Both units expected or observed more rational decisions due to the generation of the overview.

Finally, to answer RQ 3.3, the participants were asked whether the benefits justified the effort for comparing and estimating the options and generating the TD overview. Both units mostly thought the effort was justified (>80.0% agreement). Again, the units show no apparent differences (see Fig. 5).

5.2.4. Correlations between effects and benefits

As shown in the previous section, the main difference caused by the framework was that the observed team compared different solution options and estimated the costs for new TD tickets. We observed different benefits that may result from this comparison and estimations: more rational discussions, more rational decisions, and the prevention of TD. To support the observation and explore the possibility of a causal effect between the comparison and estimations and these benefits, we calculated the correlations between them. We also calculated the significance of these correlations and assumed a correlation if the significance was less than 0.05.

Table 5 shows significant correlations of medium strength between the perceived benefit of making more rational decisions and the observed effects of comparing solutions, estimating implementation costs, and estimating principal costs. The benefit of leading more rational discussions has significant correlations of medium strength to the estimation of implementation and principal costs.

Following our observations and the correlations, we can associate the comparison and estimations lead with more rational decisions. Furthermore, we associate the estimation of implementation and principal costs with more rational discussions. Both associations seem to be valid assumptions since they are also logically derivable.

The correlations do not support associations for TD prevention or the estimation of interest costs.

As we did not have a significant difference between the units related to the overview of TD, we did not calculate correlations for the overview's perceived benefits.

5.2.5. Open questions

Most comments were only made one time and are single opinions. Two points were mentioned three times each.

First, "TD is a topic" expresses that it was already helpful to talk about the topic of TD and not to ignore it. The code is related to the goal of raising TD awareness. One participant pointed out: "*I actually see the main advantage of conscious handling of the topic in the matter as such, because otherwise, it is a topic that is almost ignored in everyday development.*"

Second, "Sometimes TD is OK" shows that there were situations in which participants considered retaining TD reasonable, e.g., in legacy systems. Another participant stated: "*If a system is to be replaced, then I consider technical debts to be justifiable if they disappear after 1–2 years anyway*".

A third point was mentioned two times: "Evaluation is difficult and subjective". This topic refers to the research topic of TD prioritization.

Table 5
Correlations of comparison and estimations vs. benefits (Significant correlations are highlighted).

	TD prevention		Rational discussions		Rational decisions	
	ϕ-coeff	signif.	ϕ-coeff	signif.	ϕ-coeff	signif.
Comparing solutions	0.419	0.061	0.303	0.176	**0.623**	**0.007**
Estimating implementation costs	0.357	0.110	**0.471**	**0.035**	**0.535**	**0.020**
Estimating principal costs	0.394	0.086	**0.456**	**0.047**	**0.495**	**0.036**
Estimating interests costs	0.309	0.179	0.357	0.120	0.385	0.103

The participants mentioned that it is sometimes hard to decide which TD should be repaid first and whether a particular item is TD at all, especially if the architectural rules seem to be open for interpretation. One participant phrased it as follows: "*Even if the "teaching book" says that the current implementation does not correspond to the architectural specifications, it still seems to be a personal opinion that decides on the fundamental correctness of the specification.*" The other participant mentioned that TD is easy to define as a deviation from the norm or rules. However, it is "*very difficult to evaluate this deviation and work out the importance of fixing this debt.*"

One participant also mentioned that the "framework raises the understanding for each other": "*I'm thinking of greater domain understanding in development and greater technical understanding for business analysts.*" Regarding the "more rational discussions", one participant states: "*If the process is consciously lived and ingrained in people's minds, it reduces energy-sapping discussions about fixing technical debt.*" Another participant summarizes these effects of the comparison: "*The conscious handling of different possible solutions promotes cooperation, mutual understanding, and knowledge exchange in the team and thus generally contributes to the development of stable and accepted software.* " Lastly, one participant acknowledges the "importance of TD prevention": "*The incurrence of debt should be reduced, not the process for working it off optimized.*"

The following **drawbacks** were mentioned by the participants. One participant stated that the categories' distinction and the separation of their responsibilities might lead to a separation of the team. "*Although we keep saying that we are one team, we often stay separate and the different tickets in particular help make the differences between the team parts visible*". Another participant thought the framework "*(slightly) contradicts the agile principle of providing a functioning solution with company value as quickly as possible. At the same time, however, it levels the stable ground so that we can continue to work on new requirements quickly and without production errors. The advantage outweighs*". One participant of the operations team finds that in their team TD is not a topic and that "*This is a concept that takes place in the development cosmos.*" He recommends that the operations team should be involved in the repayment of TD.

5.3. Follow-up management survey

In this survey, we asked the IT managers of the observed and comparison unit for their perspective on their units' TD management. The unit manager responsible for both units answered for each unit separately. Thus, we got four different responses from three participants:

- team manager of the observed unit (TMO), participant A
- team manager of the comparison unit (TMC), participant B
- unit manager of the observed unit (UMO), participant C
- unit manager of the comparison unit (UMC), participant C

Overview of TD. The TMO and UMO both stated they had an overview of their TD. Both managers relied on a weekly meeting with the architects to establish this overview. In these meetings, architects and IT managers reviewed the architecture, planed the evolution of the system and architecture, and discussed the TD and maintenance projects. The architects prepared the meeting by providing a prioritized pipeline of maintenance projects and an architectural evaluation of the TD items. The TMC and UMC did not have a specific mechanism to create an overview of their TD. Still, they remarked that they had a "*holistic project pipeline that identifies technical debts as such and made it possible to plan corresponding topics in an integrative manner.*" The unit manager, who experienced the strategies of both units, remarked regarding the comparison unit: "*In terms of future operational responsibility, I would feel more comfortable if the technical debt in the project was managed continuously and transparently.*" He stated his concern that "*TD does not become clear until the handover of operations or even during ongoing operations.*"

Project Pipeline. For the TMO and UMO, the overview had a direct impact on the planning of maintenance projects. It also helped to plan capacities and estimate project duration realistically. The UMO even mentioned the possibility that "new projects start later". The TMC stated that TD "*projects are given a further measurable criterion for their urgency.*" The TMC and UMC further mentioned that it was still a problem to prioritize "between business requirements and technical necessity", while the TMO or UMO did not mention this problem.

TD Repayment Activities. The TMO and UMO planed their TD repayment activities during their weekly meeting with the architects. During these meetings, it became apparent in which cases it was sensible just to accept the TD, e.g., when the system will be re-engineered soon. The UMO further mentioned the effect the visibility may have had for the teams: "*The existing debts become visible, and in the end, developers and product owners have to decide whether further debts are justifiable or realistically repayable in the foreseeable future.*" The TMC focused on TD awareness and implied that the overview led to enhanced awareness. He stated that "*TD awareness leads to TD avoidance*" and that TD awareness "*supports the case for projects and measures to reduce this debt*".

Communication with Customers. The TMO and UMO said the pipeline has helped make effects visible, making it easier to communicate with customers. Project durations were better understood and accepted. The UMO mentioned that "*it also reflects the customers' interest in (a) being able to use the right features on time, but (b) also to use a stable and future-proof system in the medium to the long term*". He remarked that this also included communication with other stakeholders, e.g., higher management or users. The TMC saw the same benefits but pointed out that the understanding also depended on the customer's technical know-how.

6. Discussion

In this section, we will finally answer our RQs from Section 4.1 regarding the TAP framework developed by practitioners. We used different methods to answer our research questions. Table 6 gives an overview of which method and figure will be used to answer the respective questions.

To give a better overview of the results, we generated a variation of a cause–effect diagram (Fig. 6) summarizing the effect, benefits, and drawbacks. The diagram is based on the qualitative and quantitative data found in this study. Moreover, we took logical deduction into account for some of the more detailed connections, e.g the repayment phase leads to timely repayment. The literature review led to some connections that supports the qualitative data. In addition, we summarized the codes at a higher level to give a better overview. The following sections discuss these cause–effect mechanisms in detail and answer the RQs.

Table 6
Overview of methods and figures used to answer the respective questions and results section containing details.

RQ	Method	Results section	Figure/Table
RQ 1.1 (TAP framework's reasonability)	Team survey	5.2.1	Figs. 3(a), 3(b), 3(d) and 3(e)
RQ 1.2 (TAP framework's feasibility)	Team survey	5.2.1	Figs. 3(c) and 3(f)
	Tickets statistics	5.1	Fig. 2
RQ 2.1 (raised awareness)	Team survey	5.2.2	
		5.2.5	
	Management survey	5.3	
RQ 2.2 (conscious incurrence)	Team survey	5.2.2	Fig. 4(a), Table 4
		5.2.5	
RQ 2.3 (TD items' overview)	Team survey	5.2.2	Fig. 4(b)
RQ 2.4 (timely repayment)	Team survey	5.2.3	Fig. 4(c)
		5.2.5	
	Tickets statistics	5.1	Fig. 2
RQ 3.1 (TD prevention)	Team survey	5.2.3	Fig. 4(c)
RQ 3.2 (benefits and drawbacks)	Team survey	5.2.3	Fig. 4(c), Fig. 4(d)
		5.2.4	Table 5
		5.2.5	
RQ 3.3 (effort's justification)	Team survey	5.2.3	Fig. 5
RQ 4.1 (managers' TD overview)	Management survey	5.3	
RQ 4.2 (benefits of the managers' TD overview)	Management survey	5.3	

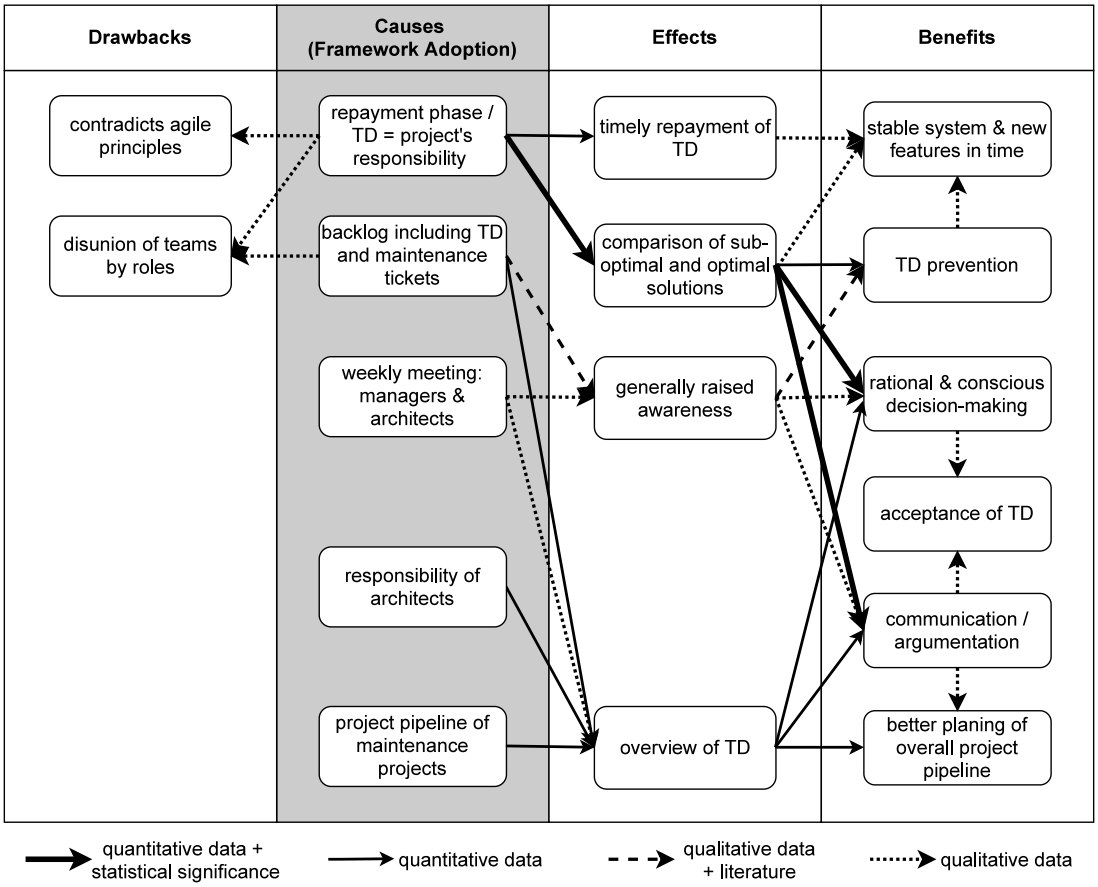


Fig. 6. Causes, perceived effects, and perceived benefits and drawbacks (arrows show effect-direction) of the TAP framework.

6.1. Application of the TAP framework

RQ 1.1/RQ 1.2: *Do practitioners find the TAP framework reasonable? Are the processes of the TAP framework feasible in practice?*

Figs. 3(a) to 3(f) shows little disagreement. Most team survey participants found the framework reasonable and thought it did work as intended. Only a few participants thought it must be optimized. The ticket statistics (see Section 5.1) indicate that the tickets were recorded and processed as intended. We conclude that the framework is feasible in practice and the following findings based on this framework are valid.

6.2. Perceived effects of the TAP framework

RQ 2.1: *Is the use of the TAP framework associated with a raised awareness for the incurrence of TD?*

On the one hand, 75% of the participants stated they were aware of taking on TD at the point in time they incur it. There was no difference between the observed and the comparison unit. On the

other hand, the answers to the open questions support the finding that the awareness was raised considerably ("TD is a topic") and that the participants started to consider when to incur or keep TD and when not to ("Sometimes TD are OK"). In the management survey, the managers also imply that the awareness was raised. This might indicate that the awareness before introducing the framework was lower in the observed unit than in the comparison unit. However, we found out that the survey question on whether the participants are aware of the TD incurrence had great potential to be misunderstood. Furthermore, developers may be biased to think they always know when they incur TD.

RQ 2.2: *Is the use of the TAP framework associated with a more conscious incurrence of TD items?*

Fig. 4(a) reveals that the observed unit showed a significantly different behavior than the comparison unit (Table 4). The observed unit's members compared different solutions and their costs before deciding on one of them. They may still have incurred TD by choosing the suboptimal solution, but the benefit was a consciously and intentionally made decision.

RQ 2.3: *Is the use of the TAP framework associated with a better overview of TD items?*

In Fig. 4(b), it is striking that in the observed IT unit, the architects were the ones who had the overview of all TD items (100% agreement). The responsibility to keep an overview of all tasks rested with the architects. We could not observe such a clear responsibility in the comparison unit. A TD overview might also be created in the comparison unit by other means. However, the evaluation indicates that the clear responsibility of the architects, which led to the TD overview, was an additional benefit of the framework.

RQ 2.4: *Is the use of the TAP framework associated with a timelier repayment of TD items?*

If the unit's members had to incur TD, they mostly paid it back timely, as seen in Fig. 2(b). Additionally, Fig. 4(c) shows that two-thirds of participants in the observed IT unit perceived they repaid TD timelier. Nevertheless, the unit did not repay all TD tickets during their respective projects, and one-third of participants were still dissatisfied with the timely repayment.

RQ 4.1: *Do the managers have an overview of their IT systems' TD?*

The observed unit's managers had an overview of their TD. The architects were tasked to (a) manage the maintenance project pipeline and (b) track TD and other maintenance tasks in the backlog. The managers got their overview primarily due to the weekly meeting with the architects. They profited from the architectural evaluation of these tasks. The comparison unit's managers did not have a comprehensive overview of their TD.

6.3. Perceived benefits and drawbacks of the TAP framework

RQ 3.1/RQ 3.2: *Do practitioners observe that TD can be prevented by the adoption of the TAP framework? Are there other benefits or drawbacks arising from the adoption of the TAP framework?*

The most apparent perceived effect of the framework was the conscious incurrence of TD by comparing different solution options. In Fig. 4(c), we presented the benefits associated with comparing solution options.

To back up these findings, we evaluated the correlation between the perceived benefits and the comparison of solution options and the estimations. We assume that the comparison and estimations can be associated with more rational discussions and decisions following the significant correlations.

The correlations (Table 5) do not support associations for TD prevention or the estimation of interest costs. The missing correlations for interest costs may be since only one-third of the participants estimate these costs. The missing correlations for TD prevention lead to the assumption that the prevention of TD is maybe not directly caused by the comparison and estimations. Still, TD prevention was perceived as a benefit by 82.4% of the participants, and it was mentioned in the open questions. Furthermore, it logically follows that rational decision-making may be able to prevent TD.

The TD overview's perceived benefits were more rational decisions and better planning of the project pipeline by justifying TD repayments and integrating TD repayments into projects (Fig. 4(d)).

Finally, the open questions support that the awareness for TD was raised, and activities like prioritization were discussed throughout the unit. Furthermore, optimized understanding and discussions were mentioned.

The researchers observed no particular drawbacks other than the (initial) effort to use the framework (see RQ 3.3). Therefore, we asked for drawbacks in open questions. One participant each mentioned the lack of adherence to agile processes and the possible separating affect the framework may have on the team. Finally, it was suggested to better involve the operations team.

RQ 3.3: *Do practitioners find these benefits justify the additional effort?*

Fig. 5 shows that most participants of the team survey think that the perceived benefits justified the additional effort for comparison, cost estimation, and the generation of the overview. These results confirm the TAP framework's usefulness.

RQ 4.2: *What are the benefits of the TD overview from a management perspective?* All IT managers agreed that TD items became visible and, thereby, the overall awareness for TD increased. In particular, the project pipeline's management profited from the visibility and architectural evaluation of TD. The amount of TD could be used to measure the importance of projects. The project's durations became discernible, and the managers could postpone subsequent projects if necessary. The managers noticed that decisions regarding the incurrence of TD were made more conscious and, by this, TD could be prevented. The communication between IT management and teams improved. The better overview led to better communication with the customers and a better acceptance of repayment measures. Ultimately, TD management has been in the customer's interest as it provides stable systems and new features in time.

6.4. Threats to validity

Construct Validity. To enhance the construct validity, we questioned and compared two units, one of which did adopt the TAP framework and one did not use any method for managing TD. As presented in Section 4.2, the scope and size of the comparison unit differ from the observed unit. However, the organizational form, particularly the agile-managed projects with deadlines, is identical. Since the time limitation is the most important basis for a meaningful framework adoption, we assume that the comparison of units is valid.

To ensure comprehensibility and to optimize the questionnaire, we did pilot tests. It could be a problem that the team survey did not ask directly but only in open questions for drawbacks. Furthermore, the quantitative survey's construction was not preceded by a qualitative survey but only by the researcher's observations. The open questions and the question about the effort's justification should have addressed parts of this problem. Additionally, we not only evaluated the questionnaire but also the ticket's statistics. This data triangulation improves the construct validity. For observer triangulation, two researchers independently coded the open questions of the team survey and the management survey. Only three managers answered the follow-up management survey. Thus, this survey only provides qualitative insights. However, the unit manager who answered this survey is the co-author of this paper and may be biased. Yet, this co-author was only involved in developing the framework and writing the framework section. He was not involved in the scientific evaluation. Still, he may be biased towards the positive impact of the framework. Two co-authors utterly uninvolved with the company enhance the validity despite a possible author's bias.

Conclusion Validity. The paper was able to show statistically significant differences between the observed and the comparison unit. We used SPSS and standard statistic techniques to evaluate the findings' significance. Nonetheless, the sample was very small as it can only encompass the two IT units, and the staff survey had an overall response rate of 55%. We carefully considered the small sample size when interpreting the findings. We suggest supporting the findings and assumptions by further replicating this study to gain more data points.

Internal Validity. We observed valuable effects, but the same effects could have been observed in other IT units. This threat was minimized by the comparison with a unit not using the TAP framework. Nevertheless, just the management of TD could have led to better developer morale as presented by Besker et al. [48] and consequently to good ratings. This threat can and should be minimized by replicating the study in other constellations. As correlations do not directly lead to causalities, the mentioned interpretations of the correlations should be validated, e.g., by follow-up interviews.

External Validity. The TAP framework has been developed and tested in an industrial environment, and it has been in use for a long time. Both these facts are meaningful benefits of this research. Nevertheless, this was just one case study, and it may be biased. We suggest replicating the framework adoption and the study in other units and companies, economic sectors, and countries to confirm the findings. Furthermore, the comparison unit should adopt the framework, or the reasons for not adopting this should be evaluated. We suggest focusing on the most relevant parts which were already mentioned in Section 3.4: the distinction in processing intentional TD as TD tickets and unintentional TD as part of maintenance tickets, the immediate recording of intentional TD, and the processing of intentional TD as part of the project they were incurred in.

For this, we provided the framework information in this paper and the questionnaires and raw data online.[8]

7. Conclusion and future work

In this paper, we present and evaluate the TAP framework developed and used in practice for a long time. The framework uses TD tickets for intentionally incurred TD to include TD management in the project management timeline. These TD tickets transfer the responsibility for TD incurred during a project to the same project's project management. The team records TD tickets at the time they incur the TD. This creates a feedback loop of team and project management behavior in the observed unit. They perceived the following benefits:

- The communication between different stakeholders, including IT management and customers, is optimized, and discussions between the unit members are led more rationally.
- The overall awareness for TD rises by making TD visible and providing an overview of TD.
- TD tickets raise the awareness for the decision of whether TD should be incurred and make these decisions conscious and intentional.
- TD items unintentionally incurred due to unconscious decisions can be prevented.
- TD items intentionally incurred during a project are part of the project and are paid back timelier.

Furthermore, TD tickets provide an in-time overview for the IT managers as these items are tracked during the project. The management survey indicates that this enables the managers to react and adjust the project pipeline if necessary.

While the idea of maintenance tickets and projects is not new, the idea of TD tickets for intentional TD and the proposed processing of these tickets as part of the project is a novel approach. To the best of the authors' knowledge, such an approach cannot be found in the research literature so far.

For industry, this paper presents a framework that can be adopted and adapted to their own needs. The idea of including TD management in project management can be an incentive to develop a similar approach. The paper demonstrates the importance of raising TD awareness and of incurring TD intentionally by taking the time to compare different solution options. The advantages shown by this paper can be helpful when negotiating options to manage TD with IT management.

For researchers, this paper provides insights into procedures for managing TD developed in the industry. The paper indicates the importance of providing a solution for managing and preventing TD under the pressure of tight deadlines. Furthermore, this research confirms that the responsibility for TD needs to be transferred to the whole unit. Developer teams cannot solve these problems alone. The TAP framework provides a solution for these problems.

For future work, we believe it could be helpful to supplement the framework with more formal processes for some TD activities, e.g., automated TD identification, TD measurement, or TD prioritization. In particular, approaches for TD prioritization may help optimize the repayment phase, e.g., a consequent distinction of potential and effective TD [46], as our research uncovered this to be a problem.

The effects of TD management on the planning of project pipelines is an intriguing aspect. Many underlying problems of TD management refer to communications issues between stakeholders, e.g., managers and developers. Hence, we think that the perspectives of different stakeholders, e.g., line managers, on TD is another relevant research topic, which we would like to evaluate further.

At various points, we suggested replicating the study. However, this framework was designed as an actual frame to be adopted in whole or part. This is not a weakness of the framework, but takes into account the reality in which IT units' processes need to be adapted to their own needs. We can see this, e.g., in the ITIL (Information Technology Infrastructure Library) processes or agile development methods. We hope to find other units that are willing to adopt all or some of the relevant parts of the framework mentioned in Section 3.4. It would be interesting to evaluate which parts may lead to similar effects and benefits in other companies.

CRediT authorship contribution statement

Marion Wiese: Conceptualization, Methodology, Resources, Formal analysis, Investigation, Data Curation, Writing – original draft, Visualization, Project administration. **Paula Rachow:** Methodology, Validation, Formal analysis, Investigation, Visualization, Writing – review & editing. **Matthias Riebisch:** Validation, Visualization, Writing – review & editing, Supervision. **Julian Schwarze:** Methodology, Resources, Writing – review & editing.

Declaration of competing interest

One or more of the authors of this paper have disclosed potential or pertinent conflicts of interest, which may include receipt of payment, either direct or indirect, institutional support, or association with an entity in the biomedical field which may be perceived to have potential conflict of interest with this work. For full disclosure statements refer to https://doi.org/10.1016/j.infsof.2022.106926. The corresponding author Marion Wiese was employed by Gruner + Jahr GmbH until March 31, 2020, and was a team member in the case unit under consideration. At the time of the development of the framework, she was therefore part of the unit under consideration and contributed to its development in her role as software architect. The evaluation of the framework took place after her employment. Julian Schwarze, co-author, is the unit leader of the case unit under consideration and also contributed to the development of the framework.

[8] https://doi.org/10.5281/zenodo.5788222.

Acknowledgment

We would like to thank the observed and comparison unit of *Gruner + Jahr GmbH - Information Technology Department* for contributing to the framework, completing the questionnaires, and supporting this research.

References

[1] W. Cunningham, The WyCash portfolio management system, in: Proceedings of the Conference on Object-Oriented Programming Systems, Languages, and Applications, OOPSLA, Vol. 2, 1992, pp. 29–30, http://dx.doi.org/10.1145/157710.157715.

[2] P. Avgeriou, P. Kruchten, I. Ozkaya, C. Seaman, Managing technical debt in software engineering, Dagstuhl Rep. 6 (4) (2016) 110–138, http://dx.doi.org/10.4230/DagRep.6.4.110.

[3] P. Avgeriou, P. Kruchten, R.L. Nord, I. Ozkaya, C. Seaman, Reducing friction in software development, IEEE Softw. 33 (1) (2016) 66–73, http://dx.doi.org/10.1109/MS.2016.13.

[4] N. Rios, R.O. Spínola, M. Mendonça, C. Seaman, The practitioners' point of view on the concept of technical debt and its causes and consequences: a design for a global family of industrial surveys and its first results from Brazil, Empir. Softw. Eng. 25 (5) (2020) 3216–3287, http://dx.doi.org/10.1007/s10664-020-09832-9.

[5] R. Verdecchia, P. Kruchten, P. Lago, I. Malavolta, Building and evaluating a theory of architectural technical debt in software-intensive systems, J. Syst. Softw. 176 (2021) 110925, http://dx.doi.org/10.1016/j.jss.2021.110925.

[6] S. McConnell, Managing Technical Debt, Construx Inc., 2008.

[7] M. Fowler, Technical debt, 2019, URL https://martinfowler.com/bliki/TechnicalDebt.html.

[8] D. Tsoukalas, D. Kehagias, M. Siavvas, A. Chatzigeorgiou, Technical debt forecasting: An empirical study on open-source repositories, J. Syst. Softw. 170 (2020) 110777, http://dx.doi.org/10.1016/j.jss.2020.110777.

[9] S. Freire, N. Rios, B. Gutierrez, D. Torres, M. Mendonça, C. Izurieta, C. Seaman, R.O. Spinola, Surveying software practitioners on technical debt payment practices and reasons for not paying off debt items, in: Proceedings of the Evaluation and Assessment in Software Engineering, in: EASE '20, Association for Computing Machinery, New York, NY, USA, 2020, pp. 210–219, http://dx.doi.org/10.1145/3383219.3383241.

[10] C. Apa, H. Jeronimo, L.M. Nascimento, D. Vallespir, G.H. Travassos, The perception and management of technical debt in software startups, in: Fundamentals of Software Startups: Essential Engineering and Business Aspects, 2020, pp. 61–78, http://dx.doi.org/10.1007/978-3-030-35983-6_4.

[11] J. Yli-Huumo, A. Maglyas, K. Smolander, How do software development teams manage technical debt? – An empirical study, J. Syst. Softw. 120 (2016) 195–218, http://dx.doi.org/10.1016/j.jss.2016.05.018.

[12] J.D. Morgenthaler, M. Gridnev, R. Sauciuc, S. Bhansali, Searching for build debt: Experiences managing technical debt at google, in: 2012 3rd International Workshop on Managing Technical Debt, MTD 2012 - Proceedings, 2012, pp. 1–6, http://dx.doi.org/10.1109/MTD.2012.6225994.

[13] T. Besker, A. Martini, J. Bosch, Carrot and stick approaches when managing technical debt, in: Proceedings of the 3rd International Conference on Technical Debt, in: TechDebt '20, Association for Computing Machinery, New York, NY, USA, 2020, pp. 21–30, http://dx.doi.org/10.1145/3387906.3388619.

[14] B. Pérez, C. Castellanos, D. Correal, N. Rios, S. Freire, R. Spínola, C. Seaman, C. Izurieta, Technical debt payment and prevention through the lenses of software architects, Inf. Softw. Technol. 140 (2021) 106692, http://dx.doi.org/10.1016/j.infsof.2021.106692.

[15] S.S. Freire, N. Rios, B. Pérez, C. Castellanos, D. Correal, R. Ramač, V. Mandić, N. Taušan, G. López, A. Pacheco, D. Falessi, M. Mendonça, C. Izurieta, C. Seaman, R. Spínola, How experience impacts practitioners' perception of causes and effects of technical debt, in: 2021 IEEE/ACM 13th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE), 2021, pp. 21–30, http://dx.doi.org/10.1109/CHASE52884.2021.00011.

[16] J.C. Rocha, V. Zapalowski, I. Nunes, Understanding technical debt at the code level from the perspective of software developers, in: Proceedings of the 31st Brazilian Symposium on Software Engineering, in: SBES'17, Association for Computing Machinery, New York, NY, USA, 2017, pp. 64–73, http://dx.doi.org/10.1145/3131151.3131164.

[17] P. Kruchten, R.L. Nord, I. Ozkaya, Technical debt: From metaphor to theory and practice, IEEE Softw. 29 (6) (2012) 18–21, http://dx.doi.org/10.1109/MS.2012.167.

[18] U. Eliasson, A. Martini, R. Kaufmann, S. Odeh, Identifying and visualizing architectural debt and its efficiency interest in the automotive domain: A case study, in: 2015 IEEE 7th International Workshop on Managing Technical Debt (MTD), 2015, pp. 33–40, http://dx.doi.org/10.1109/MTD.2015.7332622.

[19] Y. Guo, R.O. Spínola, C. Seaman, Exploring the costs of technical debt management – a case study, Empir. Softw. Eng. 21 (1) (2016) 159–182, http://dx.doi.org/10.1007/s10664-014-9351-7.

[20] T. Besker, A. Martini, J. Bosch, Technical debt triage in backlog management, in: 2019 IEEE/ACM International Conference on Technical Debt (TechDebt), in: TechDebt '19, IEEE Press, 2019, pp. 13–22, http://dx.doi.org/10.1109/TechDebt.2019.00010.

[21] M. Soliman, P. Avgeriou, Y. Li, Architectural design decisions that incur technical debt — An industrial case study, Inf. Softw. Technol. 139 (2021) 106669, http://dx.doi.org/10.1016/j.infsof.2021.106669.

[22] K. Borowa, A. Zalewski, S. Kijas, The influence of cognitive biases on architectural technical debt, in: International Conference on Software Architecture (ICSA), 2021, pp. 115–125, http://dx.doi.org/10.1109/ICSA51549.2021.00019.

[23] T. Besker, J. Bosch, A. Martini, J. Bosch, Technical debt cripples software developer productivity: A longitudinal study on developers' daily software development work, in: Proceedings - International Conference on Software Engineering, ACM, 2018, pp. 105–114, http://dx.doi.org/10.1145/3194164.3194178.

[24] A. Martini, J. Bosch, M. Chaudron, Architecture technical debt: Understanding causes and a qualitative model, in: 2014 40th EUROMICRO Conference on Software Engineering and Advanced Applications, 2014, pp. 85–92, http://dx.doi.org/10.1109/SEAA.2014.65.

[25] Z. Li, P. Avgeriou, P. Liang, A systematic mapping study on technical debt and its management, J. Syst. Softw. 101 (2015) 193–220, http://dx.doi.org/10.1016/j.jss.2014.12.027.

[26] N.A. Ernst, S. Bellomo, I. Ozkaya, R.L. Nord, I. Gorton, Measure it? Manage it? Ignore it? Software practitioners and technical debt, in: 10th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, ESEC/FSE 2015 - Proceedings, in: ESEC/FSE 2015, Association for Computing Machinery, New York, NY, USA, 2015, pp. 50–60, http://dx.doi.org/10.1145/2786805.2786848.

[27] V. Lenarduzzi, T. Orava, N. Saarimäki, K. Systa, D. Taibi, K. Systä, D. Taibi, K. Systa, D. Taibi, An empirical study on technical debt in a finnish SME, in: 2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), Vol. 2019-Septe, 2019, pp. 1–6, http://dx.doi.org/10.1109/ESEM.2019.8870169.

[28] S. Freire, N. Rios, M. Mendonça, D. Falessi, C. Seaman, C. Izurieta, R.O. Spínola, R.O. Spínola, R.O. Spínola, Actions and impediments for technical debt prevention: Results from a global family of industrial surveys, in: Proceedings of the ACM Symposium on Applied Computing, in: SAC '20, Association for Computing Machinery, New York, NY, USA, 2020, pp. 1548–1555, http://dx.doi.org/10.1145/3341105.3373912.

[29] B. Vogel-Heuser, F. Bi, Interdisciplinary effects of technical debt in companies with mechatronic products — a qualitative study, J. Syst. Softw. 171 (2021) 110809, http://dx.doi.org/10.1016/j.jss.2020.110809.

[30] J. Yli-Huumo, T. Rissanen, A. Maglyas, K. Smolander, L.-M. Sainio, The Relationship Between Business Model Experimentation and Technical Debt, in: Lecture Notes in Business Information Processing, vol. 210, 2015, pp. 5–6, http://dx.doi.org/10.1007/978-3-319-19593-3.

[31] S. Charalampidou, A. Ampatzoglou, A. Chatzigeorgiou, N. Tsiridis, Integrating traceability within the IDE to prevent requirements documentation debt, in: Proceedings - 44th Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2018, IEEE, 2018, pp. 421–428, http://dx.doi.org/10.1109/SEAA.2018.00075.

[32] Y. Yang, R. Michel, J. Wade, D. Verma, M. Törngren, T. Alelyani, Towards a taxonomy of technical debt for COTS-intensive cyber physical systems, Procedia Comput. Sci. 153 (2019) 108–117, http://dx.doi.org/10.1016/j.procs.2019.05.061.

[33] D. Feitosa, A. Ampatzoglou, A. Gkortzis, S. Bibi, A. Chatzigeorgiou, CODE reuse in practice: Benefiting or harming technical debt, J. Syst. Softw. 167 (2020) 110618, http://dx.doi.org/10.1016/j.jss.2020.110618.

[34] T. Sharma, G. Suryanarayana, G. Samarthyam, Challenges to and solutions for refactoring adoption: An industrial perspective, IEEE Softw. 32 (6) (2015) 44–51, http://dx.doi.org/10.1109/MS.2015.105.

[35] M. Kaiser, G. Royse, Selling the investment to pay down technical debt: The code christmas tree, in: 2011 Agile Conference, 2011, pp. 175–180, http://dx.doi.org/10.1109/AGILE.2011.50.

[36] R.K. Gupta, P. Manikreddy, S. Naik, K. Arya, Pragmatic approach for managing technical debt in legacy software project, in: Proceedings of the 9th India Software Engineering Conference, in: ISEC '16, vol. 18-20-Febr, Association for Computing Machinery, New York, NY, USA, 2016, pp. 170–176, http://dx.doi.org/10.1145/2856636.2856655.

[37] O. Baysal, R. Holmes, M.W. Godfrey, Developer dashboards: The need for qualitative analytics, IEEE Softw. 30 (4) (2013) 46–52, http://dx.doi.org/10.1109/MS.2013.66.

[38] C. Treude, M.A. Storey, Awareness 2.0: Staying aware of projects, developers and tasks using dashboards and feeds, in: Proceedings - International Conference on Software Engineering, Vol. 1, 2010, pp. 365–374, http://dx.doi.org/10.1145/1806799.1806854.

[39] A. Martini, J. Bosch, An empirically developed method to aid decisions on architectural technical debt refactoring: AnaConDebt, in: 2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C), 2016, pp. 31–40.

[40] S. Baars, S. Meester, CodeArena: Inspecting and improving code quality metrics using minecraft, in: 2019 IEEE/ACM International Conference on Technical Debt (TechDebt), 2019, pp. 68–70, http://dx.doi.org/10.1109/TechDebt.2019.00023.

[41] M.A. Al Mamun, C. Berger, J.J. Hansson, Explicating, understanding, and managing technical debt from self-driving miniature car projects, in: 2014 Sixth International Workshop on Managing Technical Debt, Institute of Electrical and Electronics Engineers Inc., 2014, pp. 11–18, http://dx.doi.org/10.1109/MTD.2014.15.

[42] J.-L. Letouzey, M. Ilkiewicz, Managing technical debt with the SQALE method, IEEE Softw. 29 (6) (2012) 44–51, http://dx.doi.org/10.1109/MS.2012.129.

[43] G.S. Tonin, A. Goldman, C. Seaman, D. Pina, Effects of technical debt awareness: A classroom study, in: H. Baumeister, H. Lichter, M. Riebisch (Eds.), 18th International Conference, XP 2017, Vol. 283, 2017, pp. 84–100, http://dx.doi.org/10.1007/978-3-319-57633-6_6.

[44] Y. Guo, C. Seaman, F.Q. da Silva, F. Q.B. da Silva, F.Q. da Silva, F. Q.B. da Silva, Costs and obstacles encountered in technical debt management – A case study, J. Syst. Softw. 120 (2016) 156–169, http://dx.doi.org/10.1016/j.jss.2016.07.008.

[45] N. Ramasubbu, C.F. Kemerer, Integrating technical debt management and software quality management processes: A framework and field test, Test Eng. Manage. 45 (7–8) (2019) 883, http://dx.doi.org/10.1145/3180155.3182529.

[46] K. Schmid, A formal approach to technical debt decision making, in: QoSA 2013 - Proceedings of the 9th International ACM Sigsoft Conference on the Quality of Software Architectures, in: QoSA '13, Association for Computing Machinery, New York, NY, USA, 2013, pp. 153–162, http://dx.doi.org/10.1145/2465478.2465492.

[47] F. Bachmann, R.L. Nord, I. Ozkaya, Architectural tactics to support rapid and agile stability, CrossTalk 25 (3) (2012) 20–25.

[48] T. Besker, H. Ghanbari, A. Martini, J. Bosch, The influence of technical debt on software developer morale, J. Syst. Softw. 167 (2020) 110586, http://dx.doi.org/10.1016/j.jss.2020.110586.

[49] J. Yli-Huumo, A. Maglyas, K. Smolander, J. Haller, H. Törnroos, Developing processes to increase technical debt visibility and manageability – An action research study in industry, in: Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), in: LNCS, vol. 10027, 2016, pp. 368–378, http://dx.doi.org/10.1007/978-3-319-49094-6_24.

[50] R. Rebouças de Almeida, R. do Nascimento Ribeiro, C. Treude, U. Kulesza, Business-driven technical debt prioritization - An industrial case study, in: 2021 IEEE/ACM International Conference on Technical Debt (TechDebt), 2021, pp. 74–83, http://dx.doi.org/10.1109/TechDebt52882.2021.00017, arXiv:2010.09711.

[51] S. Malakuti, S. Ostroumov, The quest for introducing technical debt management in a large-scale industrial company, in: A. Jansen, I. Malavolta, H. Muccini, I. Ozkaya, O. Zimmermann (Eds.), Software Architecture. ECSA 2020. Lecture Notes in Computer Science, Vol. 12292, Springer International Publishing, 2020, pp. 296–311, http://dx.doi.org/10.1007/978-3-030-58923-3_20.

[52] P. Runeson, M. Host, A. Rainer, B. Regnell, Case Study Research in Software Engineering, John Wiley and Sons Ltd, 2012, p. 241, http://dx.doi.org/10.1007/978-3-030-49392-9.

[53] A. Joshi, S. Kale, S. Chandel, D. Pal, Likert scale: Explored and explained, Br. J. Appl. Sci. Technol. 7 (4) (2015) 396–403, http://dx.doi.org/10.9734/bjast/2015/14975.

[54] H.B. Mann, D.R. Whitney, On a test of whether one of two random variables is stochastically larger than the other, Ann. Math. Stat. 18 (1) (1947) 50–60, http://dx.doi.org/10.1214/aoms/1177730491.

[55] J. Cohen, Statistical Power Analysis Fot the Behavioral Sciences, Lawrene Erlbaum Associates, New York, New York, USA, 1988.

[56] K. Pearson, I. Mathematical contributions to the theory of evolution. —VII. On the correlation of characters not quantitatively measurable, Philos. Trans. R. Soc. A 195 (262–273) (1901) 1–47, http://dx.doi.org/10.1098/rsta.1900.0022.

Chapter 12

IT managers' perspective on Technical Debt Management

3 *(IT Managers' Perspective)*	
Title	IT managers' perspective on Technical Debt Management
Authors	Wiese M. and Borowa K.
Year	2023
Journal	Journal of Systems and Software, Volume 202
Reference	[WB23]

TABLE 12.1: Publication No. 3 *(IT Managers' Perspective)*

The Journal of Systems & Software 202 (2023) 111700

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

IT managers' perspective on Technical Debt Management☆

Marion Wiese [a,*], Klara Borowa [b]

[a] Universität Hamburg, Department of Informatics, Vogt-Kölln-Str. 30, 22527 Hamburg, Germany
[b] Warsaw University of Technology, Institute of Control and Computation Engineering, ul. Nowowiejska 15/19, 00-665 Warsaw, Poland

ARTICLE INFO

Article history:
Received 11 October 2022
Received in revised form 23 January 2023
Accepted 5 April 2023
Available online 10 April 2023

Keywords:
Technical Debt
Technical Debt Management
Technical Debt causes
Technical Debt consequences
Technical Debt communication
Legacy systems

ABSTRACT

Context: Technical Debt (TD) is a term for software solutions that are beneficial in the short-term but impede future change.
Goal: Previous research on TD indicates various management-related causes. We analyze the perspective of IT managers on TD since they usually have a major influence on deadlines, the project's budget, and setting up a TD management (TDM) process.
Method: To determine the IT managers' perspective, we obtained and analyzed data from 16 semi-structured interviews and a three-person focus group discussion.
Results: We found that all IT managers understood the TD concept. They consider TDM to be an essential topic, though nearly none of them had set up a TDM process so far. We identified three major concerns the IT managers had regarding TDM: communicating about TD, establishing a TDM process, and dealing with vintage systems, i.e., old legacy systems We developed a model specifying causes and consequences visible to business stakeholders, causal chains, and vicious cycles.
Conclusions: Our research identifies new research gaps and demonstrates to practitioners that investing in a TDM process may be beneficial. It provides the V4CTD model of *Visibility, Cycles & Chains of Causes & Consequences of TD*, extending the TD conceptual model and facilitating communication on TD with business stakeholders.

Editor's note: Open Science material was validated by the Journal of Systems and Software Open Science Board.

© 2023 Elsevier Inc. All rights reserved.

1. Introduction

The term Technical debt (TD) was first used by Cunningham (1992) as a metaphor to explain how delivering software swiftly, results in the need for code rewrites later. The definition of TD has evolved since then. Currently, a single instance of TD, namely a TD item, can be described as a construct that is beneficial in the short term, which hinders further development and maintenance of the software in the long term (Avgeriou et al., 2016; Kruchten et al., 2019). These constructs can occur in various elements of the system or software engineering process, e.g., code, tests, or requirements (Ernst et al., 2021). These days research tends to focus, in particular, on architectural TD (ATD) (Martini and Bosch, 2015; Besker et al., 2018; Verdecchia et al., 2021). ATD was identified as being one of the most dangerous types of TD (Martini and Bosch, 2015), because there is a lack of refactoring strategies that would enable ATD repayment after it is incurred (Besker et al., 2018).

The **rationale** of this study is to focus on the IT managers' perspective on TD since improving the planning and management of TD is a significant factor in minimizing the amount of TD in software projects. In their research on TD causes and consequences, Ramač et al. (2021) show that the largest group of TD causes is related to planning and management. IT managers have a great influence on time and budget resources, i.e., project planning and deadlines, which are the most frequent causes in this category (Ramač et al., 2021). Other researchers likewise found that the effectiveness of Technical Debt Management (TDM) is related to IT managers and their knowledge and overview of TD (Ramasubbu and Kemerer, 2019; Wiese et al., 2022).

Junior et al. identified a lack of research on the business perspective of TD (Junior and Travassos, 2022). IT managers are usually in direct contact with the business stakeholders and negotiate time and budget with them. Therefore, they may have a better understanding of the business managers' perspective on TD.

While the developer's (Rocha et al., 2017), architects (Pérez et al., 2021) and even project manager's (Gomes et al., 2022)

☆ Editor: Christoph Treude.

* Corresponding author.
E-mail addresses: marion.wiese@uni-hamburg.de (M. Wiese), klara.borowa@pw.edu.pl (K. Borowa).

https://doi.org/10.1016/j.jss.2023.111700
0164-1212/© 2023 Elsevier Inc. All rights reserved.

perspectives are taken into consideration by dedicated studies, no study focusing on the IT managers' perspective on TD has been performed so far to the best of the authors' knowledge. Gomes et al. (2022) recent research underlines the significance of this gap by showing that the perspective of management practitioners might be very different from that of software developers.

Therefore, the **goal** of this study is to research the perspective of IT managers on TD by conducting 16 interviews and one three-person focus group discussion. This might provide research gaps or research areas with insufficient focus and enable IT managers to benefit from the analysis of the participant's insights.

Guided by our **research questions (RQ)**, we identified the current knowledge, three main concerns, and potential improvement strategies that IT managers have regarding TD.

RQ 1: What is the current situation regarding TD in the participants' companies?

- ***RQ 1.1.: What and how much do IT managers know about TD?*** To evaluate the involvement of IT managers in TDM, we identified how IT managers interpret the term TD and the knowledge they have about the concept of TD. Misunderstandings in this regard may be a reason for IT managers to not manage TD. Moreover, IT managers may provide previously unidentified knowledge about the concept and mechanics of TD as they are in direct communication with the business managers.
- ***RQ 1.2.: How do IT managers currently manage TD?*** To identify issues with and potential improvements for the current TDM process, we summarized the current state of TDM in the participants' companies.

RQ 2: What are IT managers' main concerns regarding TDM? The IT managers' main concerns regarding TD may differ from the developers' or architects' point of view. The results of this RQ may allow researchers to focus on relevant topics regarding TD.

RQ 3: What measures do IT managers propose to improve TDM? By answering this question, we aimed to identify further potential for TDM improvement. IT managers may be aware of potential improvements to TDM, that are not part of the TD body of knowledge so far.

As the **contributions** of our study, we answered the research questions and gathered the following major insights:

1. The TD metaphor is used as an ambiguous informal term and is not helpful in business communication.
2. There are causes and consequences on different visibility levels for business stakeholders, IT managers, and development teams, which might result in communication issues.
3. We propose to extend the TD conceptual model to include chains and cycles of causes and consequences.
4. To improve (developer) communication and increase the overall awareness of TD, an established TDM process is essential.
5. To overcome the status quo bias, IT managers need more guidance on how to establish a TDM process, preferably through using already established processes and simple methods.
6. Research on how to migrate vintage systems (i.e., systems from the 90s/old legacy systems) and on the possibility of keeping and changing them seems to be still lacking.

With regards to the second and third topics, we extended the conceptual model of TD (Avgeriou et al., 2016) and created the V4CTD model of *Visibility, Cycles & Chains of Causes & Consequences of TD*. This model not only explains the relationships between various causes and consequences but it can also be used to prepare a three-minute pitch to explain a project's TD issues to a business stakeholder.

In the following **subsections**, we present the background information that forms the basis for motivating and understanding this study. In Section 2, we explain our research methodology. We present the results, including the V4CTD model, and answer the research questions in Section 3. Afterward, we discuss our findings with regard to the identified concerns and suggest further research activities in Section 4. We compare our study with related work in Section 5 and discuss the threats to validity of our study in Section 6. With Section 7 we conclude our paper and summarize the contributions for practitioners and researchers.

1.1. Technical debt management

TDM has been intensively researched over the last decade (Li et al., 2015; Rios et al., 2018). In recent years, this resulted in publishing comprehensive management guidelines (Kruchten et al., 2019; Ernst et al., 2021) accessible to software practitioners.

Organizations establishing TDM can achieve various levels of maturity, depending on their current approach to TDM. Yli-Huumo et al. (2016) describe three levels of organizational maturity (no conscious TDM practices, partial implementation of TDM activities, and fully organized TDM). Kruchten et al. (2019) distinguish five levels of TD awareness (from no awareness of the term TD at all to having complete control over TD). Similarly, Martini et al. (2018) describe seven levels of adoption for TD tracking. However, full maturity regarding TDM is not yet a widespread phenomenon in an industrial setting. This can be clearly observed by the results of the survey conducted by Martini et al. (2018), where only 7.2% of respondents reported having a systematic approach to TD tracking.

1.2. TD conceptual models

There is a rich body of work on how technical debt and various concepts related to it should be modeled, called, and described. An overall conceptual model, which has been used as a basis for other research, was presented by Avgeriou et al. (2016). The tertiary studies on TDM by Rios et al. (2018) and by Junior and Travassos (2022) both provide extensions to the conceptual model regarding TD activities, TD types, and strategies & technologies. A similar model, designed to describe architectural debt, has been recently proposed in the work of Verdecchia et al. (2021).

1.3. TD types

Various works tackle the problem of describing possible TD types, depending on what part of the software or aspect of its development the TD item influenced. In their tertiary studies, Rios et al. (2018) found 15 different TD types, while Junior and Travassos (2022) aligned the types with the software development phases. Furthermore, new types like security debt (Ahmadjee et al., 2021) are still emerging. Moreover, TD type names are not always consistent across different works. For example, "People debt" defined Rios et al. (2018) is a very similar concept to "Management and Social Debt" defined by Ernst et al. (2021).

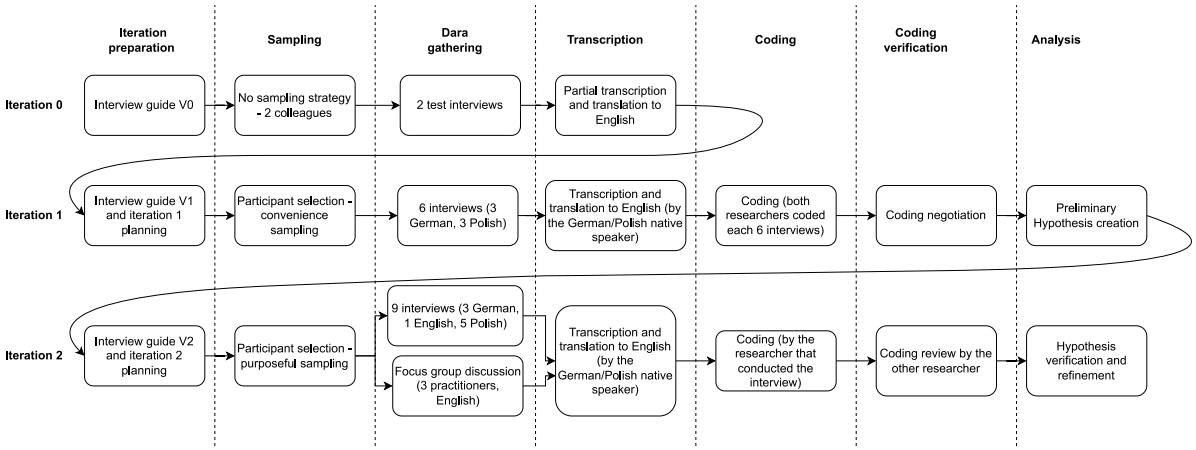


Fig. 1. Study design.

1.4. Technical debt causes and consequences

The main causes and consequences of TD have been extensively described in previous research, particularly by the InsighTD project (Anon, 2022a; Rios et al., 2020; Ramač et al., 2021), where a unidirectional model of TD causes and consequences was presented.

Notably, causes associated with management, dominate many lists of TD causes. Rios et al. (2019) found that in 34% of the TD occurrences, TD is incurred as a result of a cause from the "Planning and management" category. Ramač et al. (2021) carried out an extensive survey that found 725 "Planning and Management" cause occurrences, with the second most populated category being "Development Issues" (473 occurrences). Vogel-Heuser and Bi (2021) identified that 32% of their researched TD items were initiated by management decisions.

Regarding the consequences, Kruchten et al. expanded the unidirectional model by dividing the effects into consequences (direct results of the TD item) and symptoms (the visible impact of the TD item) (Kruchten et al., 2019). This idea is rooted in works on TD conceptual models (Avgeriou et al., 2016; Verdecchia et al., 2021). However, Stochel et al. (2020) identified that a *"smell may be considered as a symptom of a TD Item, rather than the TD Item itself"*., which contradicts this idea of visibility, at least on a business level.

1.5. Managers' perspective on TDM

A particularly striking detail from the work of Rios et al. (2020) is that participants openly stated that their project managers are ineffective when it comes to managing TD. One participant even bluntly states that project managers (and customers) simply rush to "receive something working as soon as possible" (Rios et al., 2020). Additionally, Soliman et al. (2021) show how communication between the development team and management greatly impacts debt incurring architectural design decisions. What is more, in their case study, Wiese et al. (2022) describe the impact a TDM process may have on IT managers in terms of TD overview, managing the project pipeline, and communication with their business stakeholders.

2. Method

The aim of our study was an in-depth exploratory investigation of the IT managers' perspective on TD based on their real-world experiences. Thus, we decided on an inductive (Runeson et al., 2012) approach for the study design, which follows an "iterative observational research path" (Stol and Fitzgerald, 2015). Additionally, in order to heighten the relevance of our findings (Jain et al., 2013), we flexibly adjusted each of the three iterations of the study. The overall study design is presented in Fig. 1.

Each of the three iterations had a different goal:

- **Iteration 0** served as a test of our data gathering method. We tested the interview guide, discussed and adjusted the overall study design by conducting two test interviews. The data from these interviews was not used for further analysis.
- **Iteration 1**, during which we performed 6 interviews in total, was supposed to enable the creation of preliminary findings and the creation of a stable code list that would be used for further analysis.
- **Iteration 2** was supposed to provide enough data to either prove or disapprove the preliminary findings and allow us to refine our findings.

We concluded the study after these iterations since we achieved code saturation (Saunders et al., 2018) as described in Section 2.1.

The study iterations (with the exception of the simplified Iteration 0) consisted of the following six steps, which were performed slightly differently to meet each iteration's goal:

(1) Iteration preparation: We discussed the results from the previous iteration and adjusted the plan for the upcoming one. During the preparation step, the authors decided on the sampling strategy, revised the interview guide, and planned the coding and analysis strategy. *(2) Sampling:* We recruited participants for our study, depending on the sampling strategy chosen for the iteration (see Section 2.1). The procedure of gathering data through interviews is described in Section 2.2.1, and the focus group discussion is explained in detail in Section 2.2.2. *(3) Transcription:* Since data was gathered in 3 languages (English, German and Polish), the transcriptions of the interviews were translated to English. The translation process was always performed by the German or Polish native-speaking author to avoid any misunderstandings. *(4) Coding:* We coded the transcripts according to the coding process explained in Section 2.3. *(5) Coding verification:* We verified and refined the coding as presented in Section 2.3. *(6) Analysis:* We discussed the results of the coding and memoing. More details are given in Section 2.3.

2.1. Sample

Before gathering participants, we first defined a set of roles to better specify the targeted participants and differentiate them from other roles:

Table 1
Study participants.

Participant	Company domain	Company size (no. of employees)	Team size (no. of employees)	Years working in IT	Years as an IT manager	Gender	Top manager	Former manager
I1	Finances	100–1000	25–100	>10	5–10	m	x	
I2	Hardware & Software Engineering	1000–5000	0–10	>10	2–5	m		
I3	Online Retail Services	1000–5000	0–10	>10	>10	m		
I4	Software Engineering	0–100	0–10	>10	>10	m	x	
I5	Media	1000–5000	>100	>10	>10	m	x	
I6	Human Resource Services	0–100	10–25	>10	>10	m	x	
I7	Food industry	> 5000	0–10	>10	>10	m		
I8	Software Engineering	0–100	0–10	>10	5–10	m	x	
I9	Finance	1000–5000	25–100	>10	>10	m		
I10	Logistics + Finance	100–1000	25–100	>10	>10	m	x	
I11	Telecommunications	> 5000	>100	>10	2–5	f		
I12	Human Resource Services	0–100	0–10	>10	0–2	m		
I13	Fast Moving Consumer Goods	> 5000	0–10	5–10	0–2	f		
I14	Entertainment	1000–5000	10–25	5–10	2–5	f		x
I15	Online Media	100–1000	0–10	>10	5–10	m		x
I16	Entertainment	1000–5000	10–25	2–5	0–2	m		
F1	Social Media	1000–5000	0–10	>10	5–10	f		
F2	Social Media	1000–5000	0–10	>10	2–5	m		
F3	Social Media	1000–5000	0–10	0–2	0–2	m		

Table 2
New codes with each transcript.

Interview No.	I1	I2	I3	I4	I5	I6	Focus group	I7	I8	I9	I10	I11	I12	I13	I14	I15	I16
No. of codes	52	26	22	25	30	46	16	38	8	16	18	22	9	13	8	4	2

IT managers have personnel responsibility for their development teams, i.e., they are IT line managers, and can be on different levels of the company's hierarchy. *Top managers* are IT managers that are either their companies' CTOs or are more than one hierarchy level above the developers.

As the *development team or developers*, we summarize all technical roles that do not have personnel responsibilities but solely serve the software development process, e.g., developers, project managers, product owners, and testers.

Business stakeholders can be business managers or customers. They both have the same roles regarding TDM and are only called differently depending on whether they are a part of the developing company or not.

As suggested by Baltes and Ralph (2022), we defined the goal of our sampling strategy before starting data gathering. We decided that our goal was achieving code saturation (Saunders et al., 2018), which is the standard way to assess that further data collection is not necessary.

To answer our questions appropriately, we sought IT managers that had to be responsible for scheduling projects, or at least tickets, or they had to directly communicate with the business stakeholders. This resulted in 19 IT managers; 14 of them had all required properties, eleven had additional budget responsibilities, and six were top managers. The IT managers worked in various domains, as shown in Table 1. Seven interviewees worked for German and nine for Polish companies. All three focus group participants were employed in a single German company.

Since iteration 0 was not supposed to be used for data gathering, we simply performed the test interviews with two practitioners experienced in software development.

During iteration 1, we gathered participants through convenience sampling (Baltes and Ralph, 2022), i.e., through our personal networks. Since the iteration's purpose was to create preliminary hypotheses and establish a common coding approach, we gathered participants from very different domains in order to obtain a diverse set of data. At this point, we did not take other participant characteristics into account.

For iteration 2, we decided on purposeful sampling (Baltes and Ralph, 2022) to find all possible codes, which would allow us to get to the saturation point. Having analyzed the data from iteration 1, we made observations about the types of participants we were missing, such as participants with less than five years of experience, from a creative domain, from a company with more than 5000 employees, ex-managers, and female managers. We purposefully sought participants that would meet these criteria for this iteration.

We counted the appearances of new codes after coding each transcript (see Table 2). During the coding of the last transcript, only 2 codes (of 393) emerged, which is under 0.5% of all codes. Thus, we concluded that saturation was achieved.

All participants filled out a consent form and a questionnaire about themselves (Wiese and Borowa, 2022) before participating. An overview of all participants is presented in Table 1.

2.2. Data collection

We collected our data through leading sixteen interviews and one three-person focus group discussion that focused on our preliminary findings. In the following subsections, we give detailed information on these two data gathering methods.

2.2.1. Interviews

Table 3 shows an overview of all interview questions (IQ) and their relation to the RQs. The interviews were performed in the form of semi-structured interviews, which means that slight variations from the overall procedure (like asking follow-up questions) were allowed.

Since we found that some key topics were missing in iteration 1, some of the questions were added during iteration 2 to maximize the relevance of our findings (Jain et al., 2013), Those additional questions and the rationale behind them are:

Question 1.a.: Participants did not use the financial metaphor , but instead often used their own metaphors. We wanted to explore the metaphor's usefulness.

Question 6: IT managers mostly talked about the existing TD and not much about incurring TD intentionally. We wanted to explore incurring TD intentionally further.

Question 7. and 7.a: IT managers did not explicitly mention prevention mechanisms, which we wanted to explore as well.

Table 3
Interview questions.

IQ No.	Question	Related RQ
1.	What is TD to you?	RQ1
1.a	Does the metaphor help to understand the concept?	RQ1
2.	How does your team/group manage TD?	RQ3
2.a.	Do you have an overview of your team's TD?	RQ2, RQ3
3.	What do you do to help manage/enable the management of TD?	RQ3
3.a	What is your part/role in managing TD?	RQ4
4.	What TD items does your team/group have - could you give some examples?	RQ1
5.	From these TD items, which one do you consider the most impactful?	RQ1, RQ2
5.a	How/Why did this [item] happen?	RQ2
5.b	What influence/consequence does this TD item have? (What is the problem with this TD item?)	RQ1
5.c	Did you have any influence for it to be incurred?	RQ1, RQ3
5.d	Do/Did you have any influence for it to be removed?	RQ1, RQ3
5.e	What could have been done better when dealing with this TD item?	RQ 3
6.	At which point in time do you get to know of TD? (While incurring them or later?)	RQ1, RQ2
6.a	What would be the advantages (or disadvantages) of early recognition?	RQ1, RQ2
7.	Do you think TD can be prevented?	RQ2, RQ3
7.a	Do you know of TD items that could have been prevented? (Which ones? What are the preconditions for this?)	RQ2, RQ3
8.	From your perspective, is it important and worthwhile to manage TD? And why?	RQ2
9.	How could you, as a manager, benefit from TD management?	RQ2
10.	In your opinion, as a manager, what are the drawbacks of TD management?	RQ2
11.	What do you think you could do (optimize) to manage TD (better)?	RQ3
11.a	What do you need from your developers to better manage TD?	RQ3
11.b	What do you need from your upper management or customer to better manage TD?	RQ3
12.	For former managers:	
12.a	When have you stopped being a manager?	
12.b	Why did you decide to change your role?	
13.	Is there anything you would like to add?	

Questions 11.a and 11.b: IT managers often blamed their environment for the lack of an effective TDM process and a growing amount of TD items. To identify TDM's potential improvements, we asked them about their preferred environment for better TDM.

Question 12: To overcome the survivorship bias, we included former managers, who have resigned from their positions, during iteration 2. Question 12 was used to obtain additional information about them.

2.2.2. Focus group discussion

We used the focus group discussion to provide method triangulation (Runeson et al., 2012), which reduces the bias of gathering data through only one method. To perform method triangulation, data has to be gathered through different methods. In the case of the focus group discussion, we presented our preliminary hypotheses, which were the product of the iteration 1 analysis, to a group of three IT managers from one company. During the discussion, we asked them to agree or disagree with the findings and to discuss them with each other. The following preliminary findings were covered:

- The IT managers are concerned about software aging. This topic also comprised the problems regarding iterative TD repayment and the impact this topic has on old-established companies,
- The IT management evolved over the last decades and the possible next step would be to adapt TDM. This concern included TDM issues in agile environments and the need for a responsible person to keep track of TD.
- Communication is a major source of problems regarding TDM with reference to all stakeholders. We discussed the communication with business stakeholders, including the suitability of the metaphor and causes for the business stakeholders' "it works" mentality.
- The mindsets regarding decision-making need to change to avoid TD, and important decisions must be: (1) recognized as such, (2) made consciously, (3) made by groups, (4) based on a pragmatic evaluation through a comparison of options.

The slides used for the focus group discussion are available as part of the additional material (Wiese and Borowa, 2022).

2.3. Data analysis

We used "The Coding Manual" from Saldaña (2013) for orientation on the optimal coding procedure and the MAXQDA tool (Anon, 2022b) to perform the coding.

The coding techniques that we used (all of them originally described in Saldaña (2013)) included:

- Structural Coding: marking a segment of the transcript as one containing data about a certain topic. For example, we used the code "TD knowledge - causes" to mark segments where participants mentioned specific information about any TD causes. The Structural Codes are the only codes that were planned before the analysis started. All of them were related to the RQs.
- Process Coding: Process coding was used to mark segments of the transcripts that were related to specific actions associated with TDM. All process codes contained a verb in the gerund form ("ing"). For example, "Incurring TD" was used to mark the part of the interview when IT managers talked about the specifics of how a TD item was incurred.
- Value Coding: Value codes were used to mark participants' values, attitudes, and beliefs about TD and TDM. These codes had names that were always preceded by "V:" for Value, "A:" for Attitude, or "B:" for Belief. For example, "V: TD(M) is important topic" was a code used to mark a segment where the participant stated that they believe that the topic of managing TD is of importance to their work.
- Descriptive Coding: Using codes that summarize the data from a particular segment. For example, the code "Good Developers are leaving/harder to hire" was used to mark the part of the transcripts where participants' mentioned that developers from their teams left their company because of TD, and it may be hard to hire new employees thereafter.

During the coding process, we constantly reviewed, extended, and reorganized the coding list. Codes were usually organized in a structure of sub-codes, with the structural codes as the main parent codes. For example, in "Structural \TD knowledge - consequences \Business consequences \visible - Business problems \Losing the customer's trust" the code "Losing the customer's trust" is used to mark data segments where participants mention how they lost the customer's trust - which was a business consequence visible to the business stakeholders.

Additionally, at any point in the study, we created analytic memos, which are all available in the additional material (Wiese

and Borowa, 2022). Memos are notes related to particular parts of the data (like quotes from the transcript, codes, groups of codes, or notes from the interviews) that contain various observations from the researcher performing the analysis. They are often a source for working hypotheses that can be explored later during the study. For example, the memo titled "The TD metaphor is not used/filled out" was related to the observation that the TD metaphor may not be widely used and understood.

For **iteration 1** coding, we used all of the aforementioned coding methods.

We used the coding guidelines of Campbell et al. (2013) to establish a common coding scheme, which allows a single coder to perform the analysis in the next iteration.

We solved the "Unitization Problem With Units of Meaning" (Campbell et al., 2013) (problems with the mismatch of the length of the coded text) by deciding that the segments that we will be coding should always be blocks between punctuation marks.

We applied the negotiated agreement approach (Garrison et al., 2006) to resolve the "Discriminant Capability Problem" (Campbell et al., 2013) (discrepancies in coding by different coders). This means that all six iteration 1 interviews were coded by both authors separately. Subsequently, all of the differences in the coding were discussed until a full consensus was reached. The organization of the sub-codes was an additional crucial part of these negotiations.

Furthermore, the iteration 1 coding and memoing allowed us to establish a set of preliminary findings and hypotheses, which we focused on during iteration 2.

For **iteration 2**, the interviews were coded by the researcher who led the interview to avoid misunderstandings. To reduce researcher bias, the second author reviewed the coded transcripts. During this stage, we, particularly, gathered data that filled the gaps in our coding scheme, which enabled us to fully develop our hypotheses and reach code saturation.

When exploring the causes and consequences of TD items, we used the conditional/consequential matrix described by Corbin and Strauss (2014), which is a method used to analyze the interplay between a process and its conditions/consequences on various levels (from the level of an individual person, through a group they are part of, to an international one). This allowed us to view the causes and consequences of TD incurrence in the context of their visibility for different stakeholders.

3. Results

In this section, we present the results of our research questions. We give a detailed overview of the IT managers' existing knowledge about TD in Section 3.1, present the IT managers' main concerns in Section 3.3, and summarize the potential improvements as perceived by the IT managers in Section 3.4.

3.1. RQ1.1: Managers' knowledge about technical debt

To answer our first question about the IT managers' current knowledge of TD, we summarize their knowledge regarding the Definition of the TD term and their perception of the metaphor in Sections 3.1.1 and 3.1.2. After that, we identify the mentioned TD types in Section 3.1.3 and give an overview of the IT managers' team's maturity/awareness levels in Section 3.2. We continue by identifying the elements of the TD conceptual model (Avgeriou et al., 2016) that IT managers are aware of, and the relationships between causes and consequences they describe in their examples in Section 3.1.4. In Section 3.1.5, we introduce the concept of TD's visibility for different stakeholders. Finally, we give a detailed description of the identified TD cause and TD consequence categories in Sections 3.1.6 and 3.1.7.

3.1.1. Definition

Most IT managers (17 out of 19), were familiar with the concept of TD. One participant believed it to be a colloquialism with no specific definition, and only one IT manager had never heard the term before. IT managers used the following key concepts to define the term TD (number of IT managers in brackets):

- TD consequences (15), e.g., *"[It] means that systems … are not flexible enough to be further developed, iterated on, added new features or additional functionality to".*
- TD causes (12), e.g., *"[I]it can happen that we - but this happens only rarely - we purposely develop something faster from the time perspective".*
- TD short-term benefits versus long-term consequences (9), e.g., *" And then we discussed various options and there was always a fast track. But maybe that [fast] solution is not very good maintainable".*
- Process descriptions (incurring, repaying) (4), e.g., *"[F]or me it means, in general, code that we want to refactor…".*
- TD types (4), e.g., *"[S]uch a debt can manifest itself in different places, be it at the level of the application's source code or at the level of some architectural mistakes".*
- TD as old or obsolete technology (3), e.g., *"So by technical debt I mean something …, which from an IT perspective is obsolete technology in the sense of: I don't have know-how anymore".*
- Financial metaphor (1), e.g., *"Technical debt, that's kind of an analogy, a little bit to debt that you take on in the financial world".*

3.1.2. Metaphor

Unprompted, only one IT manager mentioned the financial metaphor while defining the term TD. However, IT managers often used other metaphors when explaining the TD concepts:

- A progressing knee disease: TD was compared to a disease that initially does not cause serious problems but, if left untreated, may result in severe consequences such as the need for limb amputation (abandoning the project altogether) i.e., *"[Y]ou can, of course, have your knee hurt all the time and you don't go to the doctor because you think all the time, it's just a little thing, ... And one year after the other goes by, and at some point, your hip hurts because you have a bad posture . . .. And at some point, you go to the doctor and he says: Yes, the only thing that helps is a prosthesis".*
- Tailoring bespoke clothing: The piece of clothing was compared to a system, and various changes made according to the client's demands (incurring TD was a byproduct of these changes) gradually caused the garment to deteriorate i.e., *"When we plan a software, a digitalization project, then we tailor a piece of clothing. And at the end, when the garment is finished, then comes real life. And then . . . we have to leave it out at the hem, then we have to knit cuffs on it, then he [the client] wants to have another pocket. And after a while, it turns into not such a nice garment anymore".*
- Building cathedrals: A system was compared to a cathedral. After the construction of a cathedral, the workers responsible for it disappear with the knowledge about how the cathedral was built. This is a metaphor of Documentation TD and knowledge vaporization, i.e., *"Software and cathedrals are much the same: First, we build them, then we pray. That actually sums up the typical project process quite well. . .. And basically, it is like if you stay in the picture, the cathedral builder of the Cologne Cathedral, who ultimately only tackles it when something starts to crumble off somewhere".*

- Not fitting puzzle pieces: This describes an instance of TD where various system components were developed independently, which caused a problem during an integration process, i.e., *"I can think of more like geometric metaphors, . . . that somewhere there is a mismatch, which is based on the fact that the elements do not fit together"*.
- A freight train: The train was compared to a system that required constant maintenance to run properly. One manager explained this in detail after their interview. This IT manager negotiated TD repayment with a business stakeholder by comparing this to the maintenance costs of a freight train. He was asked by the business stakeholder: *"But the IT doesn't break, why do you need maintenance?"* The IT manager explained: *"We do even worse! We open the engine and operate directly – hands-on – in the engine, and we do this without an operations manual, which you usually have in detail for freight trains"*.
- House building: A house was equaled to an IT system that must be built in a specific way, according to engineering best practices. If a load-bearing wall is not built (incurring TD by making the process faster and cheaper), then a door cannot be mounted (TD causes development to stop) i.e., *"What you can explain to them is, if I don't install a door in a house, then I just can't get in . . .. If I want to have the door somewhere else, well, then I'll just come and mill it in . . .. [N]ow imagine that you've put a load-bearing wall there. The load-bearing wall has to go somewhere. . .. [T]he colleagues from the business are – partly at least – not technically versed, they don't think of something like a T-beam."*.
- Quicksands: A spiral of incurring more and more TD was compared to the deathly danger of being devoured by quicksand, i.e., *"So if we see this technical debt, we notice earlier that what we're doing is only going to get worse, worse, worse, then maybe we can say at an earlier stage - So let's just redesign the feature. Let's just make it from scratch, instead of wading further into these quicksands"*.

Considering this wide variety of metaphors, we additionally asked about the usefulness of the term TD as a metaphor during the second iteration of data gathering (see Section 2.2.1). Most IT managers interpreted the metaphor's or its usefulness in a **negative way**. They stated that *"this analogy points to something negative"* and is not well understood, particularly by business stakeholders. One IT manager compares it to having *" a corpse in the closet"*. Regarding the communication with business stakeholders, one IT manager explains that *"most people in the business can't really imagine what TD is, even though the metaphor itself is actually very apt . .. I think if that were better understood, then we wouldn't have the problem at all"*.

Fewer saw the metaphor in a **positive light** and mentioned that the metaphor helps to explain or understand the TD concept, and recalls the growing interest. One IT manager stated that *"the metaphor with the debt and with the interest and so on is really useful for explanations"*.

Some explained the metaphor is already **widely used in the industry** and therefore a good term, e.g., *"[F]or me it's part of the language of IT specialists or project managers, so for me it's obvious"*. One IT manager notes that *"this term has become established. And it is really used at every corner"*.

3.1.3. Technical debt types

During the interviews, we asked the IT managers for two to three examples of TD items, and then to choose the most impactful one for detailed questioning. We summarized the number of examples and mentions of various TD types and the number of IT managers mentioning them to get an idea, of which types of TD items are most relevant for IT managers.

We noticed, that IT managers frequently mentioned problems regarding the optimal point in time for an update, e.g., *"You then get into a situation where you're collecting a technical debt because you missed the point in time where you can just upgrade things easily"*. Even the infrastructure's version can become a problem, as pointed out by one IT manager: *"And recently we had such problems that we ran out of spare parts, for example, because part of the software was based on computer drivers, which simply could not be obtained anymore"*. Ernst et al. (2021) describe a similar concept on the implementation (code) level but not on an architecture level, contrary to most IT managers. As we could not find a related type on architecture level in literature, we identified **Updating TD** as a specific type of ATD that occurs when third-party components or systems are not updated on time.

Overall, we identified the following occurrences of TD types (number of examples/number of IT managers) :

- Architectural TD (55/6), e.g., if *"a lot of logic was transferred to the database level and, at some point, it turned out that there was quite a lot of traffic in the system and it generally started to lag this system"*.
- Updating TD (21/9), e.g., one IT manager's team was *"more or less surprised by the fact that the support for this component was virtually discontinued by the 3rd party and we just had to react.*
- Code TD (22/10), e.g., *"when you hire new people, young people, for example, where they . . . write some code fragment that is not optimal"*.
- Documentation TD (14/6), e.g., *"somehow they forgot to create . . . proper and orderly documentation, which in the end leads to the fact that there is exactly one developer who has an overview of the entire system"*.
- Requirement TD (13/6), e.g., one IT manager explained regarding a tool that could only deal with one-page forms: *"It was basically, well short-sightedness in terms of the direction of the requirements changes, because you could easily predict that the forms . . . can be multi-page"*.
- Infrastructure TD (10/7), e.g., *" VMWare version 5, still used Flash . . . for the web client. Adobe has ultimately replaced or discontinued Flash and, thus, the accessibility of such VSphere stuff is quite difficult, because you first have to find an old system where Flash Player is still on it"*.
- Test TD (5/4), e.g., was mentioned a few times by the IT managers when explaining what TD is, but no concrete examples were given.
- Security TD (1/1) is mentioned by one IT manager: *"In this case, such situations may even end catastrophically – some leaks of personal data. They can be fatal for very large companies"*.
- Deployment TD (1/1) is mentioned by one IT manager regarding a distributed system: *"[T]his is the process of introducing changes, which requires issuing many orders to many different organizations in order to transfer the change to the production environment"*.

This shows that IT managers are aware of the variety of TD types and the fact that TD is not just a topic for developers related to code.

Among the most impactful items were eleven ATD items, two updating debt items, two requirement debt items, and one code debt item. This means that an overwhelming majority of TD items chosen by the participants were examples of ATD.

3.1.4. Technical debt conceptual model

Comparing our study's results with the conceptual model of TD as proposed by Avgeriou et al. (2016), we identified information about TD items, types, causes, and consequences (incl.

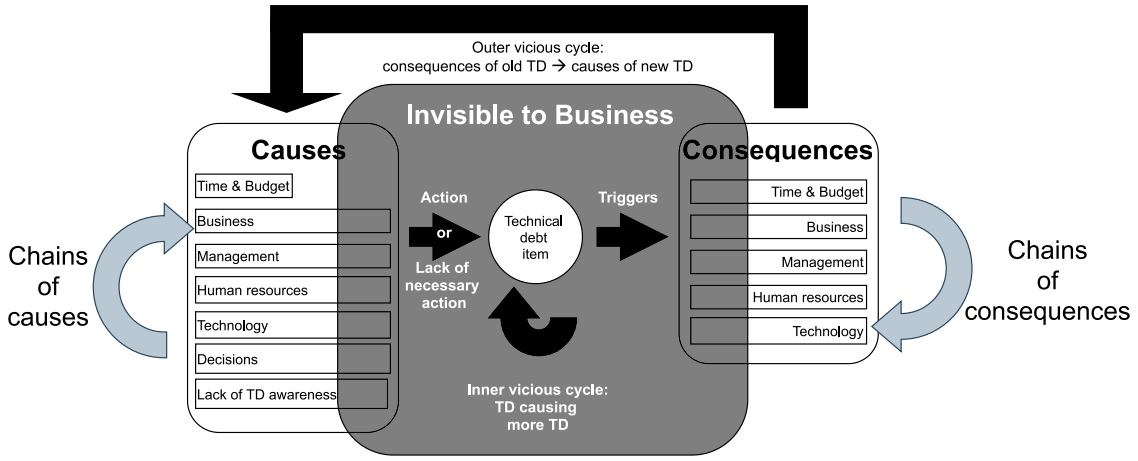


Fig. 2. V4CTD model of *Visibility, Cycles & Chains of Causes & Consequences of TD*.
(gray box shows concepts invisible to business stakeholders, i.e., TD items and part of causes/consequences).

symptoms and business goal influences from Avgeriou et al., 2016) in the transcripts. However, the idea of a unidirectional concept of TD causes leading to a TD item and the TD item leading to TD consequences and symptoms, as in the works presented in Section 1.2, did not perfectly fit our study's results.

We, therefore, suggest extending the conceptual model of TD, with regards to the causes and consequences, as summarized in our **V4CTD model** of *Visibility, Cycles & Chains of Causes & Consequences of TD* in Fig. 2. Below we explain the additions and provide one detailed example for each of them.

Chains of Causes. Various TD causes can trigger other causes. When explaining why a specific TD item was incurred, one manager explicitly stated: *"The problem, as usual, I think there are many layers, reasons"*.

In the following example, time pressure is one cause, that leads to outdated documentation, which is a (secondary) cause for knowledge vaporization. *"Certainly, there was documentation at the beginning, but then it didn't continue for various reasons. And the usual reasons are always time pressure. People leave the company, new people come in. Something serious has been changed in the architecture. No one really understood how to document it, and so on"*.

Chains of Consequences. Various TD consequences can trigger other consequences. This concept is similar to the one of "consequences and symptoms", as described by Avgeriou et al. (2016) and Kruchten et al. (2019), where invisible consequences lead to visible symptoms. However, the V4CTD models' *chain of consequences* includes the possibility of a visible symptom triggering an invisible consequence, e.g., losing a customer may cause stress and decreased morale in the development team.

A chain of consequences where technical consequences lead to business consequences is reported by one IT manager: *"But if you've started to build a complex product internally, where you've got bottlenecks everywhere in the development, then you're not only hindering your software development and driving up costs and time, but you're also actively hindering the further development of the business"*.

Inner Vicious Cycles. The TD items themselves may be the cause of incurring additional TD. This concept was already mentioned as TD vicious cycles, e.g., by Ramač et al. (2021), but evidence or detailed information for this is lacking. However, our participants described such situations: *"So they [the current staff] learned and they already knew, but when new people were introduced into the project, . . . they could get lost [in the code], like: 'Hey, what should I do here, because I don't have this option to choose from?'"* In this case, the missing documentation, which is one TD item, leads to unreadable code, which is another TD item.

Outer Vicious Cycles. The consequences of one TD item may become a cause for incurring more TD. Martini and Bosch (2015) previously described this as a TD vicious cycle. They reported the cycles of specific TD items but did not conceptualize their insights. Similarly, our participants identified cycles where TD consequences led to other TD causes and created an outer vicious cycle, e.g.: *" [W]e would need to redo everything from scratch to make it [a functional change] the proper way. And then there comes a discussion like: 'Do we even have the capacity right now and time in order to implement this stuff?' And in some bad cases, it ends up with another quick and dirty thing on top of already existing, working dirty [things]"* In this case, the consequence of one TD item is the need for a rewrite before another change. Avoiding this rewrite is then the cause for more TD to be incurred.

Moreover, by combining the different descriptions of TD items provided by IT managers, we uncovered that similar issues were often mentioned as either TD causes or TD consequences. For example, dissatisfied and stressed developers leaving the company is a TD consequence. However, a lack of developers can be the cause of TD. Likewise, a consequence on the management level is projects becoming unmanageable when TD has to be repaid unplanned. Bad project management, however, is also a cause for TD. Similarly, time and budget were frequently named as cause and as consequence. On a more abstract level, business stakeholders might cause TD, e.g., by setting tight deadlines. In subsequent iterations, not all required features may be implemented, or the next deadline may not be met due to the interest that has to be paid for that former TD incurrence.

3.1.5. Visibility of causes and consequences

Another result of the study, which has been incorporated into the V4CTD model, is the observation that not every stakeholder may have an in-depth insight into the specific mechanics of every TD item.

IT managers cannot be aware of every detail of the system's development, i.e., they may not know about or understand every TD item that is incurred and the causes and consequences related to this item. Some managers pointed this out when being asked about their influence on the incurrence of their example TD item. For one of the former managers, this had been one of the reasons to quit his position: *" [T]he longer you are in that [IT manager] position, the further away you are from the development and from the awareness that, for example, technical debts have to be reduced continuously"*. This lack of visibility may be the cause

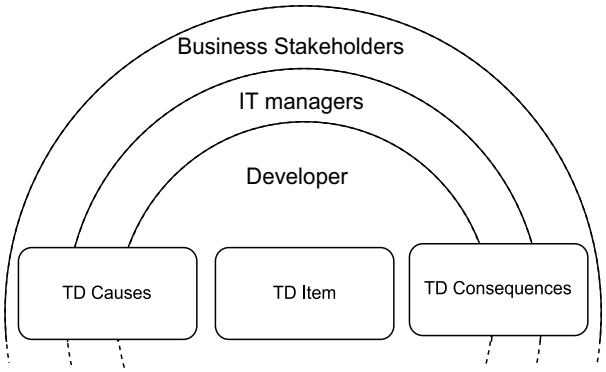


Fig. 3. Visibility and influence cycles of stakeholders.

of faulty decisions, as an IT manager explained: *"And as I said, the conditions were not clear on the table [, i.e., openly available/visible to him]. . .. And that's why you suddenly decide differently than you perhaps should have decided, retrospectively, from today's point of view"*.

Analogous, this is, even more, the case for **business stakeholders**, since their understanding of the technical subject is limited and they are less involved in the technical decisions. One top manager pointed out: *"They [business stakeholders] are not professionals. [They say:] 'Why? I bought the software; it still works. That the environment changes continuously with each update, it is not at all clear to them"*. However, business stakeholders are well versed in the field of business goals, which are impacted by the consequences as described in the conceptual model of Avgeriou et al. (2016). Additionally, they also influence the causes, i.e., their decisions may be the cause for a cause as a part of a chain of causes.

In turn, for the **developers**, the rationale for the business stakeholders' or IT managers' debt-incurring decisions may not be visible anymore.

We used the conditional matrix (Corbin and Strauss, 2014) describing activity circles of different stakeholders around an action as an inspiration to structure the causes and consequences into visibility circles for each stakeholder, as summarized in Fig. 3.

- Developers may be aware of the TD item itself and its direct causes and consequences, but they may have limited insights into debt-incurring business decisions.
- Managers may be able to understand the causes and consequences of TD, but may not understand every TD item in detail. Additionally, they have a deeper insight into business goals, and, as such, may see the causes behind causes or the consequences behind consequences.
- Business stakeholders may not understand the technical side of TD, but they can understand and influence some of the causes and suffer from some of the consequences.

In our V4CTD model, we grouped TD causes and consequences into visible and not visible causes and consequences with regard to business stakeholders. In the case of **visible causes**, the root cause is a decision made by business stakeholders, or they may be able to understand the cause. For **visible consequences**, business stakeholders may observe the consequence because the business is directly impacted. The summary of causes, consequences, and their visibility is shown in Fig. 4 and described in detail in the following sections.

3.1.6. Technical debt causes

Through our analysis, we identified the following eight categories of TD causes (number of mentions in brackets): [Faulty] decisions (138), Management causes (60), Business causes (59), Technological causes (43), Limited time (40), Human resource causes (31), Limited budget (16), and Lack of TD Awareness (8). Below, we explain each category and give one example of visible and invisible causes for each category, respectively. An overview is given in Fig. 4 and more details can be found in the additional material (Wiese and Borowa, 2022).

[Faulty] decisions. These decisions include both conscious rational ones, as well as mistakes and sub-optimal or even reckless ones. We observed that decisions were made by different stakeholders:

- Developers make decisions and possibly make mistakes or write unreadable code when implementing algorithms or functionalities.
- IT managers might, for instance, decide on a specific team structure, postpone decisions, or make mistakes during requirements gathering.
- Architectural decision-making may lead to TD, e.g., when not all options were compared, options were missing in former times, or knowledge about an option is lacking.
- A supplier might, for example, decide to discontinue the support for a component, which creates the need to change the running system.
- Business managers, e.g., might decide to set a certain deadline without considering the actual time and effort required or make inconsiderate architectural decisions.

(**Visible:**) Business managers making technological decisions without consulting the IT department are shown in this example: (**Invisible:**) IT managers often mentioned that they did not supervise the developers effectively and did not establish efficient peer reviews: *"[T]his person was getting feedback that he had to write better [code], but it took a long time [until this happened]. So I would say that it was my mistake as a manager that I left that person alone"*.

Management causes. These are causes related to IT management, i.e., mostly social or organizational issues that lead to incurring or increasing TD unnecessarily. (**Visible:**) This category includes missing processes in earlier times, i.e., the evolution of IT management: *"So in IT, certain best practices have become established . . . in managing products, in developing software, in managing infrastructure. There are entire process libraries"*. (**Invisible:**) Miscommunication issues, e.g., between business and IT stakeholders, is another cause belonging to this category, like *"The developer says: 'Of course, I can develop this for 50 days, – comma if – . . . if I then still have time to take care of refactorings and documentation.' But he never uttered that [comma if] sentence. Nobody ever understood that / heard that, and then nobody scheduled it, and it didn't happen"*.

Business causes. These causes are directly or indirectly related to business decisions or knowledge. One commonly known business cause is the need to reduce the time to market and pre-set deadlines by the customer. (**Visible:**) Causes in this category also include the change in business strategy, or a complex and growing business model, e.g., in a start-up situation. One manager mentioned with regards to this category: *"At some point, it reached its limits when we became international, with all the different languages, formatting of numbers, currencies and so on"*. (**Invisible:**) A more indirect cause is the lack of knowledge of the requirements, e.g.: *"We started preparing a project with that, like this. And we realized just in time that we were in the process of building up technical debt again because then the [new] idea came up that you could also use this smart card to track work processes"*.

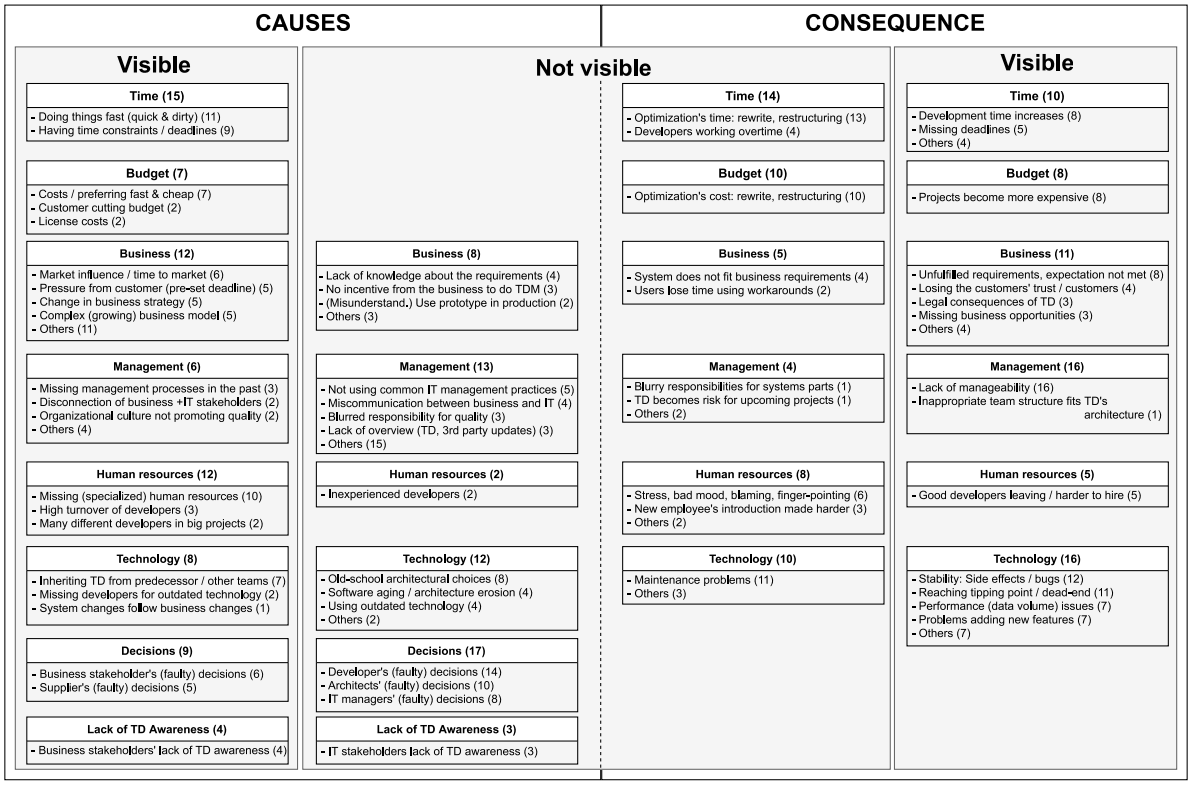


Fig. 4. TD causes & consequences visible and not visible to business stakeholders (Number of managers mentioning this topic in brackets). For details on "Others" see additional materials.

Technological causes. This category comprises causes related to the evolution of various technologies and to growing knowledge, e.g., regarding best practices and solution architectures. (**Visible:**) Many IT managers report inheriting TD from previous projects, e.g., *"the problem is that these systems in our company, they live a long time and the first versions were created when my person was crawling [as a baby] there and I did not have too much influence on what was happening here"*. This is usually understood as a problem by the business stakeholders, e.g., if two companies merge, problems are expected. (**Invisible:**) This category also includes the reports of missing technology in earlier times: *"[I]f we now go back a bit to the 90s, then there was this whole issue of identity management. There were no real solutions, or only very, very expensive solutions"*.

Time-related causes. This is another category clearly influencing TD. While time pressure is often caused by business decisions, as explained earlier, this category additionally comprises other time-related causes. IT stakeholders perceive overwhelming time pressure in the IT industry, which could lead to always choosing the fastest solutions even when there is no direct time pressure. (**Visible:**) Various examples of this overwhelming time pressure in the IT industry were given in statements like *"there just wasn't time to do anything else"*, or *"time is much more important"*, (**Invisible:**) We did not find invisible causes in this category as time constraints can usually easily be explained to business stakeholders.

Human resource causes. This category encompasses an overall lack of IT specialists, the absence of specialized or knowledgeable developers, and the high turn-over rate many companies suffer from. (**Visible:**) An example of specialists leaving the company was: *"We can also say that it [a TD item] is maybe due to a shortage of people in the DevOps department. Because we used to have two guys on the team that had different responsibilities. Now we are left with only one"*. (**Invisible:**) A lack of developers also leads to a situation where inexperienced developers implement substandard solutions. The following is explained by an IT manager who used to be a developer himself: *"[F]rom my own evolution, my own view of these things, I see that experience makes you appreciate certain aspects that in the short term are not important"*.

Budget-related causes. This category comprises funding–related causes like cutting a project's budget, license costs, or the general preference for cheap and fast solutions. (**Visible:**) One IT manager gives an example of a customer's statement: *"I simply can't afford it now to change everything from some Windows Forms application to some new web design"*. (**Invisible:**) We did not find invisible causes in this category as budget constraints are commonly discussed with business managers.

Lack of TD awareness. This category includes causes related to various decision-makers being unaware of TD or of the consequences a TD item might have. (**Visible:**) A focus group participant summarizes that the lack of awareness may be a problem: *"I think what is really important here is to create awareness on a management level"*. (**Invisible:**) The lack of TD awareness on a developer's level may lead to a lack of TD identification, e.g.: *"[S]o the disadvantage may be the lack of detection of everything, the lack of awareness that something is technological debt, which in fact may have some negative consequences"*.

3.1.7. Technical debt consequences

We discovered a set of TD consequence categories (number of cases in brackets): Technological consequences (96), Loss of time (75), Management consequences (46), Business consequences (46), Additional cost/budget consequences (41), and Human resource consequences (26).

Similarly to the causes, we give one example of visible and invisible consequences for each category, respectively. More details are in the additional material (Wiese and Borowa, 2022).

Technological consequences. These are the consequences usually easily noticeable by the developers, particularly, maintenance problems and architectural erosion. Furthermore, some technological consequences directly impact the business and can be perceived by business stakeholders, too, e.g., unstable systems, development reaching a dead-end, problems adding new features, or performance issues. (**Visible:**) In some extreme cases, system parts had to be turned off completely, as one participant told us: *"[T]hen we discovered that we have to extract it [TD] for a couple of days, weeks. And until then, we have to disable the feature because it simply doesn't work, and it slows down the entire system"*. (**Invisible:**) Issues related to working with outdated technology are another example of this category: *"So that means using outdated technology in some form that makes it difficult for us to keep systems up to date"*.

Time-related consequences. The consequences of this category are commonly the time overhead needed to add new features or fix specific errors. (**Visible:**) One IT Manager explains the impact this time overhead may have on the implementation of new features *", where we sometimes really have to wait a year and a half until they [the changes] can finally be implemented"*. (**Invisible:**) A not well-researched issue resulting from the loss of time is that developers might have to work overtime: *"The expectation is that such changes can be done in a very short period of time. It turned out that it is not [possible]. Then, of course, additional stress and overtime work are standard"*.

Management consequences. This category comprises the impact TD might have on the management of projects or project pipelines and the risk TD poses for IT management. (**Visible:**) Many IT managers described that TD *"hits you hard and surprisingly"*, e.g., *"if you haven't been aware of these things before, then at some point things hit you . . . to a great extent and usually at an inopportune time"*. (**Invisible:**) In one case, an IT manager described that the organizational structure of his company developed analogous to the debt-ridden architectural structure. This led to a situation where *" we had to rethink the underlying architecture in order to be able to say: Okay, certain teams are only responsible for certain areas"*.

Business consequences. These consequences affect the business directly like expectations about features that were not met or losing the customers' trust. (**Visible:**) An example for a visible cause is a missed business opportunity: *"So every minute that I have to run after an undocumented feature that went wrong somewhere unexpectedly, that not only costs time, but it also prevents me from developing anything forward where I can somehow make more money with it"*. (**Invisible:**) Systems created but not used by the business due to a missing business fit might be invisible to the business stakeholders. *"Sometimes you have applications that are standing there, that hardly anyone uses anymore, but you still have to apply security patches, releases somehow. [They] at least inherently by being there, simply create certain risks"*.

Budget-related consequences. The budget is influenced by TD when TD had to be repaid, e.g., by a rewrite. Furthermore, projects might get more expensive, e.g., if additional licenses are needed. (**Visible:**) A perceivable consequence is that *" the development costs had to be gradually increased higher and higher, and then the consequence was the cost of rewriting this system"*. (**Invisible:**) The simple need to rewrite a code fragment is an example of an invisible consequence: *"[TD is] a type of debt, as a result of which some additional costs are incurred . . .. [S]uch a debt has to be repaid, let's say in the form of rewriting a fragment of the code or, in the worst case, by rewriting the entire code"*.

Human resource consequences. is the final category, including influences on the development team members and a company's staffing levels. (**Visible:**) A major problem related to human resources was that developers were leaving or harder to hire. *"Outdated technology doesn't attract developers either. That is, if I'm working with ancient technology now and trying to hire people"*.

(**Invisible:**) Furthermore, TD can lead to stress, bad developers' morale, and blaming each other. An IT manager complained about *"the process of blaming each other, both sides, or crying about who was at fault and why, how angry everyone was with each other"*.

3.2. RQ1.2: Current technical debt management

We explored the current TDM process according to IT managers along the TDM activities: TD Awareness, TD Incurrence & prevention, TD identification, TD monitoring, TD measurement, and TD prioritization & repayment.

Making conscious and rational decisions about incurring and repaying TD is a crucial part of TDM. Conscious decisions about TD incurrence may lead to the prevention of TD items. Conscious decisions about TD repayment are related to prioritizing TD repayments. Therefore, we address the decision-making processes of the IT managers in Section 3.2.2 and Section 3.2.6.

3.2.1. TD awareness levels

We identified the TDM knowledge and awareness of the IT managers' teams with regard to Kruchten et al.'s TD awareness levels (Kruchten et al., 2019). Most IT managers worked at companies on levels 2 and 3 of these TD awareness levels, i.e., the IT managers knew the concept, and knew about their own TD, but were missing methods to manage TD **(level 2)** and had problems communicating these TD items to business stakeholders **(level 3)**.

Only one manager mentioned that they had never heard of the term TD before **(level 1)**: *"What is technical debt? To be honest, I've never really thought about it . . .. I'm going to be a very poor respondent here, unfortunately, but I don't think [I know] anything"*. However, after explaining the definition to them, they replied: *"Of course, this is a familiar topic"*.

One top manager pointed out a TDM process integrated into Enterprise Architecture Management **(level 4)**. This manager explains: *"[W]e have a very good repository through our enterprise architecture management, where we have documented our IT landscape and from a wide variety of perspectives . . ."*. He further reports: *"There [are] . . . regular board meetings. We call it the IT Architecture Board. This is led by our enterprise architect, who then comes in with prepared decision documents. He coordinates or works on these with his respective architects in the various specialist domains . . .. And in the course of this, of course, this topic of technical debt is also addressed"*.

3.2.2. TD incurrence & prevention

Oftentimes TD was caused or increased by reckless or simply bad decisions (see Section 3.1.6. We also assume that mistakes are unconscious decisions to do something in an incorrect way. Decisions that seemed to be valid in the beginning might turn out to be lacking after a period of time. Finally, decisions made by uninvolved stakeholders, e.g., suppliers or government, can turn out to be bad for a specific system.

To analyze this decision-making process and the reasons for TD-related decisions, we examined the reasons why IT managers were willing to incur TD and what led them to refrain from incurring TD.

Why to incur TD?. The incurrence can happen unconsciously or consciously, which was mentioned nearly equally often by the IT managers.

Firstly, they reported situations where TD was incurred **consciously** after evaluating the consequences of TD versus the benefits, e.g., *"we discussed various options, and there was always a fast track. But maybe the solution is not very good, maintainable. . . . But then we said: "Yeah, but we want to deliver, but we will do the high-end solution later on"'*.

Secondly, another conscious incurrence is the need for a quick-fix of a bug, e.g., *" So let's say with the bug fix, for example, I have a problem that I need to solve. Then I might do a quick&dirty bug fix so that the problem is solved."*.

Thirdly, one participant of the focus group explained that they incur TD consciously but already plan the repayment. *"We [are] creating things very fast and [do] not talk [negotiate TD incurrence]. . . . But what we do then is to plan a road map in order to constantly create proper data models in the background . . . so that we can shift later on . . . but the end-user already has the value"*.

Fourthly, **unconscious** TD incurrence was mostly mentioned by the IT managers as caused by a lack of awareness of TD consequences, i.e., TD items are incurred *" not necessarily with adequate preparation or anticipation of the consequences that might come as a result. "*

Finally, TD was also incurred unconsciously, when there was a lack of better options in former times, e.g., *" if we now go back a bit to the 90s. . .. There were no real solutions, or only very, very expensive solutions . . . and in this respect, we started to build small solutions [ourselves]. "*.

Why not to incur TD?. Firstly, the awareness of TD consequences is the main reason not to incur TD, as stated by one IT manager: *"I mean, you can minimize the risk of [TD's] occurrence in the sense of promoting an appropriate technical culture"*

Secondly, a conscious TDM might lead to a raised awareness and change the mindset of stakeholders. This, in turn, might lead to the prevention of unconscious TD. One manager explains: *"It is necessary to have discipline in the sense that just when there is no factor that forces us to do something very quickly, we should think about how to do it well so that the debt is not incurred"*.

Thirdly, TD can be prevented if the debt-free option is also the better solution from the business perspective. This was mentioned by an IT manager talking about a company merger and the resulting IT merger: *"I can bring that together by – kind of – doing a layer on top that glues things together. Or I have different teams do the reporting and then the reporting has to somehow be layered on top of each other in Excel"*.

Fourthly, one manager mentioned that their company only prevents TD in central components they call "tools": *"Tools yes, because we work with tools every day and ninety people use this editor of ours. So tools yes, [but] everything else [is] no longer [prevented]"*.

Lastly, the focus group discussion revealed that to avoid TD, business stakeholders should not decide on TD incurrence. *"[U]sually at least a businessperson without IT background will have no idea [of TD]. And usually, . . . they don't care at all if there is technical debt or not. Because, usually, their mindset is not what will be in one year with this product but like: 'I need these numbers now – yesterday, and it's great that they're there, and I don't care [about next year]"*.

TD prevention strategies. Regarding TD prevention strategies, it is striking that 15 managers mentioned having implemented at least some TD prevention procedures, e.g., by making rational decisions regarding technologies, architecture strategies, or when choosing a supplier. An example of the latter is: *"we do not use anything open-source, and if we already use something open-source, we have the support from some external provider"*. Moreover, they enabled TD prevention mechanisms regarding code debt by establishing (good) coding practices, e.g., unit testing, code review processes, or *"measuring the quality of the code through linter, through static code analysis"*.

3.2.3. TD identification

TD identification by chance was the most frequently mentioned way of identifying TD, e.g., *"So, it's usually in the area in which they [the developers] have likely developed features because that's where they have a look. And then they figure out that something could be done a lot better"*.

While the IT managers agree that early identification is relevant for the manageability of TD, they generally did not have any identification strategies and conceded that they usually discover TD items "too late", e.g., *"That was then actually already too late. . .. These measures were so big that you can no longer just talk about refactoring, but it was really a change in architecture and that was a really big rebuild"*.

3.2.4. TD monitoring

Regarding TD monitoring, many of the IT managers mentioned that their teams create TD tickets for each TD item, but nearly no managers had an overview of these tickets, let alone the ability to control these. Moreover, they concede having problems with *"cluttered backlogs"* and *"rotting tickets"*. One IT manager describes: *"And then they [the tickets] usually disappeared in the backlog [–long pause–] until they were deleted sometime after two years"*.

For this reason, most of the IT managers agreed that someone has to keep track of the TD items on a regular basis to create a TD overview, e.g.: *"In fact, it is also the case that we regularly . . . carry out assessments after certain periods of time"*. They agreed that a responsible person must be tasked with keeping track and warning about the potential incurrence of TD in decision-making processes, e.g.: *"[W]e would have to have someone who basically keeps track of these things and says: Watch out, we're in danger of disaster here"*. Some mention that they expect the team's architect to be this person, e.g., for the "level 4" manager, this *"is led by our enterprise architect . . ."*. While the participants all agreed on this, most of them did not have established a process or responsible person for keeping track.

3.2.5. TD measurement

IT managers conceded to having problems calculating precise values describing aspects of TD, e.g.: *"[T]here are still these options related to estimating the cost of this debt, interest, this type of thing, but it's probably not [easy] like that. No one will be able to estimate it"*.

Still, the estimation of the repayment's business value is relevant for prioritizing TD items: *"So, you know, there is always something not correct right now. And in the end, it's also about evaluating what is the relevance of that right now. And when do we need to take care of it?"*

In this regard, some managers mentioned that the accuracy of the estimation values is secondary to the time needed to carry out these estimations. One manager suggested using *"a risk factor that makes it clear to me how important that is. And just such a, let's say, complexity assessment in story points. Preferably in story points, because then I can compare it with the stories themselves"*. An idea that emerged from one interview was to use architects to get a valuable estimation: *"Surely, there should be people in the organization who are able to quantify the impact in some way, but this, in turn, is quite difficult. I think that people who have this kind of thinking are at the level of architects"*.

3.2.6. TD prioritization & repayment

Analogous to the TD incurrence and prevention, we identified reasons that lead IT managers to repay TD items and that lead them to refrain from repaying these.

Why to repay TD?. Firstly, regarding the repayment of TD, most IT managers described that repayment was possible after some kind of tipping point was reached. These tipping points can be vague, e.g., *"at the moment when such a debt begins to be burdensome, you have to get rid of it, rewrite, for example, such fragments"*. Other tipping points are more distinct, e.g., *"because the report generator doesn't exist anymore, because it's not maintained anymore. Something like that. So where basically a technological change forces a redesign on us"*.

Secondly, sometimes the evaluation of benefits and drawbacks led to some repayment activity. For example, one IT manager explained how the old architecture with high amount of business logic on the database level led to performance issues, and *"to work around the problem, additional resources were added to this database, . . . but this resulted in costs because you had to buy additional licenses . . .. So there was, you could say, interest on this debt . . .. At one point, it was decided that it had to be optimized and partially rewritten to the application layer, and that's how it [the TD] was paid off"*.

Thirdly, some TD was repaid as a result of a single person's initiative, i.e., *"there were people who were just implementing features and they saw that there was code around that needed refactoring, so they refactored it and reduced the technical debt"*

Finally, the planned repayment after conscious incurrence was mentioned, e.g.:"[I]f we introduce something that's just known to be bad but easy to do and that's the only reason we're doing it, we immediately plan a quick fix for the next sprint or in the near future . . .".

Why not to repay TD?. Firstly, by far, the primary reasons not to repay TD were other priorities, including time and budget, business value, or new features. One IT manager stated his experience: *"I have faced throughout my previous experience working in different departments that some systems are being evaluated like, 'yeah, it looks really crappy from a technical perspective, but moneywise, we are kind of fine and we do not want to invest in renewing this stuff.' "*

Secondly, reasons that led to a lack of TD repayment are management mistakes, e.g., one IT manager conceded: *"What – I think – I totally underestimated from a management perspective, actually, how stubbornly certain technical debts carry into the future"*..

Thirdly, problems distributing developers to TD topics became a problem, i.e., *"the technical debt is much deeper and a lot harder to handle because some refactoring things are so, so complicated or so expensive that it's hard to find volunteers that would gladly do this"*.

Finally, there are organizational challenges, which might prevent TD repayment. One IT manager talked about TD, which had to be repaid across teams, and postponed the repayment for a long time. *"And then it was passed on and ultimately decided on by all the teams and also designed: How do we approach this now and solve it holistically and not only extinguish fires within our area"*.

TD repayment strategies. Regarding TD repayment strategies, the IT managers mentioned various ideas for repayment strategies, e.g., regularly allocated time slots or repayment projects. An example of a TD repayment project is the following: *"They are planning now a tech agenda, and they also create projects which are dealing with working on technical debts like this [project] we are now doing"*. However, the implementation of these ideas was rarely successful in the long term, culminating in the statement: *"Someone eventually gets annoyed and starts doing it"*.

This leads to the tendency that "radical measures", e.g., starting from scratch or abandoning a project completely, were the most frequently mentioned repayment strategies. For example: *"[Then] a company-wide decision has been made that we are doing a re-engineering. Basically, not even a re-engineering, but we did it all over again"*.

Iterative repayment was only mentioned a few times and mostly in terms of migrating to a new system iteratively, e.g., *"but there's a plan which has been implemented for two years now, only migrating in stages"*.

3.3. RQ2: The IT managers' main concerns regarding technical debt

As a result of clustering the themes that evolved from coding, we identified three major clusters, i.e., concerns of the IT managers, which are described in detail in the sections below. These concerns are communication about TD (Section 3.3.1), the establishment of a TDM process (Section 3.3.2), and dealing with what we call vintage systems (Section 3.3.3). The results of this clustering were presented to the focus group to discuss and accept or reject the hypotheses. The insights of this discussion are added to the respective subsection.

3.3.1. Communicating about technical debt

IT managers are in the position of a mediator between IT and business stakeholders. Thus, their communication issues are twofold and include communication with business stakeholders and developers.

Communicating with business stakeholders. IT managers often seem to lack the business stakeholders' trust and understanding regarding TDM. However, they can only manage TD if the TDM process is accepted and financed by the business stakeholders. Many IT managers mention intense negotiations with business stakeholders about the business value of TD repayment, e.g.: *"So I had to convince the [business] managers in the organization to take a serious approach to solve these problems, not to act on the basis of just patching things up"*.

However, they are aware of the difficulties developers face in providing such specific information about a business value. One IT manager saw it as their duty to mediate in this case: *"And everything is measured, so to speak, by business value and less by technical value and that is then in turn for my area a challenge to argue why we have to do certain things"*.

When trying to explain the TD concept to their business stakeholders, IT managers were often missing appropriate methods, e.g.: *"Most of the customers we have are IT illiterate. No IT department of their own. We actually work with amateurs"*.

The focus group agreed with this concern and reported similar problems, even before this specific concern was presented to them as a point for discussion by the researchers.

Communicating with development teams. The IT managers described two potentially problematic kinds of developers: quality-oriented developers with a great interest in high software quality, and speed-oriented developers with less focus on quality.

The concern that arises with quality-oriented developers is what we would like to call **"developers crying wolf"** in reference to the fable of "The boy who cried wolf" (Aesop, 2002). These developers tend to complain about TD items regardless of their impact on the business, i.e., the effectiveness of the TD item (Schmid, 2013). For example, one manager lamented: *"[E]veryone who gets someone else's software says: it has to be written from scratch"*.

As a result, IT managers may become desensitized to the term TD, and developers may not be heard when a real crisis occurs. *"The developers also say: So how many times do I have to tell you*

that this is a big problem? Does the shit really have to hit the fan? " This makes it hard to identify and prioritize relevant TD items.

The second type of developer abides by **"it works"** and **"just do it"** principles. They have no habit of discussing solution options, prefer to work alone, do not question requirements, and refrain from voicing their concerns. Many managers mentioned this as a problem, e.g.: *"It didn't matter [to the developer] that it was all duct tape there, but it was working".*

Many IT managers in our study mentioned that regarding TDM, particularly TD prevention, they see it as their task to change these developers' mindsets. One IT manager explained: *"My idea was also to create people [nurture and educate employees] who are able to take care of these things, to educate developers so that they also care".* Yet, the IT managers are missing processes on how to change these mindsets successfully, as stated by another IT manager: *"And on this mindset – I only have a certain degree of influence on it. . .. I can't stand next to it all the time, micro-manage, and have everything presented to me".*

The focus group could relate to both developer types and the related concerns. Regarding the "developers crying wolf", one participant mentioned that particularly experienced developers are *"creating like a tunnel vision, thinking mostly about the technical stuff".* Regarding the "it works" type of developer, another focus group participant argued: *"But we, as technical experts need to, like, really suggest the solution and not just be executors of demands".*

3.3.2. Establishing a technical debt management process

As explained in Section 3.1, most IT managers knew the concept of TD, and knew about their own TD, i.e., they had at least some level of TD awareness. However, most IT managers had not established methods to manage TD properly. They had already implemented some measures for TDM, e.g., filing TD tickets or having a percentage of time for TD repayment. Nevertheless, they failed in their attempts to manage TD sustainably for various reasons.

In Section 3.2, we identified the IT manager's current knowledge and their implementation of TD activities. In the following subsections, we identify three (sub-)concerns related to the TD activities that contribute to the main concern of establishing a TDM process.

TD awareness, incurrence, and prevention. Most IT managers identified that a lack of awareness of a TD item's consequences leads to unconscious TD incurrence, i.e., TD items are incurred *" not necessarily with adequate preparation or anticipation of the consequences that might come as a result".* Furthermore, the awareness of TD consequences was the main reason to avoid TD, as stated by one IT manager: *"I mean, you can minimize the risk of [TD's] incurrence in the sense of promoting an appropriate technical culture"* Therefore, the awareness of TD is a major concern that needs to be addressed for a successful establishment of a TDM process.

TD identification and monitoring. Almost no IT manager had an overview of their TD items, although they admitted that TD monitoring is essential, e.g., *"There is this saying: 'If you can't measure it, you can't manage it.' So if I don't have an overview of what problems I actually have, then I can't tackle them.'* Their backlogs were cluttered with TD items but no structure was set into place to gain an overview of them. Furthermore, while most of the IT managers agreed that someone has to keep track of TD on a regular basis to create a TD overview, they did not assign a specific person to this task. *"[W]e didn't have a formalized method, we generally had a message to have 20% [of our time for TD repayment], but we didn't have control over it".* The derived concern of the IT managers is to structure their backlog, create an overview, and keep it up-to-date.

TD measurement, prioritization and repayment. As shown in Section 3.2, the estimation of the repayment's business value is relevant for prioritizing TD repayments and for communication purposes, i.e., counteracting the "crying wolf" problem, and negotiating with business stakeholders. However, all IT managers lacked information about TD measures to decide on the most urgent TD items before they became too big to handle properly. *"At most, you can somehow indirectly assess that this technical debt exists and in what dimension, but I think it's largely indirect . . .. [T]there are no clear indicators beyond the already glaring situations that help quantify this debt".* Regarding a business value, some managers mentioned that high accuracy of the estimated values was not crucial and that easy-to-use methods that would fit into their existing set of tools and methods were more essential. The struggle with a decision on TD repayment was explained by an IT manager: *"Sometimes these are very small things that originally cost or would have cost 20 days of developer effort. And years later, I'm standing there like this: Oh yes, to rebuild that now, but that costs 200 days, so ten times as much".* One manager described this decision-making process as a *"kind of art of making a choice whether it's worth devoting time to it and whether it will pay off in some sense".* In this case, the main concern of IT managers is to estimate as effortlessly as possible the negative impact of a TD item on a system and the positive contribution of its repayment to a business value.

The focus group agreed with the observations of all three clusters and suggested that TDM should be perceived as a part of risk management, in order to be better aware of the systems' TD. Moreover, to avoid TD, the focus group advised that business stakeholders should not decide on TD incurrence, *"because, usually, their mindset is not what will be in one year with this product but like: 'I need these numbers now – yesterday, and it's great that they're there and I don't care [about next year]".*

3.3.3. Dealing with vintage systems

Many IT managers noticed TD because of problems with outdated and very old systems with deep-seated ATD, which they refer to as systems or decisions *"of the 90s".* Particularly, out of this study's six top managers, four mentioned these 90s systems and software aging as being the main source of TD. While these 90s systems have manifold problems, they have made positive contributions in their time. One top manager summarized: *"I have to say that sometimes I talk as if what we have as software is somehow a big disaster. . . . On the other hand, you have to look at it: The software has been running for 25 years and that is a proud age. . . . The people who developed it must have done a lot of things right, otherwise, it would have gone down the drain a long time ago".*

While the term **"legacy system"** was originally used for old and outdated systems and technologies (Bennett, 1995), nowadays, the term is often used differently in practice. Any system that can no longer be supported and updated may be called "legacy" regardless of age (Khadka et al., 2014).

Therefore, we prefer to call old, valuable systems **"vintage systems"** in an analogy to vintage cars. These old systems, just like vintage cars, have survived more than 20 years, which is a great engineering accomplishment. Thus, vintage systems are a subcategory of legacy systems. They are a source of great business value and a company may even decide to keep using and improving them. Fitting the analogy, there are communities of lovers of these old systems, just as with cars, e.g., "Legacy Code Rocks" (Anon, 2022c). Engineers skilled in working with these vintage systems and cars are both hard to find and expensive.

Despite their value, the IT community and businesses may have to accept that there is a maximum age for which systems, as well as cars, can operate efficiently. The older the system or car gets, the more maintenance is necessary and it is usually

not efficient to keep such a car or system. This can sometimes cause old-established companies to fall behind younger companies, since there is so much TD in their systems, that it cannot be repaid within a reasonable period of time. One top manager mentioned in relation to a vintage system: *"[S]omeone was then assigned to investigate that and he then comes back with well, software license costs, I say, in the seven-figure range. And by the way, it takes three years. So you sit there as the responsible IT manager and then you think: I can't imagine that right now"*.

One of the IT Managers' concerns seems to be a lack of **repayment strategies** for hard-to-tackle ATD typically present in vintage systems. They miss feasible architecture refactorings in small increments. Their usual way of dealing with this kind of TD is one of the following: (1) rewrite the whole system (*"Changing anything was very difficult, and generally we had to . . . write everything from scratch"*.), (2) switch it for a third-party system or (3) get rid of the old system completely (*"So you could say we removed the debt by removing the tool"*.).

Another critical concern with vintage systems is **knowledge vaporization**. The valuable knowledge that older developers have about vintage systems makes these developers irreplaceable. If they leave, retire, or die, their knowledge might be lost. Moreover, these developers might have the "vintage mindset" to do what they are told, not to question authority, and not to share their knowledge freely; as one manager put it: *"People have only a limited interest in making a know-how exchange on a level, where they then make each other more clever"*.

This concern was one that the focus group participants could not relate to. This may be due to their company's relatively young age. They suggested to separate the TD research on younger companies, e.g., start up's, from old-established ones.

3.4. RQ3: Improvement measures for technical debt management

Some of our questions aimed to identify ideas from IT managers on how the situation regarding TD could be improved. Thus, we asked questions to identify the potential for improvement in the TDM process (Section 3.4.1), and the communication related to TDM (Section 3.4.2). We did not ask for potential improvements related to vintage systems, since this concern emerged later during the analysis.

3.4.1. Technical debt management

In Section 3.2, we present the IT managers' knowledge about TDM and, in Section 3.3.2, we show their concerns regarding the establishment of a TDM process. In this section, we present the TDM process' potential improvements from the IT managers' perspective. For this purpose, we asked IT managers about their role in the TDM process, including the actions they can take to influence and improve the process, and the benefits and drawbacks they expect from implementing a TDM process.

IT manager's role in the TDM process. Firstly, all IT managers saw it as their role to manage TD. One IT manager summarized: *"Basically, I'm responsible for the technical state of the platform, so I'm also responsible for the technical debt"*. Secondly, they state to have the authority to manage TD, e.g., *"[I] did have the influence to put these corrective actions in place"*. Finally, they conceded that they should have an overview of TD, e.g., *"I suppose also my role would be to have an overview of things. That cannot be too specific, but then I should encourage people that are in the team to have an eye out for everything"*.

TDM process benefits. Most IT managers saw the benefits of a TDM process primarily in terms of manageability of TD and their systems: *"[T]o me it just brings . . . a manageability to the whole thing . . .. [I]f you don't take care of it and you don't have an overview of what the problems are, you can't reduce them"*. A manager pointed out: *"It's important for me as a manager to have an overview in order to be able to set priorities"*. Another manager focused on predictability: *" If we only discover the technical debt when we approach something and realize now we don't need one hour to do something, but 20 h, because we've reached a dead end. Then, of course, we always have a resource problem"*. Other IT managers have not even considered the benefits of a TDM process before: *" When the system runs well and the team is happy, then pretty much my job is well done. I honestly never, never thought of this question. "*

TDM process drawbacks. While the benefits were self-evident, all managers had problems giving examples of any TDM drawbacks: *"[You] definitely have to do it [TDM]. I don't know any disadvantages. I just know – Frankly, I don't know – So what should be a disadvantage? That you put time into it?"*

The most frequently mentioned drawback was the effort, i.e., time and budget that needs to be put into such a process: *"[I]t would take people who could do something else at that time, ... That's probably the only thing that comes to mind"*. However, most IT managers also mentioned immediately that this effort would be justified by the benefits they could gain from such a process, e.g.: *"Well, it costs: You have to sit down, of course, and it just takes a bit of time. But I would say that is negligible"*.

After some thought, some IT managers pointed out that it is difficult to obtain the budget for TDM from their customers, e.g., *"The only exhausting thing, I would say, is that at the end of the day - especially with our customer clientele - it's hard to get money for it"*.

Other topics mentioned as potential drawbacks by a few IT managers were the risk of over-engineering and micro-managing. Regarding over-engineering, one IT manager explained: *"[D]evelopers are also interested in 100% solutions. I can understand that they want it to be perfect, but then you have to manage away this over-engineering character a bit."* Regarding the danger of micro-management, another IT manager stated: *"It just gets a little bit of bureaucracy out of it . . .. [P]ulling away from the current thing that needs to be patched up immediately"*.

The particularly interesting topic of TD becoming overwhelming and leading to a loss of motivation was mentioned by two IT managers: *"Another challenge may be, . . . that when someone sees how much [TD] there is to do, they lose motivation to do anything at all"*.

Three other disadvantaged were mentioned once each: (1) TDM might not be worth doing in small and short projects, (2) TD items are cluttering the backlog, and (3) TDM creates communications problems. The last topic was explained as follows: *"[TDM] puts additional demands on the development team and also on the project sponsor to understand the situation and if you try to do it in a situation where there's a lack of competence and a lack of understanding, it generates additional problems"*.

3.4.2. Communication

In Section 3.3.1, we showed the concerns IT managers have regarding communication with business stakeholders and developers. Below, we present the role IT managers believe in having regarding this communication topic. Furthermore, to identify potential improvements, we asked the IT managers what they need from the developers and business stakeholders, respectively, to improve TDM.

IT managers. IT managers interpreted their own role as being the mediator between developers and business stakeholders. Their role is to *"kind of try to negotiate between your tech team and business people, . . . because one side wants quick and fast and the other side wants the proper way"*. Some even mentioned the need to influence the company's culture regarding TD: *"Why does a company actually tolerate such technical debt? You often have corporate cultures that - especially in classical cultures - are not used to questioning things, questioning decisions, questioning orders, questioning solution proposals"*.

In their communication effort with the **business stakeholders**, some IT managers see it as their role to ascertain the business fit of the solution. *"One thing that is very important to me is that we do a solid project preparation. So where we do exactly these things like: goal, scope. What is in-scope? What is out-of-scope? "*

Moreover, they believe that it is their role to educate the business stakeholders on the topic of TD. One IT manager sees it as their task *" to create an understanding for this [TD], to get such discussions going with the management as a whole, i.e., to explain this to the management"*. Explaining the issue of TD is necessary to negotiate with business stakeholders to obtain resources for TD repayment, which is another major role of IT managers regarding TDM, i.e., *"granting the freedom also to reduce these technical debts is then up to the management, i.e., to make the time available.'*

In their communication with the **developers**, IT managers perceive their role as mentoring, teaching, and incentivizing a TDM process, e.g.: *"The other thing that I might do is definitely an element of education . . . so that they [the developers] are more aware of this technical debt"*. Another way IT managers try to create this TD awareness is to *"start to manage that debt in general, which is to be aware of those gaps or things, that we need to do somewhere, that we've put off doing in the past"*.

Furthermore, they see it as their role to create an environment based on trust: *"[T]hey [developers] have to be sure enough of their place in the project or such, to feel supported, to bring such things out, to say them somewhere"*. In turn, IT managers have to trust the developers and their decisions: *"If you have . . . the confidence in the development that they are not twiddling their thumbs for weeks, but that something is really being changed, . . . then you also know how important it is to manage these things [TD]"*.

Developers. Firstly, IT managers wished that their developers would communicate better and earlier about issues that occurred. They wanted the developers to proactively voice their concerns and question decisions. They want them to *"come forward and not only focus on the small thing that they are working on without considering how it interacts left and right with something else"*. One IT manager emphasizes the need to escalate such issues: *"I have to train my employees . . . to recognize something like that and to escalate it in time"*.

Secondly, IT managers want the developers to take individual action when needed, i.e., do small refactorings regularly. One IT manager wanted the developers to consider: *"Okay, if I get past such and such corners that are difficult, why don't I clean them up right away?"* They further concede that this *"would mean, for example, from a project management point of view, plan a little time for refactoring, but above all this awareness on the part of the developers"*.

Lastly, an IT manager mentioned that they need the developers to provide the information necessary to manage TD. They mentioned costs, impact, and risk estimations, but also that the shortage of qualified developers can become a bottleneck: *"[M]aybe it's not bad to know – sometimes you have such expertise within the team – who of the developers, for example, can even tackle this"*.

Business stakeholder. Firstly, most IT managers wished for better communication of business decisions. They wished for the business stakeholders to be actively involved in projects, because then *" he can understand the problems and then he can make those decisions, whether we leave it or take the time to change it"*. IT managers want to be informed and understand the rationale behind business decisions, i.e., they need *"communication about changes before they happen. Not that I open the editor [with the requirement], and everything is different than it was yesterday"*. According to our participants, business stakeholders should provide *"a clearer roadmap of where we want to go, so that we can think forward at least a couple of months"*.

Secondly, one issue was the business stakeholders' trust in the IT managers. *"So I need trust from my manager. Trust that what we're tackling is important. That it's important that we now reduce this technical debt that we've decided to do [repay]. I need the confidence"*. Another manager shared a similar observation: *"I'm responsible for the implementation, and then let me be responsible"*.

Thirdly, IT managers want the business stakeholders to recognize the importance of TD and to better understand its consequences, e.g.: *"Well, and he [a business manager] also has just to recognize that [TDM] as one of the important factors of building a business. . . In particular, the lack of [TD] management will have a negative impact"*. The focus group discusses a situation where warnings from the development team regarding the data quality of a report were simply ignored. They warned the business stakeholders *" Hey, there is . . . this quick and dirty solution. Please don't use it, if you want to have reliable information"*.. But they observed that the business *"people don't care, because as long as they have something, they are fine with it"*.

Finally, IT managers needed proper conditions, particularly time, to enable TDM without recurring negotiations for each TD item. This time aspect is twofold. For one thing, this means time for the developers to repay TD, *"because I think that every employee, if he is told that he has, for example, 15% of the time, 10% of the time a day or a month to do some verification and so on, to complete such information, and to propose software updates, then he will be happy"*. Additionally, IT managers need time to manage TD properly, e.g., *"[W]ell, these are things again, like this technical debt, when I know that some things would be nice to do, maybe to implement [a TDM process], but I don't have time to do it"*.

4. Discussion

In this section, we discuss the results of Section 3 with regards to the IT managers' main concerns as identified in Section 3.3. These concerns are communication about TD, establishing a TDM process, and vintage systems. Regarding the communication about TD, we focus on communication with business stakeholders, while communication with developers is discussed as part of the "establishing a TDM process" concern.

4.1. Business communication

Both the misunderstanding of the TD metaphor and communicating on different visibility levels are possible sources of communication problems.

4.1.1. Metaphor

TD is a frequently, yet colloquially and ambiguously, used term in practice. However, the idea behind the metaphor, e.g., principal and interest, is often hidden from practitioners. They usually do not consider the financial metrics of the metaphor, like interest and principal. This indicates that the metaphor does not help with business communication, for which it was originally created

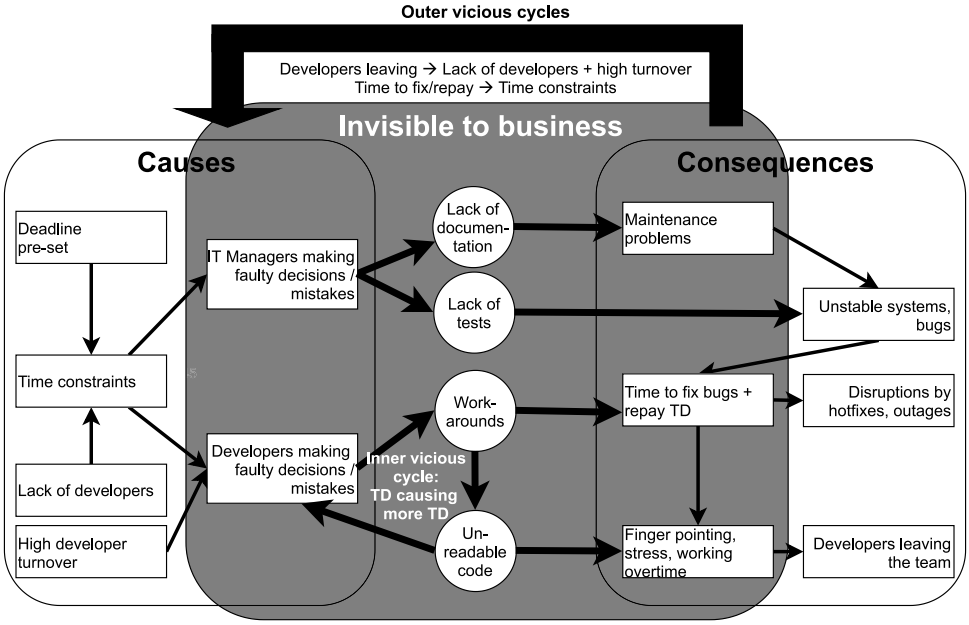


Fig. 5. Example usage of a 3-minute slide. (small arrows = chains of causes/consequences; big arrows = cause–effect relation).

for Cunningham (1992). Furthermore, IT managers invented their own metaphors and used them in business communication.

One IT manager suggested that a metaphor should be domain-specific to be understandable to business stakeholders. In retrospect, this was the case for Cunningham's TD metaphor that was developed while working in the financial sector.

Other IT managers suggested a kind of metaphor list or collection, from which IT stakeholders could choose an adequate metaphor. A starting list for such collection could be extracted from our results (see Section 3.1.2). An expanded list could be collected, e.g., via social media, though the usefulness of such a list would still need to be proven.

4.1.2. V4CTD model of visibility, cycles & chains of causes & consequences of TD

Through our V4CTD model's identification of invisible causes and consequences, we uncovered potential reasons for communication problems with business stakeholders. IT stakeholders often communicate using causes and consequences that business stakeholders do not understand. Sometimes developers argue by explaining the TD item itself, which might be invisible even to their IT managers. Yet, it is important to help business stakeholders understand the underlying dependencies of their own actions and the TD symptoms they may observe.

Firstly, it would be helpful if IT managers particularly asked their development team about these benefits and gave them guidance on how to prepare the information about these benefits. A top manager stated his mistake in underestimating the impact of TD, in particular, TD growth: *"That is the mistake. We should have worked out the benefits more precisely and said: 'What do we gain from all this?'"*

Secondly, our V4CTD model can help with the explanation of TD and argumentation for TD repayment. This kind of tool was even explicitly requested by one participant: *"How do I argue correctly? What does not serve as argumentation? A one-hour argumentation does not help. A manager usually has no more than a three minutes attention span"*.

The V4CTD model can be used, to create a **"three-minute slide"** for an "elevator pitch": First, IT managers, with the help of their developers, may use the overview of in-/visible causes and consequences (Fig. 4) to look for their own chains and cycles of causes and consequences. Second, they can transfer these chains to the V4CTD model from Fig. 2. Finally, they can use this one-slide overview to explain and discuss TD with their business stakeholders.

In our example in Fig. 5, time pressure, and a lack of developers lead to unstable systems, business disruption, and developers leaving the team. Workarounds leading to unreadable code create an inner vicious cycle. Furthermore, unreadable code may lead to more developers' decisions to create workarounds. The lack of developers and developers leaving the team creates an outer vicious cycle.

4.2. Establishing a technical debt management process

Comparing the results of RQ 3 (Section 3.4.1) with the issues regarding TDM presented for RQ 1 and RQ 2 (Sections 3.2 and 3.3.2), we noticed a contradiction between their actual actions and their belief that they are responsible for TDM. On the one hand, most IT managers stated that TDM is an important topic. None of them saw sufficient disadvantages in TDM to outweigh its benefits (Section 3.4.1). On the other hand, only one IT manager's team used any kind of basic process to manage TD. The others did not have an overview of their TD, making TD unmanageable (Section 3.3.2). This may be an effect connected to the **status quo bias**, i.e., a cognitive bias that presents itself as an irrational preference for maintaining the current state of things (Kahneman et al., 1991). The status quo bias may make individuals more likely to choose the default option of not changing anything, particularly when a high number of alternatives is possible , which is the case when deciding on actions related to TDM.

Overall, IT managers are a crucial part of TDM. Establishing a TDM process and communication with business stakeholders is almost impossible without their support. Though all of the IT managers had some degree of control over time or budget, most of them did not connect their control of these two crucial factors with TDM.

To address the status quo bias, we propose researching how a TDM process might be successfully established rather than focusing solely on the details of the TDM process' structure and

activities. To provide a starting point for this research, we identified three sub-concerns in Section 3.3.2, which also represents a sub-structure of closely related TD activities: (1) TD Awareness, Incurrence, and Prevention, (2) TD Identification and Monitoring, (3) TD Measurement, Prioritization and Repayment.

The **first sub-concern** is raising TD awareness to avoid the incurrence of unnecessary TD. Addressing this might be as simple as establishing a basic TDM process as a starting point, i.e. to overcome the status quo bias and "just get started". Some IT managers even decided to start a TDM process as a result of their participation in our study. An established TDM process might also help with the communication issues with developers (Section 3.3.1) as indicated by some IT managers (Section 3.4.2). Moreover, a structured and traceable process can give **"it works" developers** a framework for their work and help them understand and pursue quality objectives. The improvement measures that IT managers wish from developers (Section 3.4.2) may also be addressed through a TDM process as it could enable structured communication about TD. Finally, establishing a structured process might lead to a raised TD awareness not only for the developers but for the whole company, including business stakeholders. The (initial) effort needed for establishing such a TDM process is a drawback mentioned in Section 3.4.1. It is, therefore, essential to focus on introducing a TDM process in a manner requiring minimal effort.

Regarding the **second sub-concern** of creating a TD overview and keeping track of TD, the IT managers usually suggested the use of simple strategies (Section 3.2). Most recommended the use of already established ticket tools, e.g., Anon (2022d), Anon (2022e), or a wiki page to create a TD overview. One manager suggested that TD identification may take place during retrospectives. Other IT managers suggested including TDM in the Scrum process and using story points for TD measurements. Finally, an idea mentioned by two IT managers was the integration of TDM into risk management.

The **third sub-concern**, i.e., measuring TD to be able to prioritize TD items and to justify the repayment of TD, is the most difficult to solve. The idea of using financial metrics for TD, as the metaphor suggests, was not familiar to many IT managers (Section 4.1.1). However, most of them noted that a TD repayments' business value might help in negotiations with business managers (Section 3.3.2). A benefit IT managers have not considered is the positive influence an implemented prioritization mechanism might have on the issues with the **"developers crying wolf"**(Section 3.3.1). Analogous to the first and second sub-concerns, the IT managers mentioned that the measurements need to be easily calculated without putting additional (time) pressure on the developers. In return, they are prepared to accept the lower accuracy of these values.

Summarizing the sub-concerns, we presume that the TDM process and the identification of estimates must be part of an already established process in the company to enable a simple approach and to enforce the process in a sustainable way. Existing methods from academic research might be too disconnected from methods that are already used in practice, and further research might fill this gap.

4.3. Vintage systems

Vintage systems are a source of deep-seated TD, particularly ATD. When asked for the most impactful of their TD item examples, two managers decided not to choose the TD item that could simply be repaid by investing a sum of money. To them, this seemed to be an easy solution and, therefore, a less problematic TD item. Instead, they decided on a more complicated TD item that would not be easy to repay. *"One [TD item] is just an investment ruin, . . . you can solve that to some extent with money. [So let's focus on the other TD item.]"* This indicates that the worst cases of TD are those that cannot be repaid by a simple investment alone. These types of TD, which are usually part of intricate vintage systems, are extremely challenging to repay. IT managers responsible for vintage systems were often missing ideas or processes on how to deal with these problems. What is more, the great amount of accumulated TD might lead to the loss of motivation, as explained in Section 3.4.1.

We suggest that research should further focus on iterative approaches to migrate vintage systems without disturbing the business. Additionally, the research community could also explore methods that allow crucial parts of these vintage systems to live on and be useful for an even longer span of time. Finally, methods to sort through big amounts of TD might be helpful for IT managers when migrating vintage systems or, particularly, when avoiding a complete migration. There are similar methods for code debt in static analysis tools, e.g., Anon (2022f), but they are still missing on a general level.

However, companies with systems that are more than 20 years old need to be aware of the issues involved and implement investment plans to fund modernization projects or purchase new systems.

5. Related work

Our related work section focuses on the main insights that our research revealed regarding (1) the suitability of the TD metaphor, (2) TD causes, consequences, and visibility, (3) TD communication with business stakeholders and developers, (4) establishing a TDM process, and (5) vintage/legacy systems.

5.1. Technical debt metaphor

On the topic of the TD metaphor, Klaus Schmid (Schmid, 2013) identified the "analogy breaking points" and advised not to over-extend the metaphor's applicability to avoid side-tracking research. Stochel et al. (2020) discussed the ambiguity of the term TD. They attributed this ambiguity to the metaphorical origin of the term, which allows for individual interpretation. Our work supports these perceptions. Most practitioners do not understand the subtleties of the metaphor. Moreover, most of them also do not see the need to use it more intensively.

5.2. Technical debt causes, consequences, and their visibility

Overall, the TD causes and consequence types that we found are quite similar to the findings from the InsightTD project (Rios et al., 2020; Ramač et al., 2021). However, our list is not as extensive on the topic of technical/development type causes and consequences and focuses more on other areas, e.g., business-related causes.

The relationship between TD causes, items, and consequences has been described in various ways by researchers so far as presented in Section 1.4. However, as far as we know, we are the first to divide TD causes into visible and invisible. Additionally, the distinction between consequences and symptoms has been interpreted ambiguously, so far.

Various researchers have noticed "vicious cycles", where the existence of one TD item was the trigger behind incurring growing amounts of debt, e.g., (Besker et al., 2018). This problem has been explored in detail by Martini and Bosch (2015) in the context of ATD, where they explain how ATD growth can be particularly contagious, and thus dangerous.

In our work, we went a step further in describing the relationships between TD causes, consequences, and items on a

conceptual level. Not only did we distinguish between outer and inner TD vicious cycles, but we also defined chains of causes and consequences that describe how various causes/consequences impact each other.

5.3. Technical debt communication

Various studies suggest that communication about TD is an important factor in TDM. Verdecchia et al. (2021) describe communication as one of the core categories influencing ATD. Tom et al. (2013) state that poor communication is a cause of inadvertent TD and propose a framework that may help development teams communicate with their managers about TD.

Besker et al. (2022) show how managers communicating the need to keep TD levels low to their development teams can significantly impact the amount of TD in a system. Wiese et al. (2022) show how an established TDM process can make discussions about TD within development teams more rational. This indicates that our idea of a TDM process being the source of better communication with and within development teams might be worth exploring further.

A way of improving communication with business stakeholders is involving them in the TDM process, as Reboucas De Almeida et al. (2018), Reboucas De Almeida (2019) achieved by including both business and technical stakeholders in business-driven TD prioritization. As shown by Kruchten et al. (2012), TD effects may be visible (symptoms) and invisible (consequences) for business stakeholders, which impacts communication. Our V4CTD model enables better communication by explaining communication issues through the concept of visibility.

5.4. Establishing a technical debt management process

There are two recent tertiary studies on TDM by Rios et al. (2018) and by Junior and Travassos (2022) that give an overview of the TDM research field. They both found that TDM is viewed as a set of TD activities and their supporting strategies and tools. Both did not find any research on the systematic establishment of a TDM process. Both also agreed that a holistic approach to TDM, including various TD activities and various TD types, is missing, which makes it hard for practitioners to profit from the research results. This coincides with our finding that IT managers seem to lack suitable guidelines on how to establish TDM.

A general guideline for establishing TDM was presented in the book of Kruchten et al. (2019), particularly, in the "What can you do today?" sections. Yet, the feasibility of this guideline is not evaluated and may need to be improved.

Some case studies provide information on establishing a TDM process. For example, Ramasubbu et al. provided a case study that integrated TD management into quality management steps (Ramasubbu and Kemerer, 2019). Wiese et al. (2022) presented a framework integrating TDM into project management. However, these and other case studies do not provide a guideline for other companies and did not evaluate their work in other companies, e.g. in different work environments.

Additionally, Rios et al. (2018) identified four TDM macro activities, i.e., Prevention, Identification, Monitoring, and Payment. While the idea of macro activities is similar to our clusters of interrelated TD activities, the clusters themselves are different. While Rios et al.'s study's clustering results from an academic literature review, our clustering shows the perception of IT managers in practice.

5.5. Technical debt and vintage/legacy systems

Regarding the relationship between TD and legacy systems, Boss et al. (2016) stated in 2016 that no work on handling architectural technical debt in legacy systems existed and showed a way to deal with these problems using architectural checks. Gupta et al. gave an example of how the amount of TD in a legacy code base was decreased by using agile practices (Gupta et al., 2017). Holvitie et al. stated that over 50% of TD could come from "Legacy from an earlier team/individual working" (Holvitie et al., 2018), which means that dealing with these old "inherited" systems should be a major topic when dealing with TD. In their case study, Lenarduzzi et al. (2020) showed that migrating from a legacy monolithic architecture to a microservice-based one, may cause an initial spike of TD, but does reduce TD in the long run.

Overall, while research on legacy/vintage systems in the context of TD has been done, it has not been very extensive. Our study shows that this is an important topic that should be explored in the future, particularly, researching ways to preserve or adapt parts of those systems.

6. Threats to validity

We present threats to validity based on the guidelines provided by Wohlin et al. (2012).

Construct validity. The selection of participants and interview questions have a great impact on the validity of the study. We used purposeful sampling and achieved code saturation to ascertain that we provided relevant and exhaustive results. Regarding the participants, we overcame the survivorship bias by interviewing former managers and used two iterations to vary the participants in iteration 2 even more. Still, the IT managers already suffering from TD may have been more willing to participate in the study than managers that did not suffer from TD.

Regarding the interview questions, we asked for the most impactful TD item for a detailed analysis. Therefore, the study may be biased towards the impact of ATD on the systems and companies since ATD items were usually considered to be the most impactful.

Internal validity. Internal Validity describes the threats of drawing wrong conclusions about causal relationships between treatment and outcome. Therefore, internal validity mainly refers to quantitative studies such as experiments. In an interview study, there is the possibility of drawing wrong conclusions from the interview transcripts, e.g., by misunderstanding the participants. To make sure that the coding of the interviews did not lead to such misconceptions, the coding was done by two researchers. Additionally, we were able to contact and ask the participants for clarification if some of their explanations could be misunderstood. A strict and communicative process, as described in Section 2, further minimized these internal risks to validity.

External validity. External validity threats refer to the ability to generalize the result. However, the goal of a qualitative study, such as ours, is to create new theories and not generalize the results. The generalizability could be improved by further studies validating the results, e.g., by surveying a representative group of IT managers and confronting them with this study's results.

Conclusion validity. The visibility categorization shown in our V4CTD model is based on interviews with IT managers but provides testimonies about business stakeholders' knowledge of TD. An additional study with business stakeholders, where they evaluate our V4CTD model in practice, should, therefore, follow this study. Moreover, the distinction between visible and invisible

causes and consequences highly depends on the business stakeholders' knowledge of the IT domain. Therefore, the distinction might differ from company to company and can never be considered absolute. However, the contribution of this study is that we made the distinction obvious and gave practitioners a starting point for their own 3-minute slide. The focus group comprised team leaders from the Business Intelligence group of a single company. This may cause their insights to be limited to one company and this IT domain. However, since those IT managers' IT domain was underrepresented during the interviews, we were able to test the transferability of our findings by consulting them with this group. Still, the results must be evaluated carefully as they might be biased.

7. Conclusion

We interviewed 16 IT managers and had one three-person focus group discussion. We identified the IT managers' knowledge, their three main concerns regarding TD(M), and the potential improvements they perceived. We compared the results of the study to the current body of knowledge to identify knowledge gaps in research and new insights.

- We uncovered that the original metaphor may not be helpful when conveying the meaning of TD. The financial metaphor is rather used as an an ambiguous informal term and is often not understood in detail.
- We identified causes and consequences on different visibility levels for business stakeholders, IT managers, and development teams. These different levels may be the cause of further communication issues.
- We created the V4CTD model to include chains and (vicious) cycles of causes and consequences as an extension of the TD conceptual model.
- We uncovered the duality of the IT managers' problems in communicating with developers: the "developers crying wolf" and the "it works" developer. Both of these issues can be addressed by establishing a TDM process.
- We observed the impact of the status quo bias on establishing a TDM process and we identified three sub-concerns that need to be addressed to overcome this bias. Two of these sub-concerns can be addressed by establishing a basic TDM process as a starting point. There exists a cyclic dependency between the concerns regarding the TDM's establishment and the TDM's establishment itself, which has to be interrupted by "just starting".
- We displayed the value of old systems and called them vintage systems. These systems were notably mentioned by top managers and occurred mostly in old established companies. Though it is a major issue in practice, research in this field seems to be lacking.

For practitioners, we propose the use of our V4CTD model to overcome miscommunication about TD. Furthermore, we provide insights into the status quo bias and the before-mentioned cyclic dependency. This could lead to more IT managers agreeing to adopt a TDM process.

For researchers, the V4CTD model expands the conceptual model of TD and we provided a sub-structure of interrelated TD activities. Additionally, we identified the need for methods on how to establish a TDM process in a feasible and sustainable way. Finally, we pose the (research) question if and in which cases it might be an option to keep and adapt vintages systems.

7.1. Future work

We suggest evaluating the feasibility of using our V4CTD-model-based three-minute slides in practice and validating the list of visible and invisible causes and consequences by business stakeholders. Additionally, the possible use of other metaphors may be researched further. Moreover, future research on TDM should focus on the establishment of TDM processes and how to overcome the status quo bias. Since we did not analyze the potential improvements regarding vintage systems, we could not provide any aid for practitioners on this topic. Therefore, this might be an additional problem worth researching.

CRediT authorship contribution statement

Marion Wiese: Conceptualization, Methodology, Formal analysis, Investigation, Resources, Data curation, Writing – original draft, Writing – review & editing, Visualization, Project administration. **Klara Borowa:** Conceptualization, Methodology, Formal analysis, Investigation, Resources, Data curation, Writing – original draft, Writing – review & editing, Visualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The data is available on Zenodo as referenced in the manuscript.

Acknowledgment

We thank the participants of our research for their participation, their trust, and the insights they provided.

References

Aesop, 2002. In: Gibbs, L. (Ed.), Aesop's Fables. Oxford University Press.

Ahmadjee, S., Mera-Gomez, C., Bahsoon, R., 2021. Assessing smart contracts security technical debts. In: Proceedings - 2021 IEEE/ACM International Conference on Technical Debt, No. March 2021. TechDebt 2021, pp. 6–15. http://dx.doi.org/10.1109/TechDebt52882.2021.00010.

Anon, 2022a. InsighTD project – InsighTD project. URL http://www.td-survey.com/.

Anon, 2022b. MAXQDA | All-in-one tool for qualitative data analysis & mixed methods - MAXQDA. URL https://www.maxqda.com/.

Anon, 2022c. Legacy code rocks!. URL https://www.legacycode.rocks/.

Anon, 2022d. Jira | Issue & project tracking software | Atlassian. URL https://www.atlassian.com/software/jira.

Anon, 2022e. Trello | Manage your team's projects from anywhere. URL https://trello.com/.

Anon, 2022f. SonarQube. URL https://www.sonarqube.org/downloads/.

Avgeriou, P., Kruchten, P., Ozkaya, I., Seaman, C., 2016. Managing technical debt in software engineering. Dagstuhl Rep. 6 (4), 110–138. http://dx.doi.org/10.4230/DagRep.6.4.110.

Baltes, S., Ralph, P., 2022. Sampling in software engineering research: a critical review and guidelines. Empir. Softw. Eng. 27 (4), http://dx.doi.org/10.1007/s10664-021-10072-8.

Bennett, K., 1995. Legacy sysfems: Coping with success. IEEE Softw. 12 (1), 19–23. http://dx.doi.org/10.1109/52.363157.

Besker, T., Bosch, J., Martini, A., Bosch, J., 2018. Technical debt cripples software developer productivity: A longitudinal study on developers' daily software development work. In: Proceedings - International Conference on Software Engineering. ACM, pp. 105–114. http://dx.doi.org/10.1145/3194164.3194178.

Besker, T., Martini, A., Bosch, J., 2022. The use of incentives to promote technical debt management. Inf. Softw. Technol. 142, http://dx.doi.org/10.1016/j.infsof.2021.106740.

Boss, B., Tischer, C., Krishnan, S., Nutakki, A., Gopinath, V., 2016. Setting up architectural SW health builds in a new product line generation. In: Proccedings of the 10th European Conference on Software Architecture Workshops. ECSAW '16, Association for Computing Machinery, New York, NY, USA, http://dx.doi.org/10.1145/2993412.3003392.

Campbell, J.L., Quincy, C., Osserman, J., Pedersen, O.K., 2013. Coding in-depth semistructured interviews: Problems of unitization and intercoder reliability and agreement. Sociol. Methods Res. 42 (3), 294–320. http://dx.doi.org/10.1177/0049124113500475.

Corbin, J., Strauss, A., 2014. Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory, fourth ed. Sage publications.

Cunningham, W., 1992. The WyCash portfolio management system. In: Proceedings of the Conference on Object-Oriented Programming Systems, Languages, and Applications, Vol. Part F1296, No. 2. OOPSLA, pp. 29–30. http://dx.doi.org/10.1145/157710.157715.

Ernst, N., Kazman, R., Delange, J., 2021. Technical Debt in Practice. The MIT Press Cambridge, Massachusetts.

Garrison, D.R., Cleveland-Innes, M., Koole, M., Kappelman, J., 2006. Revisiting methodological issues in transcript analysis: Negotiated coding and reliability. Internet High. Educ. 9 (1), 1–8. http://dx.doi.org/10.1016/j.iheduc.2005.11.001.

Gomes, F.G., Dos Santos, E.P., Mendes, T.S., De Mendonça Neto, M.G., Silva Freire, E.S., Spínola, R.O., 2022. Investigating the point of view of project management practitioners on technical debt - A preliminary study on stack exchange. In: Proceedings - 2022 IEEE/ACM International Conference on Technical Debt, Vol. 1, No. 1. TechDebt 2022, ACM, pp. 31–40. http://dx.doi.org/10.1145/3524843.3528095.

Gupta, R.K., Manikreddy, P., Arya, K.C., 2017. Pragmatic scrum transformation: Challenges, practices & impacts during the journey a case study in a multi-location legacy software product development team. In: Proceedings of the 10th Innovations in Software Engineering Conference. ISEC '17, Association for Computing Machinery, New York, NY, USA, pp. 147–156. http://dx.doi.org/10.1145/3021460.3021478.

Holvitie, J., Licorish, S.A., Spínola, R.O., Hyrynsalmi, S., MacDonell, S.G., Mendes, T.S., Buchan, J., Leppänen, V., 2018. Technical debt and agile software development practices and processes: An industry practitioner survey. Inf. Softw. Technol. 96, 141–160. http://dx.doi.org/10.1016/j.infsof.2017.11.015.

Jain, S., Babar, M.A., Fernandez, J., 2013. Conducting empirical studies in industry: Balancing rigor and relevance. In: 2013 1st International Workshop on Conducting Empirical Studies in Industry, CESI 2013 - Proceedings. IEEE, pp. 9–14. http://dx.doi.org/10.1109/CESI.2013.6618463.

Junior, H.J., Travassos, G.H., 2022. Consolidating a common perspective on technical debt and its management through a tertiary study. Inf. Softw. Technol. 149, http://dx.doi.org/10.1016/j.infsof.2022.106964.

Kahneman, D., Knetsch, J.L., Thaler, R.H., 1991. Anomalies: The endowment effect, loss aversion, and status quo bias. J. Econ. Perspect. 5 (1), 193–206.

Khadka, R., Batlajery, B.V., Saeidi, A.M., Jansen, S., Hage, J., 2014. How do professionals perceive legacy systems and software modernization? In: Proceedings - International Conference on Software Engineering, No. 1. pp. 36–47. http://dx.doi.org/10.1145/2568225.2568318.

Kruchten, P., Nord, R., Ozkaya, I., 2019. Managing Technical Debt: Reducing Friction in Software Development. Addison-Wesley Professional.

Kruchten, P., Nord, R.L., Ozkaya, I., Visser, J., 2012. Technical debt in software development. ACM SIGSOFT Softw. Eng. Notes 37 (5), 36–38. http://dx.doi.org/10.1145/2347696.2347698.

Lenarduzzi, V., Lomio, F., Saarimäki, N., Taibi, D., 2020. Does migrating a monolithic system to microservices decrease the technical debt? J. Syst. Softw. 169, http://dx.doi.org/10.1016/j.jss.2020.110710.

Li, Z., Avgeriou, P., Liang, P., 2015. A systematic mapping study on technical debt and its management. J. Syst. Softw. 101, 193–220. http://dx.doi.org/10.1016/j.jss.2014.12.027.

Martini, A., Besker, T., Bosch, J., 2018. Technical debt tracking: Current state of practice: A survey and multiple case study in 15 large organizations. Sci. Comput. Program. 163, 42–61. http://dx.doi.org/10.1016/j.scico.2018.03.007.

Martini, A., Bosch, J., 2015. The danger of architectural technical debt: Contagious debt and vicious circles. In: Proceedings - 12th Working IEEE/IFIP Conference on Software Architecture. WICSA2015, pp. 1–10. http://dx.doi.org/10.1109/WICSA.2015.31.

Pérez, B., Castellanos, C., Correal, D., Rios, N., Freire, S., Spínola, R., Seaman, C., Izurieta, C., 2021. Technical debt payment and prevention through the lenses of software architects. Inf. Softw. Technol. 140, http://dx.doi.org/10.1016/j.infsof.2021.106692.

Ramač, R., Mandić, V., Taušan, N., Rios, N., Freire, S., Pérez, B., Castellanos, C., Correal, D., Pacheco, A., Lopez, G., Izurieta, C., Seaman, C., Spinola, R., 2021. Prevalence, common causes and effects of technical debt: Results from a family of surveys with the IT industry. J. Syst. Softw. http://dx.doi.org/10.1016/j.jss.2021.111114.

Ramasubbu, N., Kemerer, C.F., 2019. Integrating technical debt management and software quality management processes: A framework and field test. Test Eng. Manag. 45 (7–8), http://dx.doi.org/10.1145/3180155.3182529.

Reboucas De Almeida, R., 2019. Business-driven technical debt prioritization. In: Proceedings - 2019 IEEE International Conference on Software Maintenance and Evolution. ICSME 2019, pp. 605–609. http://dx.doi.org/10.1109/ICSME.2019.00096.

Reboucas De Almeida, R., Kulesza, U., Treude, C., Cavalcanti Feitosa, D., Lima, A.H.G., 2018. Aligning technical debt prioritization with business objectives: A multiple-case study. In: Proceedings - 2018 IEEE International Conference on Software Maintenance and Evolution. ICSME 2018, pp. 655–664. http://dx.doi.org/10.1109/ICSME.2018.00075.

Rios, N., de Mendonça Neto, M.G., Spínola, R.O., 2018. A tertiary study on technical debt: Types, management strategies, research trends, and base information for practitioners. In: Information and Software Technology, Vol. 102. pp. 117–145. http://dx.doi.org/10.1016/j.infsof.2018.05.010.

Rios, N., Oliveira Spínola, R., de Mendonça Neto, M.G., Seaman, C., 2019. Supporting analysis of technical debt causes and effects with cross-company probabilistic cause-effect diagrams. In: 2019 IEEE/ACM International Conference on Technical Debt. TechDebt, pp. 3–12. http://dx.doi.org/10.1109/TechDebt.2019.00009.

Rios, N., Spínola, R.O., Mendonça, M., Seaman, C., 2020. The practitioners' point of view on the concept of technical debt and its causes and consequences: a design for a global family of industrial surveys and its first results from Brazil. Empir. Softw. Eng. 25 (5), 3216–3287. http://dx.doi.org/10.1007/s10664-020-09832-9.

Rocha, J.C., Zapalowski, V., Nunes, I., 2017. Understanding technical debt at the code level from the perspective of software developers. In: ACM International Conference Proceeding Series, No. i. pp. 64–73. http://dx.doi.org/10.1145/3131151.3131164.

Runeson, P., Höst, M., Rainer, A., Regnell, B., 2012. Case study research in software engineering: Guidelines and examples. In: Case Study Research in Software Engineering: Guidelines and Examples. http://dx.doi.org/10.1002/9781118181034.

Saldaña, J., 2013. the Coding Manual for Qualitative Researchers, second ed. SAGE publications, London, England, http://dx.doi.org/10.1108/qrom-08-2016-1408.

Saunders, B., Sim, J., Kingstone, T., Baker, S., Waterfield, J., Bartlam, B., Burroughs, H., Jinks, C., 2018. Saturation in qualitative research: exploring its conceptualization and operationalization. Qual. Quant. 52 (4), 1893–1907. http://dx.doi.org/10.1007/s11135-017-0574-8.

Schmid, K., 2013. On the limits of the technical debt metaphor: Some guidance on going beyond. In: Fourth Workshop on Managing Technical Debt, Workshop At the International Conference on Software Engineering. IEEE, pp. 63–66. http://dx.doi.org/10.1109/MTD.2013.6608681.

Soliman, M., Avgerioua, P., Lia, Y., Avgeriou, P., Li, Y., 2021. Architectural design decisions that incur technical debt — An industrial case study. Inf. Softw. Technol. 139, http://dx.doi.org/10.1016/j.infsof.2021.106669.

Stochel, M.G., Cholda, P., Wawrowski, M.R., 2020. On coherence in technical debt research : Awareness of the risks stemming from the metaphorical origin and relevant remediation strategies. In: Proceedings - 46th Euromicro Conference on Software Engineering and Advanced Applications. SEAA2020, pp. 367–375. http://dx.doi.org/10.1109/SEAA51224.2020.00067.

Stol, K.J., Fitzgerald, B., 2015. Theory-oriented software engineering. Sci. Comput. Program. 101, 79–98. http://dx.doi.org/10.1016/j.scico.2014.11.010.

Tom, E., Aurum, A., Vidgen, R., 2013. An exploration of technical debt. J. Syst. Softw. 86 (6), 1498–1516. http://dx.doi.org/10.1016/j.jss.2012.12.052.

Verdecchia, R., Kruchten, P., Lago, P., Malavolta, I., 2021. Building and evaluating a theory of architectural technical debt in software-intensive systems. J. Syst. Softw. 176, http://dx.doi.org/10.1016/j.jss.2021.110925.

Vogel-Heuser, B., Bi, F., 2021. Interdisciplinary effects of technical debt in companies with mechatronic products — a qualitative study. J. Syst. Softw. 171, http://dx.doi.org/10.1016/j.jss.2020.110809.

Wiese, M., Borowa, K., 2022. Additional material for IT managers' perspective on technical debt management. http://dx.doi.org/10.5281/zenodo.7803013.

Wiese, M., Rachow, P., Riebisch, M., Schwarze, J., 2022. Preventing technical debt with the TAP framework for technical debt aware management. Inf. Softw. Technol. http://dx.doi.org/10.1016/j.infsof.2022.106926.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A., 2012. Experimentation in software engineering. In: Experimentation in Software Engineering. Springer Science & Business Media, pp. 1–236.

Yli-Huumo, J., Maglyas, A., Smolander, K., 2016. How do software development teams manage technical debt? – An empirical study. J. Syst. Softw. 120, 195–218. http://dx.doi.org/10.1016/j.jss.2016.05.018.

Marion Wiese is a Research Assistant at the Universität Hamburg. She obtained her Diploma in Informatics at Universität Hamburg in 2001. Her main research interest is technical debt management. She has published three peer-reviewed

papers so far. She lectures Software Architecture, Project Management, and Software Re-Engineering.

From 2001–2020 she worked at Gruner+Jahr GmbH as a software architect and senior developer. She was part of many different projects, like the optimization of a distributed environment's integration architecture, the definition of her company's reference architecture for business intelligence applications, and the development and project management of a subscription management system for magazines.

Klara Borowa is a Research Assistant at the Warsaw University of Technology. She obtained her Master of Science degree in Computer Science at Warsaw University of Technology. Her main research interest is the influence of human factors on software engineering and software architecture. She authored papers published at various venues such as ICSA (International Conference on Software Architecture), ECSA (European Conference on Software Architecture) and IEEE Software. Previously she worked in the software industry at companies such as Hewlett Packard Enterprise, DXC Technology, and Atos.

Chapter 13

Learning From Each Other: How Are Architectural Mistakes Communicated in Industry?

4 *(Learning from each other)*	
Title	Learning From Each Other: How Are Architectural Mistakes Communicated in Industry?
Authors	Wiese M., Brand A.-F., van Hoorn A.
Year	2023
Conference	7th European Conference on Software Architecture (ECSA)
Reference	[WBH23]

TABLE 13.1: Publication No. 4 *(Learning from each other)*



Learning from Each Other: How Are Architectural Mistakes Communicated in Industry?

Marion Wiese(✉), Axel-Frederik Brand, and André van Hoorn

Universität Hamburg, Department of Informatics, Hamburg, Germany
marion.wiese@uni-hamburg.de

Abstract. *Context.* Own experiences and faulty decisions can be an important source of information for software architects. The experiences and mistakes of other architects can also be valuable information sources. *Goal.* Under the assumption that the knowledge about faulty decisions, i.e., mistakes, regarding software architecture is not shared adequately in practice, this work qualitatively investigates the handling and particularly communication of those mistakes by software architects. *Method.* We conducted a grounded-theory study in which we interviewed ten German software architects from various domains. *Results.* We identified software architects' definitions of architectural mistakes, their handling of these mistakes, and their preferred communication strategies regarding these mistakes. We found that architects communicate mistakes mainly within their project teams and seldom within or across companies. *Conclusions.* We derived strategies to make learning and prevention of mistakes more effective. To share experiences and knowledge beyond architects' peer groups, companies should invest more effort in discussing mistakes more consciously and create an environment where mistakes can be discussed openly.

Keywords: Software Architecture · Software Architecture Knowledge · Software Architecture Decisions · Software Architecture Communication

1 Introduction

"Errare humanum est" — to err is human, is common knowledge since ancient times [10]. In software architecture, we also make mistakes. By sharing our experiences with mistakes, we enhance the chance of not repeating them. In other domains, e.g., economics, it has become common to learn from each other's mistakes by sharing experiences, e.g., by conducting "Fuck up nights" [1]. However, in software architecture (SA), it seems that mistakes are not a commonly discussed topic. Even using the term "mistake" in the context of SA may seem offensive.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023
B. Tekinerdogan et al. (Eds.): ECSA 2023, LNCS 14212, pp. 319–336, 2023.
https://doi.org/10.1007/978-3-031-42592-9_22

Technical debts are constructs in software systems that are beneficial in the short term but hinder changes in the long term [2]. Technical debt in SA was identified as one of the most dangerous types [13] because there is a lack of refactoring strategies [3].

One goal to avoid architectural technical debt is to amass knowledge and have at least some experience before deciding on an SA issue. A lot of research focuses on the optimization of SA processes, particularly with regard to SA decision-making [4,21] and gathering SA knowledge [22]. Soliman et al. found that when searching for SA knowledge on Google, experience reports on design decisions and drawbacks of solution options are relevant to a decision but are underrepresented in search results [20].

Architects deal with mistakes, i.e., design decisions that turn out to be sub-optimal,
by doing "post-mortems" [5] or by defining anti-patterns [17], and architecture smells [14]. However, the question of how to communicate the mistakes between architects is not yet researched systematically to the best of the authors' knowledge.

Our goal is to fill this gap by conducting a grounded theory study. We interviewed ten software architects to gather their approaches to sharing bad experiences regarding decisions in SA, i.e., SA mistakes (SAMs). To follow this goal, we answer the following research questions (RQs):

- **RQ 1:** ***How do architects define a SA mistake?*** This question lays the foundation for our work, as the term mistake is not yet commonly used or defined in relation to SA.
- **RQ 2:** ***How do architects manage SA mistakes?*** We explain how the architects identify and handle SAMs. It is relevant to understand the overall management of SAMs to identify communication within these processes. Furthermore, we created a comprehensive model of the SAMs' management.
- **RQ 3:** ***How do architects communicate SA mistakes?*** We identified the ways architects communicate SAMs and particularly on which level, i.e., personal, team-wide, company-wide, or even outside of their company. Furthermore, we identified factors that suppress or promote the communication of SAMs. Thereby, we uncovered opportunities for process improvements.

As the contribution of our study, we answer the research questions and provide:

- A definition of SA mistakes
- A theory of a model of SA mistakes' management
- SA mistakes communication issues and potential for improvement

In Sect. 2, we explain our research methodology. We present the results and answer the research questions in Sect. 3. In Sect. 4, we discuss our findings. We compare our study with related work in Sect. 5 and discuss the threats to validity of our study in Sect. 6. With Sect. 7, we conclude our paper and summarize the contributions for practitioners and researchers. Additional material is available [24].

2 Method

To answer our research question, we did a grounded theory study. Grounded Theory was first introduced by Glaser & Strauss [7]. In grounded theory, the collected data is analyzed interpretatively, and identified concepts are comprehensibly collected. The concepts are compared, structured, and summarized into categories. From these categories, theses are derived that are valid across the data. The final goal of Grounded Theory is to develop a theory on the researched topic. This theory must then be validated by further studies, e.g., surveys. The coding and analysis process takes place in parallel to the collection of new data, whereby the categories are iteratively reconsidered, compared, and questioned. This is referred to as the method of constant comparison. The process of collecting and analyzing data is carried out until the theory is saturated. Saturation is reached when new data hardly generate any new insights in the form of new concepts and categories.

We collected and analyzed data in three phases that merged into one another. In each phase, we adapted the interview guide, did the interviews, and analyzed the corresponding data.

2.1 Data Collection

We conducted ten interviews with architects selected according to our sampling strategy and in accordance with our interview guide. We transcribed the recordings of the interviews for further analysis. All interviews were held in the German language.

Sample. We recruited software architects from our personal network, via social media and speakers from conferences in three phases. All participants completed a survey about their personal background, e.g., domain, company size, and experience level (see [24]). With this information and the iterative approach, we ensured to gather participants from different domains, with different levels of experience, and consulting and company architects. A variety of participants is important to validate that code saturation is reached for the given context, i.e., German software architects. An overview of the participants can be seen in Table 1.

Interview Guideline. We adapted the guideline before each of the three phases to iteratively focus on the emerging theory. All guidelines are available on Zenodo [24] In the end, i.e., in phase three, this led to the following nine questions:

1. What is a SAM or faulty decision in this context?
2. Can you give me examples from your own experience of a SAM or faulty decision? (plus sub-questions regarding the handling of this specific example)
3. What is the role of communicating bad decisions and mistakes for software architects?
4. How do you talk about bad decisions with your colleagues or other architects?
5. How do you talk about SAMs with your team lead or other managers? Does your communication style differ from your colleagues?

Table 1. Study Participants

Participant	*Company Domain*	*Company size (#employees)*	*Team organization*	*Years in IT*	*Years as architect*	*Gender*
P1	Media	1000-5000	agile	>20	5-10	f
P2	Consulting	>5000	agile	10-15	10-15	m
P3	Consulting	>5000	agile	>20	10-15	m
P4	Aviation	>5000	agile	15-20	10-15	m
P5	Industry	>5000	mix (agile/classic)	10-15	2-5	m
P6	Consulting	0-100	agile	15-20	5-10	m
P7	Consulting	100-1000	agile	5-10	5-10	m
P8	Consulting	1000-5000	agile	>20	15-20	m
P9	Consulting	0-100	agile	>20	>20	m
P10	Insurance	>5000	agile	>20	>20	f

6. How do different error cultures (e.g. through in-sourcing/out-sourcing or international work/different companies) influence you in dealing with SAMs or faulty decisions?
7. How are faulty decisions or SAMs dealt with in your company for the purpose of knowledge transfer?
8. How has the way you deal with and communicate mistakes changed over time?
9. With regard to concrete faulty decisions — how and with whom should software architects communicate them?

2.2 Data Analysis

We coded the transcriptions according to Charmaz's guidelines and reached theoretical saturation after ten interviews [6].

Coding. According to Charmaz [6], the process of coding consists of two phases: initial coding and focused coding. *Initial coding* is an inductive approach. Data segments are considered word-by-word and sentence-by-sentence, and all initial analytical concepts are collected without further comparison. *Focused coding* describes hierarchical sorting and abstracting, i.e., merging of concepts or categories as well as the selection of core categories on which the analysis is focused. The process no longer focuses on the raw data but only on the codes.

Furthermore, through cross-coding phases, during which we constantly compared the codes and coded segments, we iterated through every interview multiple times to make sure that the codes were comparable and identical in meaning (Code list see [24]).

Theoretical Saturation. The goal of the methodology is to achieve theoretical saturation. Theoretical saturation means that the evaluation of new data, for

example, through further interviews, no longer fundamentally changes the existing structure of the category system. To evaluate saturation, the number of new codes in each new interview is evaluated, and the goal is to amass fewer codes with each interview's evaluation, particularly focus codes.

As we used cross-coding, the number of codes varied a lot and even got smaller if codes were merged in the process. During our last three interviews, we reduced the codes from 291 codes to 264 codes. During the last interview, we still gained 13 codes (see [24]). However, all these new codes were initial codes and did not provide new insides for the report of the results. Therefore, we assume theoretical saturation for the results presented in this paper.

3 Results

In this section, we give a detailed overview of the definition and characteristics of SA mistakes and how to detect them. We present how architects handle detected SAMs and, finally, summarize how they communicate the detected SAMs. Each section concludes with a summary.

3.1 RQ1: Definition and Characteristics of SA Mistakes

From the transcripts, we extracted the following definition of SAMs and their characteristics.

Definition. A SA decision is classified as a mistake if, for a given problem, an alternative solution exists that solves the problem better than the implemented solution. The choice of an architect is always associated with trade-offs and considerations. Whether a problem is solved "better" is not easy to judge objectively. Therefore, some participants describe SAMs by their negative impact on non-functional quality attributes. *"When it comes to architectural decisions, that [a mistake] is if something doesn't work, I mean more the non-functional [if] you will not meet a non-functional requirement."*

The effects of a SAM are often noticed particularly late, i.e., they have a long feedback cycle. *"The question is, however, after some time, when you reach an appropriate scale, and the application has a corresponding scope, whether you then realize: Yes, you've got it working, but it's not really running ideally."*

SAMs are considered difficult to identify because the alternatives are not directly known to the architects. Even if the architects know the alternatives, the effects of SA decisions are difficult to assess. *"However, you often do not see what an alternative implementation would have brought and what difficulties it would have brought with it."*

SA is an overarching topic because SA should fulfill many different, conflicting goals with many influencing factors and constraints. Therefore, problems are not simply attributable to individual decisions: *"So I think with architecture mistakes, it's often the case that you can't even say that they did something wrong at that point, but it is rather that somehow the realization arises that something is not going right."*

The goal of SA is to meet non-functional requirements. A SAM is defined as the non-fulfillment of these requirements. SA pursues many different and contradictory goals, and alternatives are often not obvious, which makes the evaluation difficult. A wrong decision often becomes apparent only after a long time.

Characteristics. For the participants, a SAM results from the comparison of an actual state with a target state. One participant explained that a SAM is *"everything that turns out afterward that it somehow doesn't work out the way you imagined it."* This comparison between the actual and target state is a subjective evaluation. The subjective approach results in subjective error levels. The situations are classified as acceptable or not acceptable, which in situations with several actors can result in a different understanding of the situation. *"We use our own developed framework, which is already twelve years old because Person C wants it that way, and he gets in the way when we use something else. ... these are not only wrong decisions, this is almost wanton destruction."*.

The term "mistake" is subjectively perceived in different ways, i.e., there are different connotations associated with the term. Some architects mentioned that making SAMs is normal. *"I think making faulty decisions is okay. It can happen, and you have to deal with that somehow."* Some participants also mentioned the positive aspect of making SAMs and learning from them. *"It's that feeling that to do something really well, you have to have failed at it at least once."* However, much more frequently, the participants mention negative aspects of making SAMs: *"However, it is unfortunately often the case that due to the fact that SA [decisions] are the expensive decisions, mistakes very often cannot be communicated ... because then it often looks as if people have wasted a lot of money out of stupidity."*

Another characteristic of mistakes is that they are avoidable. We already established that architects think "making mistakes is normal". This can be interpreted as "making mistakes is unavoidable". Yet, this inevitability only applies to mistakes in general. *"I don't think we could have actually done much differently about the situation. ... That falls under [can be considered as] 'shit happens."'* However, on the level of a single mistake, mistakes are considered to be avoidable. *"If you had trusted the expertise [of the people involved], this wouldn't have happened."*

Finally, another characteristic of a SAM is that we often don't know whether a decision is a mistake until a certain amount of time has passed. *"I think, overall, it came to light too late that this was even a problem at all."*

- A SAM is a subjective evaluation of a situation.
- A SAM may be viewed as acceptable by some and unacceptable by other persons.
- The term SAM can be given positive or negative connotations.
- The single SAM can be seen as avoidable.
- In general, the occurrence of SAMs is regarded as unavoidable.
- A SAM is often recognized retrospectively after a certain period of time.

3.2 RQ2: Management of SA Mistakes

We asked the participants questions about their example situation, particularly how and when they came to the conclusion that the situation was a SAM and how the situation was handled afterward. The results are clustered into four subsequent actions:

1. Detecting SAMs
2. Handling SAMs
3. Learning from SAMs
4. Preventing SAMs

We created a theory of a model to visualize the communication-based interdependencies between these categories and their sub-categories, depicted in Fig. 1.

Detecting SAMs. Mistake detection is typically initiated by a trigger. These triggers can be driven by the architects themselves, e.g., dissatisfaction with the situation or questioning of requirements. *" If I look at it [the decision] like this, to what extent is this decision that was made in management supported by the teams that use it [the system] . . .? There was a moment when I realized: Not at all."* More often, however, triggers from outside were mentioned, e.g., changes in project team structure, functional tests with users, changes in technical conditions, architecture reviews, or new requirements. *"So you suddenly have certain load situations and realize that you have now found the bottleneck. You made a decision back then that now presents you with extreme problems."*

In the next step, the trigger is used to try to understand the situation. It is not always apparent whether there is a mistake or exactly where the mistake is. *"If you notice a problem somewhere and you ask the swarm [all colleagues combined], then the first question is: Where does it come from? What is the cause? And then you come to the conclusion that at some point you made a wrong decision or a wrong assessment."*

To evaluate the situation, the effects of the SAM are analyzed, alternatives are examined, and, if possible, reasons for decisions are reconciled. *"Often the question is: Was this the best, the second best, or the fourth best solution for something? Nevertheless, the question is: Does it work? Or does it not?"*

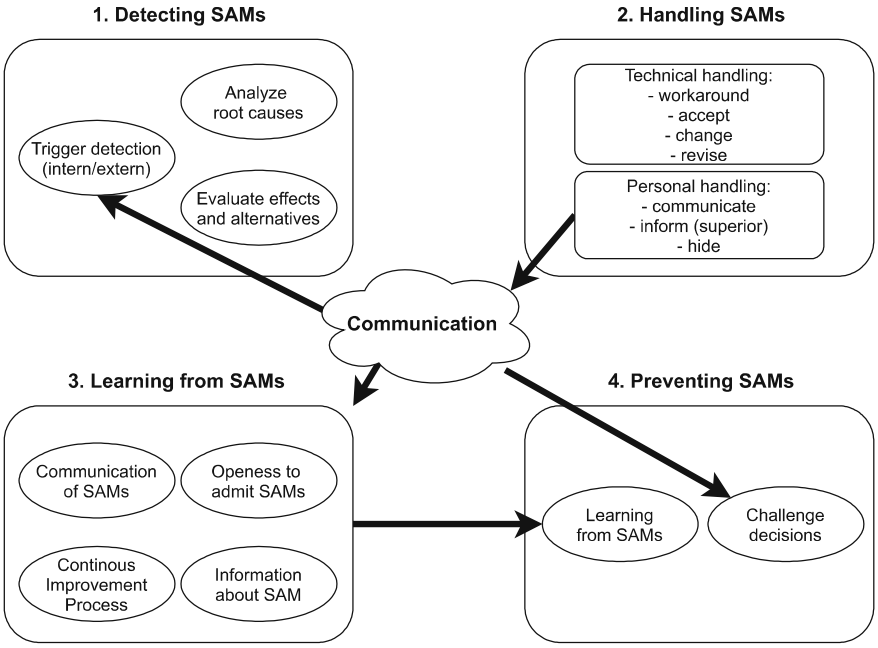


Fig. 1. Theory of a model of SAM's Management

A SAM is detected when a trigger causes the re-evaluation of a situation. These triggers can originate from the architects themselves (internal) but also from external triggers such as changes in the project team, new requirements, functional tests, or external audits. After detection, the situation is analyzed, and alternatives are examined, which triggers a follow-up action.

Handling SAMs. The SAM's handling describes the follow-up action resulting from the evaluation. This corresponds to the technical handling of the SAMs. Other actions are directly performed by the people involved, which corresponds to the personal SAM handling. The core of SAM handling is often the intention to do something different in the future. *"Well, I wouldn't do it that way again. I would try it differently."*

Regarding technical handling, sometimes a workaround can be created to restore functionality. The most common handling approaches are to accept a sub-optimal situation or to develop the architecture further. *"And if it [the SA] does [work], then it is probably more sensible to live with it than to switch everything to the best solution, come hell or high water."* Only rarely do architects revise their architecture completely. *"This project completely failed. We came in there as a different consulting company and had to stop everything, redo everything, and set everything up again."*

As personal SAM handling, all participants report that as a result of a SAM's assessment, the SAM must be communicated, e.g., *"We have highlighted these*

problems over and over again. We have said this again and again." Furthermore, it is important to communicate with superiors to get their support. *"I talk very openly with my direct superiors if I think something is right or wrong. Then we usually think about whether we want to change it or not."* Interestingly, many participants reported that they did not use to communicate SAMs in the past, and this changed over time. *"In the past, I was much more defensive about admitting my own mistakes."* The topic of SAMs' communication will be studied in more detail in the following Sect. 3.3.

SAMs are handled on a technical and personal level. The technical handling comprises creating workarounds and accepting, changing, or revising the SA. Often the sub-optimal solution is accepted with the intention of deciding differently next time. Personal handling comprises communication with colleagues and superiors. The participants conceded their openness to communicate SAMs did develop over time.

Learning from SAMs. We asked the architects if and how they learn from SAMs, i.e., we asked them about the requirements to learn from SAMs.

Firstly, all participants consider that communication is an effective strategy for gaining knowledge from a SAM. They talk about how, in their assessment, a particular situation is a mistake in order to challenge that assessment through the perspective of others. Talking about mistakes means a common evaluation of a situation, the comparison of ideas about an ideal situation, and thinking about further possibilities for action. *"Yes, mistakes need to be talked about. ... It's only through communication and discussion that you come to a good alternative solution that you want to apply."*

Secondly, whether architects are able to learn from SAMs depends on their personal openness to SAMs and the openness of the environment to SAMs. On a personal level, we classified the attitude of all participants as constructive. The participants were open to and aware of their own mistakes without seeing them as a defeat. Similar to the personal level, mutual acceptance of mistakes in the environment is also important The expectation should not be "It is bad to make mistakes" but "It is bad not to learn from mistakes." *"We're all human, we all make mistakes, that's okay. You have to accept that in order to get better. Actually, it's not a mistake, but you learn something from it."*

Thirdly, the participants stated that an established continuous improvement process (CIP) [11] is essential for the learning process. A CIP should specifically try to increase the triggers for SAM detection and make detection a regular process. Furthermore, the SAM's detection is promoted by refining the idea of the ideal solution and the collection of information that can be used in the evaluation. The CIP tries to derive concrete actions that prevent SAMs in the future. *"[After a SAM situation,] I directly asked: What could have been done better? Could this have been foreseen beforehand? What is being done to avoid it in the future?"*

Finally, it is considered important to have the necessary information to classify the SAM situation. For SAMs, the focus is on the context of the decision. What were the reasons for the decision? What were the goals? The documenta-

tion of decisions, e.g., by Architecture Decision Records (ADR) [9], is mentioned but not considered to be widespread. *"I am a big fan of Arc42 and also ADR ... I often start a system design with quality scenarios and quality metrics, especially with new customers. ... I also do a lot of architecture reviews."*

Architects need to communicate the mistake with other stakeholders to be able to learn from SAMs. The ability to learn from SAMs depends on the openness to admit SAMs on a personal and organizational level. An established CIP helps with SAM detection, communication, and information gathering. Important information regarding a SAM are SA decisions and their rationale, which should be documented.

Preventing SAMs. Learning from mistakes is one form of preventing further mistakes. *"Even if it is a mistake, it is, of course, important, especially in the architecture, that you try to prevent it in other areas that might run into this same mistake."*

Furthermore, the architects approach people to challenge their decisions before implementing them. *"We had meetings, I presented the advantages and disadvantages from my point of view, and we did not get together afterward."*

The architects also mentioned other means of preventing SAMs, e.g., taking responsibility for SA, making group decisions, and making more conscious decisions. However, these topics are already discussed in detail in many other research papers.

To prevent SAMs, we should communicate and learn from past SAMs. Challenging SA decisions, i.e., communicating them in advance, is a way of preventing SAMs.

3.3 RQ3: Communication of SA Mistakes

Communication is used as a strategy for SAM detection, for learning from SAMs, and for SAM prevention (see Fig. 1). This section presents the communication partners, communication facilitating and hindering factors, and the structures in which communication takes place.

Communication Partners. All interviewees work in agile project teams, and their most mentioned form of communication is communication on the team level, i.e., with team colleagues. The colleagues are often directly affected by and interested in resolving the situation. It is a common practice in their teams to discuss a solution in a group. A special kind of group communication is discussions in the context of retrospectives. *" This has led to retrospectives in which people have said: 'That's annoying, and it keeps coming back. Can we do something about it or not?' ... Then we had a special meeting about it with the topic: Where are we? Where do we stand? Where do we have to go?"*

The client is also a frequent target of communication, especially due to the need to coordinate further action and the financial impact of decisions. Involving the client in retrospectives or debriefings is perceived as valuable. Likewise, in acute SAM situations, the client is usually involved. In such a situation, some

architects considered communication to be more difficult in some companies. *"[I]f you point out mistakes in a project team with other external people or even with customer employees, particularly in a customer environment where you also have other [consulting] companies, you really have to be very sensitive and really think twice about what you say."*

In the context of SA decisions, communication with direct superiors was not considered to be difficult, e.g., *"I would assess myself in such a way that I can admit mistakes, even to my boss."* One participant even stated: *"My boss gets to listen to a lot. And my boss's boss and my board. So, I'm really there – I have a big mouth. I so don't care."*

The participants rarely mentioned talking about SAMs with architects outside the team. This is only done if necessary by asking specific questions or in the context of an architecture board. *"[T]here is a regular meeting once a month, called the architecture circle. And then all the brains of my employer sit there and give their suggestions."*

Concrete examples of active company-wide communication of SAMs are scarce. Special formats for the presentation of mistakes, such as 'Fuckup Nights,' are mentioned only once. Open spaces and all-hands workshops are also mentioned as spaces with the possibility that SAMs are discussed. *"We do a lot of 'Open Spaces,' including technical 'Open Spaces.' Half of them are not about how cool something is that you built but to look at the mistakes I made."*

Very occasionally, SAMs are discussed with outsiders, e.g., in the form of conference talks or networking events. The personal conviction that SAMs present valuable knowledge is the motivation for this. *"I think I can teach people more when I talk about [SA] mistakes than how to do it right."*

SAM communication takes place on six different levels: with team colleagues, with the clients, with the superiors, with other architects, company-wide, and with outsiders. The most common form is communication within the team. Company-wide discussions of SAMs or communication with outsiders rarely occur.

Communication Facilitating and Hindering Factors. The communication preferences and observed communication patterns can be divided into facilitating and hindering factors. However, some of these factors can be seen as both facilitating and hindering, depending on the architects' characteristics, e.g., professional status and self-confidence, and psychological safety.

The biggest facilitating factor in communicating SAMs successfully is ensuring access to information. This access to information is influenced by the diversity of experiences, i.e., multiple cultural backgrounds but also different domains of IT experience. Diversity ensures that many requirements are represented. *"We have a regular meeting in development. ... Anyone can bring in any topic, be it from testers or other teams. Not development teams, but other teams. That's when a few problems come to light."*

However, this also means a diversity of communication preferences. Responding to these preferences or even being aware of differences between team cultures can further facilitate communication. *"I am also someone who is not afraid of*

speaking in large crowds. There are also people who perhaps have relevant experience who don't want to speak in front of a large crowd right away, where [for whom] you would then perhaps have to find other means to communicate that."

Transparent communication promotes exchange. Having access to or being exposed to SAMs information enables the absorption of information, which means that learning from the mistakes of others is possible. In the office, you inevitably run into other people. In times of remote work, teams need to communicate in open channels rather than in private. *"Otherwise, I often ghost around [roam around] in some channels, i.e., Microsoft Teams channels, in order to also capture topics that are being discussed."*

The biggest hindering factors are potential negative consequences. The communication of SAMs can generate costs, e.g., in contracting situations or raise the question of guilt. *"Unfortunately, it is often the case that software architecture is the most expensive decision and that mistakes are often not communicable."* Additionally, the personal expectation of the outcome of communication can trigger negative consequences in individuals, such as disappointment. *"With younger colleagues, the disappointment is usually great when this [developed SA proposal] decision is not followed."*

The participants also consider the environment to be of decisive importance for the willingness to communicate SAMs. *"If you're in the corporate bubble, then most people ... can admit mistakes and reflect on them. In other environments, I have experienced different situations where people can't admit it [mistakes]."* This is called psychological safety in the domain of psychology, which promotes information sharing [16]. *"[My] Superiors always say that reflection is important; admitting mistakes is important."* A lack of psychological safety hinders SAM communication.

Furthermore, discrepancies among communication participants in reputation, status, or hierarchy seem to make communication more difficult. *"That was a challenge, to do a [meeting] format with a developer from a customer and his board of directors ."*

The participants also mentioned that as one's own professional status and self-confidence grow, the willingness to communicate SAMs increases, particularly in unclear situations. This, however, means that stakeholders with less status and self-confidence might not share their experiences with mistakes. *"So that you can admit mistakes openly, you also need a certain standing. And if you don't feel confident, you won't do it."*

People do not only act rationally, either. The exchange of SAM information is hindered by human characteristics such as emotional bias, e.g., human conflicts may hinder communication. Seeking direct conversation does not work if there is no access or willingness to talk. *"So the question is, as an employee, how do I get the board to listen to me? That's very difficult."*

Finally, communication is complicated by conflicts of interest between the role in the project context and personal goals. Even if there is no fear of dismissal, communicating SAMs could damage one's own reputation or that of the company, no further assignment may follow, or conflicts may arise. *"But some-*

times that is not the right architecture at all. Nevertheless, you might defend it that way. You're not going to admit that mistake and say, 'I did that because it helps me sell my people [consultants]."

Facilitating factors for SAM communication are access to information, i.e., transparent communication and diverse teams in terms of cultures and expertise, as well as supportive communication structures. Hindering factors are potential negative consequences, a missing psychological safety, discrepancies in reputation, status, or hierarchy, a missing professional status and self-confidence, a missing psychological safety in the environment, emotional biases, and conflicts of interest.

Communication Structures and Strategies. The participants' preferred communication structure is to talk directly with others. *" In a team, we are closer to each other, and we sit down together when there are issues."*

While the participants seek direct communication for acute problems, regular appointments are nevertheless a frequently observed pattern for active exchange. These appointments take place regularly and for various purposes. They shape communication not just in response to problems but proactively: *"These classic escalation issues have become fewer because we have managed to have regular appointments. As overarching architects, we have 'Face To Faces' with all the teams ... to sit down and identify and discuss issues."*

During these conversations, participants pursue the strategy of communicating constructively and explaining their own points of view in a well-founded manner. In this context, the participants perceive it as helpful not only to communicate problems and not to create accusations. *"You have to say, 'I don't think this is well designed, for such and such reasons, we have such and such disadvantages.' ... It's not about, 'Peter, you wanted to do it then, and this is bad now."*

Predominantly, architects try to convince their colleagues rather than overrule them. By this, they hope that the other person will adopt the values in the long term rather than simply accept them without understanding them. *"So they might do it then, grudgingly accept it. But they do not understand it. And without understanding, they won't do it any better next time."*

Architects prefer direct communication with colleagues. Regular appointments are important and have the advantage of addressing problems proactively. Communication should be constructive and personal positions should be explained in a well-founded manner. It is important to convince people rather than overrule them.

4 Discussion

In retrospect, the initial hypothesis that SAMs are typically not sufficiently communicated has been only partially confirmed. In the project teams and the peer groups of the participants, there is an active exchange about SAMs. However,

the situation changes on the company level or outside the company. Despite a high level of commitment on the part of the participants, there are only a few examples in which SAMs are discussed outside the involved project teams. In terms of the participants' companies' size, this means that many employees do not have access to the learned information.

Obviously, the processing of information in a larger circle must be approached differently and requires more effort than in a small circle of individuals. The information and facts must be prepared more systematically, and there is less time for individual questions. However, making an information exchange process publicly accessible within the company, e.g., in open spaces or all-hands workshops, would counteract information silos.

Practical indications can be derived from the action strategies of the participants. Regarding the process model presented in Fig. 1, goals can be formulated, which are discussed below.

4.1 Increase Detection Trigger

People and organizations can influence the triggers for SAM detection in the process shown in Fig. 1. More triggers mean more opportunities for the SAMs' detection.

On a personal level, a constructive mindset is most crucial. A different personal attitude may cause SAM detection to be blocked. The point perceived as most important is the change in expectations mentioned in Sect. 3.2. "It is bad to make mistakes" changes to "It is bad not to learn from mistakes."

On a personal and organizational level, formalized and regular CIPs should be established. They can be used to check the validity of SA ideas and implementations and is thus a trigger for SAM detection. The participants report regular architecture reviews or exchanges with architects as one way to achieve continuous improvement, but they require lightweight approaches.

4.2 Ensure Communication Opportunities

Our results show that it is extremely important that SAMs are communicated. Successful communication involves not only talking but also listening. The stigma of failure must be removed so that open discussion, rapid learning, and innovation can occur instead of employees hiding failures in order to protect themselves. Conditions for open discussion should be created, and invitations to participate should be formulated and responded to productively.

The negative connotation of SAMs was discussed in Sect. 3.1. Therefore, some people try to portray SAMs in a positive or euphemistic way. A constructive mindset also means learning from mistakes. However, this should not lead to a euphemistic presentation of SAMs. SAMs, in general, are not positive but have negative consequences and are avoidable. The fact that they also have positive consequences does not make them positive as a whole.

Firstly, these findings can be taken as an impetus for an analysis of one's own handling of SAMs. Secondly, they are also a call for companies to further enable and support SAMs communication among their employees, e.g., by creating company-wide communities of interest. Finally, there should be more open communication about SAMs outside of companies or across companies. For example, governments or chambers of commerce could support this information exchange across companies.

5 Related Work

Our related work section focuses on the main topics of our work, which are SAMs and their communication.

Software Architecture Mistakes. The topic of SAMs is highly related to SA decision-making. A variety of research has been done in this research field, e.g., [4]. However, most of this research focuses on the decision-making process when deciding on an SA or a part of a SA.

Tang et al. did a systematic literature review of the human aspect in decision-making [21]. One of the mentioned decision-making practices is "knowledge management", i.e., knowledge capture, sharing, and communication, which are also important aspects of our work. Yet, all mentioned studies focus on knowledge documentation. While Tang et al. acknowledge that "experience and knowledge play a role in decision making," all identified studies focus on the knowledge aspect [21]. The experience aspect, which includes learning from other people's experiences, is not mentioned. In their semi-systematic literature review, Bhat et al. [4] also identified different studies stating the relevance of experience and knowledge, e.g., [8]. Yet, no work seems to explore how to improve the experience of architects.

Less research has been done on how to deal with bad decisions after they have been made. Some of these aspects are discussed in the research field of architectural technical debt, e.g., [3]. For example, Kruchten et al. present a guideline on how asking about SAMs might lead to the root causes for some architectural technical debts [12]. However, in their recent interview study, Wiese et al. still identified a lack of refactoring strategies for old legacy systems as IT managers' main concern [23].

Software Architecture Mistake Communication. In the field of SA Communication, a wide range of work has been done, e.g., regarding documentation [9], visualization [18], or stakeholder communication [15]. Smolander et al. argue the importance of organizational and inter-organizational communication between architects [19]. However, regarding the communication of SAMs between architects, we were not able to find related studies.

We argue that communication is essential to amass SA knowledge. In their study about SA knowledge in search engines, i.e., Google, Soliman et al. identified a lack of knowledge on bad decisions in the search engine results [20]. Our study supports the idea that this lack is not caused by a faulty search strategy but by

a lack of willingness to share this knowledge. However, we did not find research on if or how communication of SAMs might enhance knowledge gathering.

6 Threats to Validity

We present threats to validity for our qualitative study by addressing the construct, internal and external validity.

Construct Validity. For an interview study, the selection of participants and interview questions have a great impact on the validity of the study. In our study, we only interviewed architects working in an agile environment. As we did not enforce this as a precondition, we suspect that an agile team organization is state-of-the-art. We, therefore, suggest using our study's results specifically for agile working companies.

Furthermore, architects that are more open to discussing their SAMs might be more willing to participate in our study, which might bias our study. We also recruited participants from our personal network. These participants might not be as open regarding their SAMs but might still participate for our sake.

Finally, we interviewed ten architects, which might seem too few. However, we reached code saturation on the focus code level, i.e., codes that were reported in this paper did not change during the last three interviews. Further studies in other countries and with more participants, e.g., surveys, should validate our theories.

Internal Validity. In the case of our study, internal validity means the possibility of wrongly interpreting and misunderstanding the interview transcripts. To avoid such misconceptions, the coding was done by two researchers. Furthermore, we were able to ask the participants for clarification in cases we were unsure about a statement.

External Validity. External validity concerns the generalization of the result. However, the ability to generalize the results is not the goal of a qualitative study such as ours. Nonetheless, we only surveyed German architects, and architects in other countries might have different insights, particularly as cultural aspects are important in our study. The generalizability of our results could be enhanced by further studies, e.g., by surveys generating quantitative data and by interview studies in other countries.

7 Conclusion

In our study, we analyzed the communication of SAMs in the industry by performing an interview study with ten architects. We derived their definition and list of characteristics of SAMs, their handling of such SAMs after their detection, and finally, their ways of communicating SAMs. We identified optimization potential regarding the enhancement of SAM's detection triggers.

While the communication of SAMs takes place on the team level, we identified optimization potential on a company and cross-company level. Furthermore, we identified communication partners, structures, and strategies, and we presented communication facilitating and hindering factors.

For practitioners, our study could spark the company-wide exchange of SAMs. Furthermore, we provided many ideas on how to optimize communication on a personal level, e.g., by visiting public communities of interest and changing our own mindset from "It is bad to make mistakes" to "It is bad not to learn from mistakes."

For researchers, we provided the theory of model on how practitioners manage SAMs. This, in turn, forms the question of how to decide whether to accept a SAM, change the architecture, or revise the architecture completely. Furthermore, we identified the research question on how to optimize the communication of SAMs on the enterprise level besides the ideas mentioned by our participants. For these ideas, an evaluation of their effects is also still missing.

For future work, we suggest performing a subsequent survey study to validate our theories and refine them by taking the different contexts, e.g., domains, countries, and cultural environments, into account. In addition, this could help to generalize our results. After that, the aforementioned research questions on how to optimize SAM communication should be addressed by solution-seeking studies.

References

1. Fuckup Nights (2023). https://en.fuckupnights.com/
2. Avgeriou, P., Kruchten, P., Ozkaya, I., Seaman, C.: Managing technical debt in software engineering. Dagstuhl Reports **6**(4) (2016)
3. Besker, T., Martini, A., Bosch, J.: Managing architectural technical debt: a unified model and systematic literature review. J. Syst. Softw. 135 (2018)
4. Bhat, M., Shumaiev, K., Hohenstein, U., Biesdorf, A., Matthes, F.: The evolution of architectural decision making as a key focus area of software architecture research. In: Proceedings of the IEEE 17th International Conference on Software Architecture, ICSA 2020 (c) (2020)
5. Bjørnson, F.O., Wang, A.I., Arisholm, E.: Improving the effectiveness of root cause analysis in post mortem analysis: a controlled experiment. Inf. Softw. Technol. **51**(1) (2009)
6. Charmaz, K.: Constructing grounded theory. Sage Publications (2014)
7. Glaser, B.G., Strauss, A.L.: The discovery of grounded theory: strategies for qualitative research. Nurs. Res. **17**(4), 364 (1968)
8. Hassard, S.T., Blandford, A., Cox, A.L.: Analogies in design decision-making (2009)
9. van Heesch, U., Avgeriou, P., Hilliard, R.: A documentation framework for architecture decisions. J. Syst. Softw. **85**(4) (2012)
10. Hieronymus, S.E.: Errare humanum est. In: Seneca, Epistulae morales VI,57,12; Cicero, Orationes Philippicae 12,2, vol. VI,57,12 (2023)
11. Imai, M.: Kaizen. No. 201, Random House Business Division, New York (1986)

12. Kruchten, P., Nord, R., Ozkaya, I.: Managing technical debt: reducing friction in software development. Carnegie Mellon University, Software Engineering Institute (2019)
13. Martini, A., Bosch, J.: The danger of architectural technical debt: contagious debt and vicious circles. In: Proceedings of the 12th Working IEEE/IFIP Conference on Software Architecture, WICSA2015 (2015)
14. Mumtaz, H., Singh, P., Blincoe, K.: A systematic mapping study on architectural smells detection. J. Syst. Softw. 173 (2021)
15. Márquez, G., Taramasco, C., Astudillo, H., Zalc, V., Istrate, D.: Involving stakeholders in the implementation of microservice-based systems. IEEE Access **9** (2021)
16. Newman, A., Donohue, R., Eva, N.: Psychological safety: a systematic review of the literature. Hum. Resour. Manage. Rev. **27**(3) (2017)
17. Richards, M.: Software architecture patterns. O'Reilly Media (2015)
18. Shahin, M., Liang, P., Babar, M.A.: A systematic review of software architecture visualization techniques. J. Syst. Softw. **94** (2014)
19. Smolander, K., Päivärinta, T.: Describing and communicating software architecture in practice: observations on stakeholders and rationale. In: Pidduck, A.B., Ozsu, M.T., Mylopoulos, J., Woo, C.C. (eds.) CAiSE 2002. LNCS, vol. 2348, pp. 117–133. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-47961-9_11
20. Soliman, M., Wiese, M., Li, Y., Riebisch, M., Avgeriou, P.: Exploring web search engines to find architectural knowledge. In: International Conference on Software Architecture (ICSA2021) (2021)
21. Tang, A., Razavian, M., Paech, B., Hesse, T.M.: Human aspects in software architecture decision making - a literature review. In: International Conference on Software Architecture (ICSA2017) (2017)
22. Weinreich, R., Groher, I.: Software architecture knowledge management approaches and their support for knowledge management activities: a systematic literature review. Inf. Softw. Technol. **80** (2016)
23. Wiese, M., Borowa, K.: IT managers' perspective on technical debt management. J. Syst. Softw. **202** (2023)
24. Wiese, M., Brand, A.-F., van Hoorn, A.: Additional material for learning from each other: how are architectural mistakes communicated in industry? (2023). https://doi.org/10.5281/zenodo.7924700

Chapter 14

The TechDebt Game - Enabling Discussions about Technical Debt

	5 *(TD Game)*
Title	The TechDebt Game - Enabling Discussions about Technical Debt
Authors	Wiese M., Heinrichs A., Rusieshvili N., de Almeida, R. R., Borowa K.
Year	2025
Conference	IEEE/ACM International Conference on Technical Debt (TechDebt)
Reference	[Wie+25]

TABLE 14.1: Publication No. 5 *(TD Game)*

The TechDebt Game - Enabling Discussions about Technical Debt

Marion Wiese*,
Angelina Heinrichs,
Nino Rusieshvili
Department of Informatics,
University of Hamburg
Hamburg, Germany
*marion.wiese@uni-hamburg.de

Rodrigo Rebouças de Almeida
Federal University of Paraíba
Department of Exact Sciences
Rio Tinto/PB, Brazil
rodrigor@dcx.ufpb.br

Klara Borowa
Warsaw University of Technology,
Institute of Control and
Computation Engineering
Warsaw, Poland
klara.borowa@pw.edu.pl

2025 IEEE/ACM International Conference on Technical Debt (TechDebt) | 979-8-3315-0221-8/25/$31.00 ©2025 IEEE | DOI: 10.1109/TechDebt66644.2025.00010

***Abstract*—*Context.* Technical Debt (TD), defined as software constructs that are beneficial in the short term but may hinder future change, is a frequently used term in software development practice. Nevertheless, practitioners do not always fully understand its definition and, in particular, conceptual model. Previous research highlights that communication about TD is challenging, especially with non-technical stakeholders. Discussions on this topic often cause conflicts due to misunderstandings related to other stakeholders' perspectives. *Goal.* We designed a board game to emulate TD concepts to make them tangible to all stakeholders, including non-technical ones. The game aims to encourage discussions about TD in an emulated and safe environment, thereby avoiding real-life conflicts. *Method.* To evaluate the game's effectiveness, we surveyed 46 practitioners from diverse domains, positions, and experience levels who played the game in 13 sessions following extensive testing during its development. In addition to the players' general feedback, we examined situations where players recognized new insights about TD or connected game scenarios to real-life experiences. *Results.* Overall, the feedback on the game and its enjoyment factor were highly positive. While developers and software architects often connected game situations to their real-world experiences, non-technical stakeholders, such as scrum masters, product owners, and less experienced developers, encountered multiple new insights on TD. Numerous players have shifted their attitudes toward TD and have outlined a plan to modify their behavior regarding TD management. *Conclusions.* Although the game may not lead to long-term behavior change among stakeholders, participants' feedback provides evidence that it might serve as a valuable starting point for team discussions on technical debt management.**

***Index Terms*—Technical Debt, Technical Debt Management, Game-Based Learning**

I. INTRODUCTION

Avgeriou et al. defined Technical debt (TD) as a result of a trade-off decision for IT constructs between short-term beneficence and long-term maintainability [1], [2]. These constructs can occur in various elements of the system or software engineering process, e.g., code, tests, or requirements [3]. There is a substantial body of work on how TD and its related concepts should be modeled, named, and described [2]–[4], e.g., a single instance of TD is named a TD item [2]. Avgeriou et al.presented an overall conceptual model of TD [1], which has been used as a basis for various research, e.g., [5]–[7]. As part of the InsighTD project [8]–[10], the main causes and consequences of TD have been extensively described.

Initially, W. Cunningham employed the term Technical debt (TD) as a metaphor to explain to business stakeholders how fast delivery of makeshift software, results in the need for code rewrites [11] . Communicating the TD concepts to business stakeholders is still challenging, as current research shows [7], [12]–[14]. The communication on the TD topic poses the risk of misunderstandings and unproductive discussions between different stakeholders [15]. Moreover, the vision paper on TD management highlights the relevance of socio-technical aspects of TD [14]. Hence, this study's **rationale** is to enable and improve communication between the stakeholders involved in creating IT systems and harmonize their knowledge level about TD.

For this purpose, we designed a board game in an offline and online version to imitate the TD concepts in an emulated environment with the **goal** of enabling safe and engaging communication free of real-life conflicts between stakeholders of various backgrounds, including technical and business stakeholders. The game was designed along what we called *aha*-moments, i.e., recognition moments where players could identify new insights about TD or relate game scenarios to their real-life experiences. We intended the game as a starting point to discuss TD rationally and, ultimately, set up a TD management process within a team or company.

A total of 46 stakeholders from diverse domains, experience levels, and positions played the TechDebt Game in 13 game sessions of usually four stakeholders each. The games were played in Brazil, Poland and Germany. We evaluated the outcome by surveying the players after each game, asking for feedback, determining their motivation to change their behavior, and specifically, assessing the *aha*-moments.

Our **research questions (RQ)** focus on evaluating the effectiveness of using a board game to teach TD concepts, stimulate discussions among practitioners, and provide an enjoyable and psychologically safe environment for exploring and discussing technical debt.

979-8-3315-0221-8/25/$31.00 ©2025 IEEE
DOI 10.1109/TechDebt66644.2025.00010

Authorized licensed use limited to: Bibliothekssystem Universitaet Hamburg. Downloaded on August 26,2025 at 10:01:46 UTC from IEEE Xplore. Restrictions apply.

RQ 1: To what extent does the game emulate the real-life experiences that technical stakeholders encounter? To be meaningful to all stakeholders, the game must provide realistic elements to provide genuine insights while remaining abstract enough to ensure an engaging flow. Thus, we asked the players if they had experienced situations related to the game and how enjoyable they found playing it.

RQ 2: Do the players encounter new insights and learn about additional facets of TD management? Since one of the game's goals is to educate stakeholders about various TD-related concepts to harmonize the level of knowledge between stakeholders, we questioned the players about which *aha*-moments were previously unfamiliar to them and how these offered new perspectives on TD.

RQ 3: How does the game change players' attitudes and possibly behaviors regarding TD? Finally, the game is set up to stimulate discussions on embedding TD management into the player's real-world workflows, e.g., their software development life cycle. Thus, we asked about the players' attitudes towards TD and plans to change their behavior regarding TD management after they played the game.

As the **contributions** of our study, we provided a strategic board game built on the theoretical foundation of game-based learning that emulates the real-life experiences of developers regarding TD. We demonstrated that the TechDebt Game can be used to educate stakeholders, foster communication on TD management, and support discussion about TD management strategies. Particularly, business stakeholders and junior developers profited from new insight. We revealed that the game might encourage a player's behavior change, which provides a starting point for embedding TD management if played in a team working together on one IT system.

In the following section, we compare our study with related work (Section II). We present the theoretical background on game-based learning that forms the basis for understanding the game design in Section III and the game design itself in Section IV. We explain the game's evaluation approach in Section V, and the respective results in Section VI. Afterward, we discuss our findings in Section VII and the threats to the validity of our study in Section VIII. With Section IX, we conclude our paper, summarize the contributions for practitioners and researchers, and identify future work. The additional material comprises the TechDebt Game and details on the evaluation [16]. The TechDebt Game is also available online [**?**].

II. RELATED WORK

Our related work section focuses on game-based learning approaches used in software engineering and, particularly, TD management.

Problems and Programmers is a card game on the subject of software development designed by Baker et al. [17]. Due to the small number of participants, the findings only indicate that, according to the players' subjective opinions, the game facilitates collaborative learning.

PlayScrum and GetKanban v4.0 teach frameworks for agile software development. Fernandes et al. evaluated PlayScrum with 13 master's students using questionnaires [18]. The players rated it as entertaining and effective for learning Scrum concepts. GetKanban v4.0 was evaluated against predefined learning objectives [19]. Heikkilä et al. found that the game was motivating but had no significant effect on the knowledge transfer of Kanban methods.

DecidArch (v2) and SmartDecisions are board games that teach decision-making in software architecture. DecidArch learning objectives are (1) the rationale of design decisions, (2) understanding the diversity of solutions, and (3) the impact of changing design decisions [20]. The game was evaluated with students in two game iterations to improve the game design. SmartDecisions focuses on the "Attribute-Driven Design" method (see [21]) and was evaluated with 41 participants, including students, educators, and practitioners, employing feedback forms and game results [22].

Hard Choices is the only empirically evaluated board game designed to teach the concept of TD [23]. In this game, players compete against each other to be the first to bring a product to market. Players can skip some development phases by incurring TD in the form of bridge cards evaluated as minus points in the final calculation. Ganesh et al. tested the game with 41 IT students using pre- and post-tests and a survey. The results revealed a significant improvement in the understanding of TD after the game.

Hard Choices is the only game similar to our game as it focuses on TD concepts. However, the main purpose of this game and its evaluation is to educate IT students. Our game is designed to foster not only learning but discussion among and behavior change in practitioners, including non-technical stakeholders, and was evaluated using players from practice.

Other educational games on the topic of TD, such as the "Technical Debt Game" [24] and the "Technical Debt Game-for non-technical people" [25] were not evaluated, and thus, their effectiveness has not been determined.

III. THEORETICAL BACKGROUND

In this section, we provide the theoretical background on game-based learning and game design that forms the basis for understanding the design approach of our game.

A. Game-based Learning

Game-based learning is a learning method in which teaching content is passed on to the learners by playing a game. Suits et al. define a game as an interactive activity to achieve a goal, which is restricted by rules [26]. In educational games, the game aims to impart knowledge through the activity of playing. Some game definitions provide greater detail regarding the presence of a conflict, which plays a prominent role in cooperative and competitive games [27], [28].

In their article on the basic laws of game-based learning, Plass et al. defined the four most fundamental arguments

Authorized licensed use limited to: Bibliothekssystem Universitaet Hamburg. Downloaded on August 26,2025 at 10:01:46 UTC from IEEE Xplore. Restrictions apply.

for using game-based learning [28]: *Motivation* of players has been proven to be increased by game-based learning compared to learners participating in conventional learning activities [29]. The increased motivation has, in turn, improved the learning outcomes [30]. *Player engagement* is the player's willingness to engage with the game and its content. *Adaptability* describes the flexible adjustment of the difficulty to the player's abilities at the game's runtime. *Graceful failure* is a frequently occurring advantage of Game-based learning and describes the concept of consequence-free failure. Consequence-free refers to the fact that all effects of failure do not leave the virtual environment of the game. This allows players to experiment with the game, e.g., exploring a style of play that contrasts with their real-life approach.

B. Game Design

Educational game elements. While not every game leads to improved learning outcomes, there are some game elements that can improve outcomes regarding educational content [31]. Thus, it is integral to defining the individual game element categories to transfer real-life knowledge. Plass et al. define the following game element categories [28]:

- *Game mechanics* are the player's interactions with the game, including every playful activity in which the game's rules are followed. Game mechanics can either teach content and skills directly or provide feedback for other actions.
- *Aesthetic design* describes the presentation of information and game concepts. An appealing and, at the same time, informative visual design is relevant to all game elements, including cards, game boards, or even coins.
- *Narrative design* is crucial for creating the virtual environment. In board games, texts can tell a story and embellish the virtual environment. Narratives are an important medium for passing on theoretical knowledge.
- *Incentive systems* are essential for increasing player motivation. Intrinsic motivation can be achieved by promoting enjoyment and, thus, increasing the desire to play the game. Extrinsic motivation can be reached by fostering competition outside of the specific game environment, e.g., by awarding points for a final tally.
- The *musical value* of a game refers to the management of attention and mood through the use of music, background noise, or auditory cues.
- *Content and skills* formulate the learning objectives that shall be acquired with the game. Theoretical content and skills in a subject form the basis for good game elements.

Creation of an educational game. Pivec et al. outlined the following steps for creating educational games [27]:

(1) *Learning approach.* Before starting the development process, the theoretical foundation of the learning objectives must be defined. The content and the approach to the previously described game element category "content and skills" are determined. (2) *Creating a model world.* The medium and genre of the game are determined, which has a direct impact on the other game elements. The specification of the model world provides the framework for the rest of the game. (3) *Executing the details.* The theoretical foundations and the model world are brought together through narrative design. The first elements of knowledge transfer can be incorporated. (4) *Incorporating learning support structures.* Clear goals that the player should achieve and appropriate consequences for pursuing these goals are defined. (5) *Transforming learning activities into interactions with the game interface.* The game mechanics, e.g., drawing cards, rolling dice, or moving pieces, for solving the game's underlying conflict are defined. (6) *Transforming concepts into interface objects.* The perceptible objects of the game are created, including visual representations of the game board, cards, or game resources (aesthetic design), background music (musical value), and points and achievements (incentive system).

IV. TECHDEBT GAME

The TechDebt Game is a board game that aims to playfully increase understanding and discussion of TD by emulating the software development process. Two teams of two players compete against each other for a predefined duration (e.g., 60 minutes) to develop a system by figuratively working on software architecture and features. As the team completes tasks, they accumulate users (the game's points). Although taking on technical debt may speed up task completion in the short term, it ultimately makes future tasks more difficult to accomplish and results in a lowered score by the end of the game. The team that accumulates the most users (points) within the duration of the game wins. Players can employ different strategies to manage TD and encounter the respective results (e.g., "gracefully failure"). The two-on-two format encourages discussion among team members, creating a collaborative dialogue where all participants gain from exchanging and learning from each other's insights. Figure 1 shows the TechDebt Game board. The game is available as a PDF and must be augmented with dice and game pieces. The game PDF and manual are part of the additional material [16].

A. Game design, elements and mechanics

- *Game board:* The game board represents the progress of a project and the placement of the modules. Each team builds one system consisting of three modules (A-C).
- *Architecture tickets and feature tickets:* To start a module, an architecture ticket has to be completed and placed in the first row of the system. After that, completed feature tickets can be placed on the subsequent rows. Feature tickets earn points, which are represented by the users gained through a feature. Each ticket shows the six digits of a dice, some of which are blocked, i.e., crossed out by a red line on the ticket's respective dice symbol (see Figure 2). Two dice are rolled to "work" on a ticket. If one of the rolled digits is not blocked, progress is made, i.e., one task of the ticket is completed. The number of tasks required to complete the ticket is shown with empty circles.
- *TD incurrence:* TD can be incurred consciously by adding TD to a dice blocked for the ticket, resulting in immediate

Authorized licensed use limited to: Bibliothekssystem Universitaet Hamburg. Downloaded on August 26,2025 at 10:01:46 UTC from IEEE Xplore. Restrictions apply.

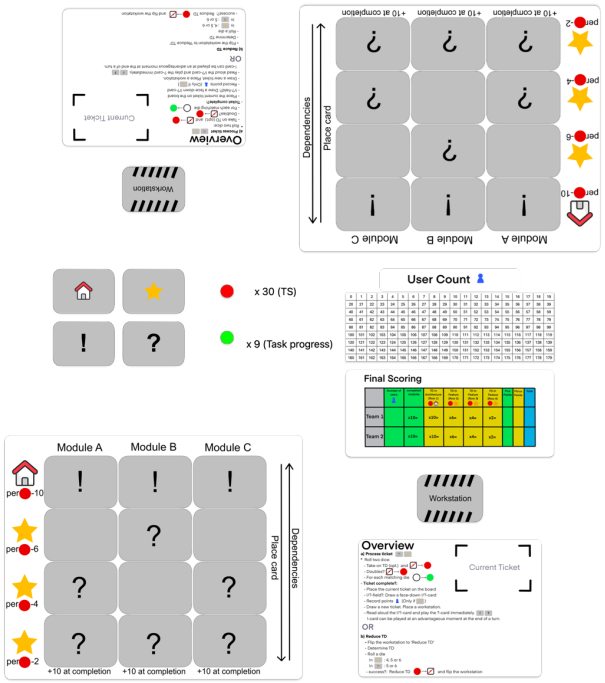


Fig. 1: Overview of the Game Board (details see [16]

success in completing one task if that digit was rolled. TD can be incurred unconsciously by throwing a double. Then, TD must be incurred at the respective digit, and two tasks will be completed immediately.

- *Workstation/TD Repayment card:* This card is placed at the location where the completed ticket will be laid out to identify dependencies on other tickets. To repay TD, the card has to be flipped over to the TD repayment side, and the team has to dedicate a turn specifically for the repayment. To repay TD on feature tickets, a four, five, or six must be rolled. As repaying architectural TD is harder, only rolling a five or six leads to a successful repayment. This means they are investing time that could have been spent on another task, representing the cost of addressing and repaying the TD.
- *TD dependencies:* The current ticket depends on all previous tickets in the same module, which means that TD from previous tickets hinders the current ticket's development. Each digit that contains a TD on one of the depending tickets is blocked when working on the current ticket, i.e., if the current ticket would allow a roll of six to be successful, but a previous ticket contains a TD on a six, the six is no longer available to complete a task. Figure 2 shows an example where the tasks of the last ticket can only be completed by rolling a six.
- *Event and action cards:* Event cards (question marks) have immediate negative effects and represent the causes and consequences of TD. Action cards (exclamation marks) can be used strategically and represent positive TD management actions.

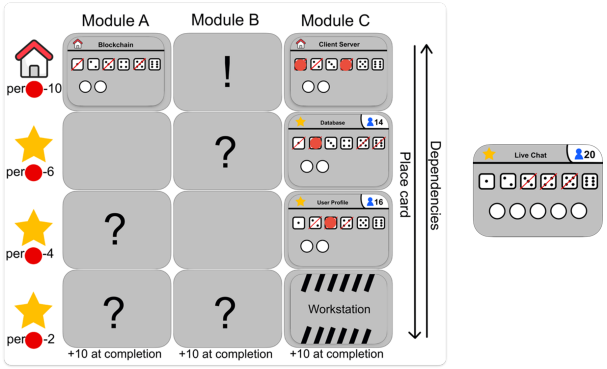


Fig. 2: Example for dependencies created by incurring TD (red tiles) – only rolling a six will complete tasks

B. Game Design Iterations

We designed the game in three main iterations, each focusing on one topic: (1) set up the initial game design, (2) simplify the game by removing non-TD-related game elements like the scrum cycle, (3) digitalizing the game and adapting the board game so both fit the requirements of digitalization. For iterations 1 and 2, we played nine games overall. During the second iteration, we optimized the game along pre-defined learning objectives, which we derived from current literature on TD, e.g., [1], [2], [7]. We referred to those learning objectives as *aha*-moments, a term that reflects the expression people often use when they recognize or understand something new. Table I gives an overview of all *aha*-moments and their descriptions. During the second iteration, each game was recorded, transcribed, and coded with the *aha*-moments. The *aha*-moments were then counted to ensure they could be encountered by the players.

Furthermore, small changes were made throughout all iterations and sometimes between games, optimizing the game flow, the rules, and the introduction. For example, we simplified the game process by removing the prioritization between various feature tickets. While this prioritization is a real-life task that happens regularly as part of the software development life cycle, it prolonged the game without changing the player's perception of TD.

C. Summary

We consider that the TechDebt Game meets the key features of an educational game [28] as described in Section III:

- *Game mechanics:* Activities like rolling dice ("working"), placing TD tiles, integrating tickets into the system, or drawing action and event cards form the game's basis.
- *Aesthetic design:* The game features a minimalist design emphasizing key functionalities by using specific colors and symbols, such as red tiles for TD or a house to represent architecture.
- *Narrative design:* The game follows a straightforward narrative: A system comprising features and software architecture has to be built, and whoever develops a better system

Authorized licensed use limited to: Bibliothekssystem Universitaet Hamburg. Downloaded on August 26,2025 at 10:01:46 UTC from IEEE Xplore. Restrictions apply.

TABLE I: *Aha*-moments and their explanations (causes and consequences are taken from Wiese et al. [7])

Group	Variable	Description
Causes	Time	Time pressure can be a cause of TD (deadlines).
	Budget	Cost pressure can be a cause for TD (license costs).
	Business	Business decisions can be a cause of TD (change in requirements, change in strategy).
	Management	Management decisions can be a cause of TD (broken communication, poorly planned projects).
	Personnel	Personnel can be a cause of TD (lack of personnel, inexperienced or unmotivated personnel, frequent changes).
	Technology	chosen technology can be a cause of TD (outdated technology).
	Decisions	Incorrect decisions can be a cause of TD (architectural decisions).
	Awareness	Lack of awareness of TD can be a cause of TD.
	Chains	Causes of TD can trigger other causes of TD.
Incurrence	Conscious	TD can be incurred consciously.
	Unconscious	TD can be incurred unconsciously.
Consequences	Time	TD can lead to more time expenditure (overtime, missed deadlines, longer development process).
	Budget	TD can lead to higher costs (optimization costs, project becomes more expensive).
	Business	TD can negatively affect the business (unmet requirements, loss of customers, legal consequences).
	Management	TD can negatively affect management (lack of controllability, future risk).
	Personnel	TD can lead to personnel problems (terminations, stress, new developers having to be trained).
	Technology	TD can lead to technology problems (maintainability, bugs, dead-end).
	Chains	Consequences of TD can trigger further consequences of TD.
Vicious Cycle	Inner	TD can lead to further TD (broken window phenomenon).
	Outer	Consequences of TD can become causes for new TD.
Repayment	Difficult	Paying back TD is difficult.
	Time-consuming	Repaying TD is time-consuming.
	Benefits	The repayment of TD can create advantages for further development.
	Simplified	Certain measures make it easier to repay TD (refactoring, engaging specialists, communication).
Architecture	Critical	TD items in architecture are the most critical debts.
	Hard to repay	TD items in architecture are the hardest to repay.
	Prevents TD	Architecture can help deal with TD.
TD management	Identifying TD	To fix TD, they must first be detected.
	Prioritizing TD	To make decisions, TD must be prioritized.
	Ignoring TD	It is not always reasonable to fix (all) TD.
Business	Invisible	TD are invisible in themselves and can only be recognized through symptoms.
	Perspective	Causes and consequences of TD can be difficult to discern from a business/management perspective.

with more users wins. The predefined duration adds time pressure that motivates the competition and might lead to an accumulation of technical debt. Using TD in the system can speed up the development of the current ticket but increases the difficulty of completing dependent tickets.

- *Incentive system:* Points and visible progress in the game encourage competition with the opposing team. Action and event cards act as rewards and punishments.
- *Musical value:* As this was originally a board game, there is no background music. Additionally, music could have drowned out the discussion between players.
- *Content and skills:* The learning objectives, i.e., *aha*-moments shown in Table I, served as guiding principles for the game's development.

The game is available as a PDF file as part of the additional material [16] and for future development on GitHub [32]. It must be augmented with materials like dice, or game stones. A digital version is available online [33].

V. METHOD

In this section, we present how we evaluated the TechDebt Game by playing it with IT stakeholders and surveying them.

A. Sample

A total of 46 practitioners played the game in 13 game sessions, of which nine were played online. Four games were played on-site (using the PDF version), by players who work together as a team or collaborate within their company, aligning with the study's initial goal to foster discussion on embedding TD management processes in a team. For the nine

TABLE II: Participants (PM/PO - project manager/ product owner)

Game	location	Country	Domain Domain	Position	Experience (in yrs.)
1	on-site	GER	Electronics	Developer	6 - 10
1	on-site	GER	Electronics	IT-Manager	3 - 5
1	on-site	GER	Electronics	Developer	0 - 2
1	on-site	GER	Electronics	Architect	>10
2	on-site	GER	Electronics	Developer	3 - 5
2	on-site	GER	Electronics	Developer	>10
2	on-site	GER	Electronics	Developer	0 - 2
2	on-site	GER	Electronics	Developer	6 - 10
3	on-site	GER	Manufacturing	Developer	>10
3	on-site	GER	Manufacturing	PM/PO	>10
3	on-site	GER	Manufacturing	Developer	>10
3	on-site	GER	Manufacturing	Developer	0 - 2
4	on-site	GER	Insurance	Developer	0 - 2
4	on-site	GER	Insurance	Developer	3 - 5
4	on-site	GER	Finance	Developer	0 - 2
4	on-site	GER	Insurance	PM/PO	>10
5	online	GER	Transportation	IT-Manager	>10
5	online	GER	IT Consulting	Business	3 - 5
5	online	GER	IT Consulting	Architect	>10
5	online	GER	Telecommunic.	IT-Manager	>10
6	online	GER	Finance	PM/PO	3 - 5
6	online	GER	IT Consulting	Developer	3 - 5
6	online	GER	Finance	Developer	3 - 5
6	online	GER	Finance	Developer	0 - 2
7	online	GER	Energy	PM/PO	>10
7	online	GER	E-commerce	Developer	0 - 2
7	online	GER	E-commerce	Developer	3 - 5
7	online	GER	Pharmacology	IT-Manager	>10
8	online	BRZ	E-commerce	Developer	0-2
8	online	BRZ	E-commerce	Developer	0-2
8	online	BRZ	Academic	Architect	0-2
8	online	BRZ	Government	Developer	0-2
9	online	BRZ	IT Consulting	PM/PO	>10
9	online	BRZ	E-commerce	Developer	>10
10	online	BRZ	Finance	Developer	6 - 10
10	online	BRZ	E-commerce	Developer	0-2
10	online	BRZ	E-commerce	Developer	0-2
10	online	BRZ	Finance	IT-Manager	>10
11	online	BRZ	Finance	PM/PO	>10
11	online	BRZ	Finance	User	>10
12	online	BRZ	Finance	PM/PO	>10
12	online	BRZ	Finance	PM/PO	>10
12	online	BRZ	Finance	PM/PO	>10
13	online	PL	Ecommerce	Architect	>10
13	online	PL	Finance	IT-Manager	>10
13	online	PL	Electronics	Developer	6 -10

Authorized licensed use limited to: Bibliothekssystem Universitaet Hamburg. Downloaded on August 26,2025 at 10:01:46 UTC from IEEE Xplore. Restrictions apply.

online games, we used our personal network and assembled interested players promoting the game on social networks. Seven games were played in Germany, five in Brazil, and one in Poland. In two of the games, two participants played as a team against a researcher, and in two other games, a researcher played alongside one practitioner against other practitioners. We included these games in the study since they represent sessions that may occur in real life, e.g. some players may prefer to play with an outsider experienced with the game, or they may struggle to find exactly four players.

Ultimately, the players comprised one customer, one user, nine project or product managers, six IT managers, four software architects, and 25 developers. Only four participants had not heard of the term TD before the game. An overview of all players is given in table II.

To compare the effect the game had on different stakeholder types, we categorized the players into three stakeholder categories:

- *Business stakeholders:* Business management or customers, users, and project managers or product owners as their deputies in software development projects.
- *Junior technical stakeholders:* Developers, architects, and IT managers with up to 5 years of experience.
- *Senior technical stakeholders:* Developers, architects, and IT managers with over 5 years of experience.

Of the 35 technical stakeholders, 17 were senior stakeholders with more than 5 years of experience.

B. Data collection

Each study's session comprised four steps:

- a 15-minute long introduction to the game and its rules.
- a 15-minute test game to understand the rules and develop a strategy
- 60 minutes or until one team has completed all three modules to play the (main) game
- 15 minutes to fill out the questionnaire

For two games, the time for the main game had to be reduced to 30 minutes for organizational reasons. We recorded each main game session and transcribed the recordings to allow us to explore intriguing discussions.

Additionally, each participant answered a questionnaire after the game. First, the questionnaire asked for demographic information on the participant, e.g., domain or experience. Second, we asked about each of the *aha*-moments (see Table I) that guided the game design. For each *aha*-moment, the participants could choose one of the following options:

- "I had the AHA moment, but the realization is NOT NEW to me."
- "I had the AHA moment, and the realization is NEW for me."
- "I didn't have the AHA moment, but the realization is NOT NEW for me."
- "I didn't have the AHA moment, and the realization would be NEW for me."

Third, we asked whether the game changed the participant's attitudes and whether they plan to change their behavior regarding TD using two open questions. Finally, we asked for feedback on the game and ideas for its improvement.

C. Data Analysis

We quantitatively analyzed the questionnaire's data on the *aha*-moments using Excel and created descriptive statistics for the results. We do not expect all participants to experience all *aha*-moments. Depending on experience and background, different topics might be relevant to the players, and thus, other insights might be memorable to them. For example, while a business stakeholder might recognize or be surprised by the outer viscous cycles, a developer might be more surprised about the business causes and chains of causes.

We qualitatively analyzed the answers to the open questions and the recordings' transcriptions. To track changes in attitudes and behaviors, we counted responses that indicated a change or a plan to change. We also noted how many responses stated no intention to alter their current attitudes or behaviors. Furthermore, we identified repeated explanations for their changes in attitudes and behavior and also counted their frequency. For the feedback on the game, we counted the number of players who expressed their satisfaction with the game and identified further potential for improvement. As these three questions were optional, some players skipped them, leading to fewer responses than the total number of players. Finally, we read through the recordings' transcriptions and marked interesting discussions and remarks that showed participants could relate the game to their real-life experiences.

To answer RQ 1 on the emulation of real-life experiences, we utilized the *"I had the AHA moment, but the realization is NOT NEW to me."* answers and combined those with the answers to the open questions and the transcriptions' analysis. To answer RQ 2 on the new insights on TD and TD management, we utilized the *"I had the AHA moment, and the realization is NEW for me."* answers and also combined those with the answers to the open questions. For RQ 3 on the changes in attitude or planned behavior, we utilized a qualitative analysis of the answers to the open questions.

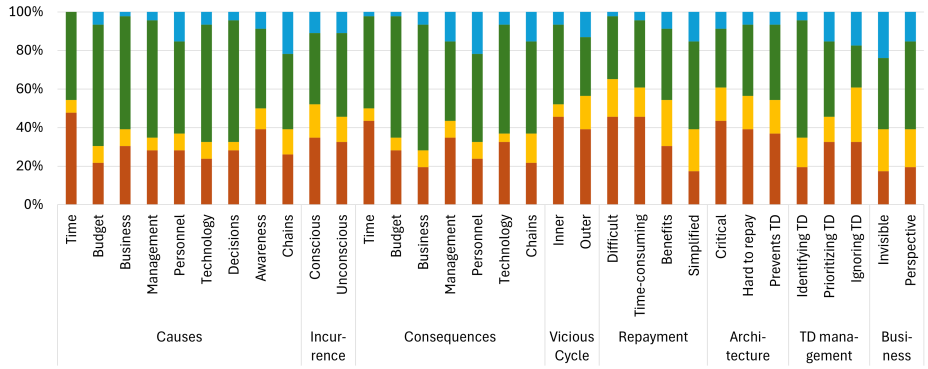
VI. RESULTS

In this section, we present the results of our evaluation by answering each research question.

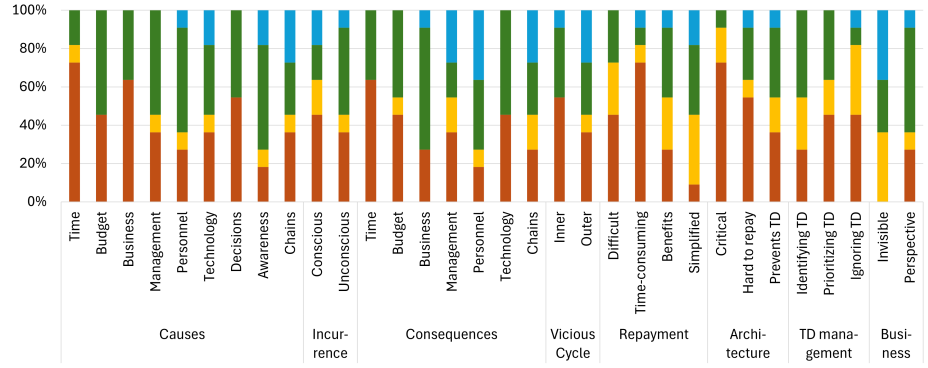
A. RQ1: To what extent does the game emulate the real-life experiences that technical stakeholders encounter?

Figure 3(e) shows that for all of the *aha*-moments, only a few senior technical stakeholders did not recognize them from their previous experience (yellow + blue bars), which demonstrates that the *aha*-moments represent real-life situations. Furthermore, Figure 3(a) shows that for each aha-moment, 30% to 60% of all players had the respective *aha*-moment (orange + yellow bars). *Aha*-moments for vicious cycles, repayment, architecture, and TD management were experienced most often (yellow + orange bars), while causes and consequences were recognized less frequently. Particularly, the players rarely experienced the *aha*-moments of

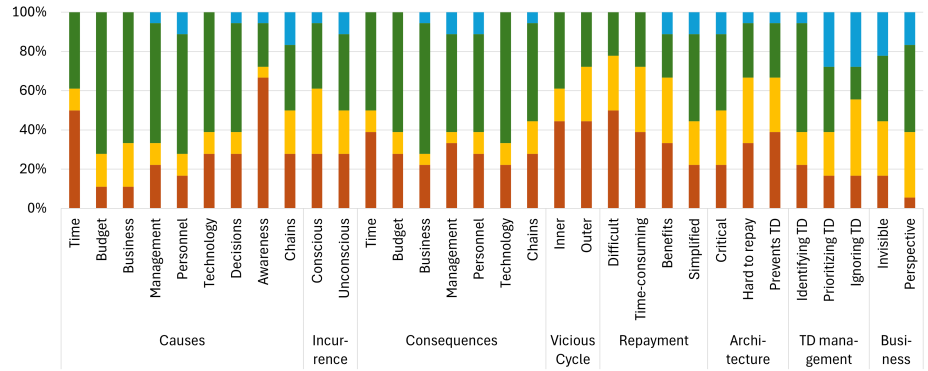
Authorized licensed use limited to: Bibliothekssystem Universitaet Hamburg. Downloaded on August 26,2025 at 10:01:46 UTC from IEEE Xplore. Restrictions apply.



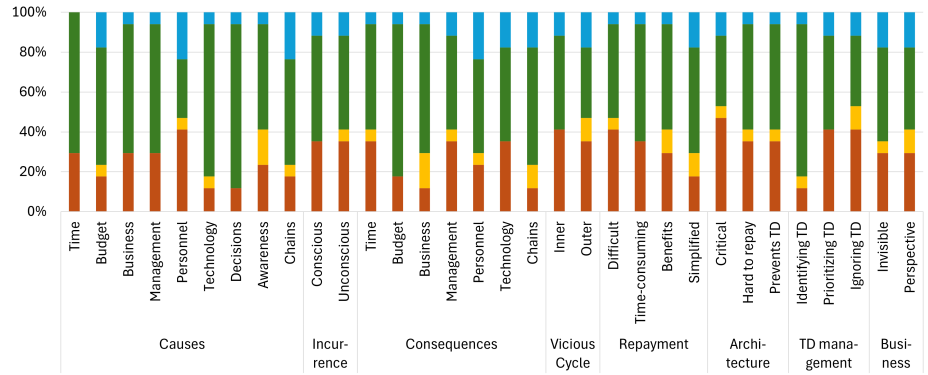
(a) over all players



(b) business stakeholders



(c) junior technical stakeholders



(d) senior technical stakeholders



(e) senior technical stakeholders

Fig. 3: Percentage of players that had the respectiv ments for all players and per stakeholder category

Authorized licensed use limited to: Bibliothekssystem Universitaet Hamburg. Downloaded on August 26,2025 at 10:01:46 UTC from IEEE Xplore. Restrictions apply.

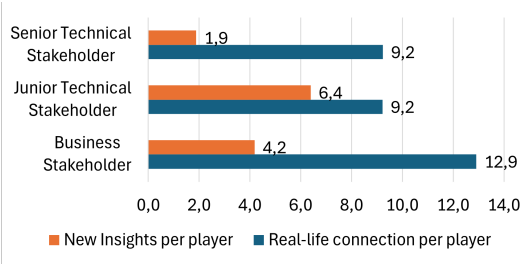


Fig. 4: Average new insights (I had the aha moment, and it is new to me) and average real-life connections (I had the aha moment, and it is not new to me) per stakeholder category.

budget causes and business consequences. These low values might be addressed by adding more action and event cards to the game. As all *aha*-moments were perceived by at least some players, we assume that the game offered the opportunity to experience all targeted *aha*-moments.

Figure 4 also shows that from each stakeholder category, each player could relate at least nine *aha*-moments to their real life, further proving the realistic design of the game.

Further, we identified that connections to real life were made during the games, as the following examples show: *"P1: We already have so much [TD]in the system. / P2: That's always one [TD per feature]. (Laughter) / P1: It's really like reality. (Laughter)"* and *"P1: [Card is read out] Developer retires. A good developer retires. (Laughs) He wrote good code but, unfortunately, didn't document it very well. Oh, that sounds familiar. (Laughs)"*

Finally, the survey's feedback on the game included statements supporting the game's realism: *"I liked how it [the game] simulates the difficulties and the setbacks we often find during software development." "The ?-cards were sometimes quite hard and set you back in the game, but that's how it is in reality with technical debt, so they are justified."* Some more critical feedback involved the ease of repaying architectural TD: *"But I do not like that repaying in this game TD of architecture TD has 50% of success [b]ecause I think repaying TD in architecture Level is much worse to do."* This feedback can be used to adapt the game, e.g., by blocking more digits to repay architectural TD.

Regarding RQ 1, the *aha*-moments, the transcriptions, and the feedback led us to consider the game as a realistic emulation of the real world.

B. RQ2: Do the players encounter new insights and learn about additional facets of TD management?

Figure 3(a) shows the *aha*-moments that were new to the players in yellow. All *aha*-moments were new to at least one player showing the relevance of each *aha*-moment. Notably, there is a substantial difference between stakeholders, which is also revealed in Figure 4. While senior technical stakeholders encounter, on average, less than two new insights, i.e., new *aha*-moments, business stakeholders more than four, and junior technical stakeholders encounter more than six. This illustrates that the game might assist in harmonizing the level of knowledge of all stakeholders.

Business stakeholders often noted new insights regarding measures that simplify refactorings, ignoring TD as a possibility and the fact that some TDs are invisible to business stakeholders and can only be noticed by their symptoms (yellow bars in Figure 3(b)). Further, the difficulties in repaying TD and identifying it were new to multiple business stakeholders. This indicates that crucial aspects of their work in determining which TD items should be prioritized as high or low in the backlog may have been previously overlooked.

For junior technical stakeholders, the new insights are broadly distributed across all *aha*-moments (yellow bars in Figure 3(c)). The topics of TD incurrence, vicious cycles, TD repayment, the relevance of TD in architecture, TD management, and the business perspective on TD are often perceived as new insights by them. At the same time, they less frequently recognized *aha*-moments on causes and consequences. This indicates that the broad perspective on TD embodied in the game is a valuable way of making TD knowledge tangible for junior technical stakeholders. The effect on junior stakeholders is also mirrored in their answers to the open questions: *"If you have no experience with the topic of TD, the game could very well help you learn the basics in a playful way"*

For RQ 2, we summarize that while all players gain new insights, business stakeholders and junior stakeholders acquire the most knowledge from the game. In the end, this harmonizes the knowledge about TD concepts between stakeholders, facilitating knowledgeable discussions on TD management after the game.

In the end, this harmonizes the knowledge about TD concepts between stakeholders, enabling knowledgeable discussions on TD management after the game.

C. RQ3: How does the game change players' attitudes and possibly behaviors regarding TD?

Out of all 46 players, 26 of 44 responding players replied that they changed their attitude towards TD, and 27 of 34 responding players plan to change their behavior.

Change in attitude comprises (number of mentions in brackets) more awareness of causes (5) and consequences (11), repayment of TD (7), dependencies between TD items (5), the importance of TD management (2), easier TD identification (2) and the significance of architectural TD (1). For example, one player said: *"The game clearly shows what consequences TD can cause in a module and how important it can be to fix TD. That's why the game sensitized me."* Another mentioned: *"It made me realize that the impact of TD on subsequent tickets is not negligible and should be dealt with strategically."* Additionally, five players recalled the benefits of incurring TD for swift product delivery. One of them mentioned that TD is *"[n]o longer a concept of fear because it was conveyed that you can deal with it or work with it."*

Among the 18 players who did not change their attitude, 17 reported that they were already familiar with the TD

Authorized licensed use limited to: Bibliothekssystem Universitaet Hamburg. Downloaded on August 26,2025 at 10:01:46 UTC from IEEE Xplore. Restrictions apply.

concepts and how they work, therefore, they felt no need to alter their attitude. One technical stakeholder stated: *"My attitude hasn't changed; technical debt is part of the core business of my job."* A business stakeholder noted: *". . . I had to fight for many years against management that prioritized time for short-term features above all TD."*

Planned behavior changes (number of mentions in brackets) comprise making decisions on TD incurrence more rational by evaluating options and their consequences (12), employing a more structured approach to TD (10), starting tracking TD (7), repaying TD more regularly (3), and being more aware of TD overall (4). For example, one player said, regarding TD incurrence, that they *"plan to look for TDs in a more critical way and see how it can impact the development process in the future."* Regarding tracking, one player noted that *"[a]lthough some TD are invisible, there are enough visible TD . . . or those that become visible, which should also be clearly named."* Concerning a structured TD management approach, one player noted *". . . it seems even more important to me to manage technical debt thoughtfully."*

Two players who did not plan to change their behavior mentioned other people or institutions, like the management or the company, that already manage TD, e.g., *". . . in my context, there are already processes for dealing with technical debts and . . . they are under the responsibility of other people."* Five players noted they did not plan to change their behavior but stated they were already aware of TD and they *"always try to avoid technical debt and only give in when pressure comes from above."*

For RQ 3, we conclude that most players are open to changing their attitude and behavior regarding TD and TD management after playing the game. Furthermore, almost all players unwilling to change their attitude or behavior justified this by claiming they already knew about TD and tried to manage it. In both cases, the players were open to productive and rational discussions, which was one of the game's goals.

VII. DISCUSSION

In this section, we discuss the results of Section VI in terms of the game's effect on different stakeholders. Additionally, we summarize the feedback on the game itself.

A. Effect on Senior Technical Stakeholders

Senior technical stakeholders got the least new insights from the game. However, we were still able to fill some knowledge gaps, e.g., for three seniors, the fact that TD might have consequences for the business was new, which is important new knowledge. Senior technical stakeholders also exhibited the least change in attitude towards TD since they already had vast TD knowledge. Nonetheless, many of them were open to behavior changes after playing the game, and ideas on what to change were reinforced. Besides filling knowledge gaps, the most significant benefit of having senior technical stakeholders participate in the game is sharing their knowledge with junior and business stakeholders. This exchange might be a valuable starting point for TD management discussion.

B. Effect on Junior Technical Stakeholders

Of all stakeholders, junior technical stakeholders profited the most from new insights. The fact that sometimes ignoring TD is a valid option was new to twelve of them, and seven mentioned that they had this *aha*-moment. Being still new to IT, developers might be looking for the perfect solution and lack the experience to sometimes accept compromises. The game taught them that compromises must sometimes be made to reach business goals. Additionally, academic literature shows that TD incurred inadvertently by inexperienced developers is one frequently mentioned cause of TD [7], [34]. The game might support junior developers in being more careful during implementations and showing more understanding in discussions with experienced developers.

C. Effect on Business Stakeholders

Academic literature reveals that communication with business stakeholders can be challenging [7], [35], [36]. By playing the game, business stakeholders had substantially more new insights than senior technical stakeholders and thus learned new characteristics of TD, which might be beneficial in future discussions about TD prioritization. In particular, the reality that some TD might not be visible to them was new to eight of eleven business stakeholders and mentioned by four of them as a new learning. Providing this crucial knowledge might be useful for business stakeholders to put more trust in the technical stakeholders' evaluation of TD. This may be a particularly useful effect of the game since missing business stakeholders' trust makes communication about TD especially difficult [7].

D. Feedback on the TechDebt Game

The overall feedback on the game was encouragingly positive. 15 players explicitly stated that they had fun playing the game, e.g., *"The game is a lot of fun! I would also play privately!"* or *"With just a few minutes playing the game, I felt immersed and could think about some strategies."* Even while playing the game, some players wanted to immediately "buy" the game: *"P2: . . . I think it's a really good idea as a board game. . . . I would buy it. (Laughs) / P1: Ah, okay. New Ravensburger [German game manufacturer] Technical Debt game."* Accordingly, after one of the games, we even got an offer to support professionalizing the game. Moreover, one company was considering using the game company-wide to change the company's culture regarding TD.

Some players made suggestions for improvement, e.g., regarding TD causes, one player mentioned that *"[t]he game was able to convey the "time" factor for the creation of TD very well, but the "resources" factor (money, personnel) less well. The event cards were good for this, to get examples."* Regarding the action and event cards, two players mentioned that *"[m]ore different '!' [action] or '?' [events] effects would make it [the game] more interesting."* Another player mentioned: *"The event cards were a nice way to impart knowledge*

Authorized licensed use limited to: Bibliothekssystem Universitaet Hamburg. Downloaded on August 26,2025 at 10:01:46 UTC from IEEE Xplore. Restrictions apply.

about TD, and I believe many of my 'I did not have the aha moment' would have appeared through other cards in the game." One player missed interaction: *"The only thing I missed was more interaction with the other teams in the game."* Finally, as mentioned in the previous sections, some players also complained about the influence of luck and the ease of repaying architectural TD.

VIII. THREATS TO VALIDITY

We present threats to validity based on the guidelines provided by Wohlin et al. [37].

a) ***Construct Validity****:* First, we built the game along the *aha*-moments as learning objectives. We derived these moments from scientific literature, e.g., the works of Li et al. [4], Wiese et al. [7], Martini et al. [38], Kruchten et al. [2], and Ramac et al. [39]. While we might have missed important aspects of TD and TD management, the game is designed in a way that might embed further *aha*-moment by creating additional action or event cards. Second, in four game sessions, we deviated from the original study design and let a researcher play against or alongside a participant, posing a threat to construct validity. However, this may improve external validity as these game sessions replicate real-life situations.

b) ***Internal Validity****:* First, to analyze the game's effectiveness, we analyzed the subjective perspective of the participants and did not provide objective measures. We assume that this is in line with the overall goal of the game, which is to serve as a starting point for discussing TD and to foster an open yet educated mindset on this topic, which is a highly subjective perception. Second, the qualitative analysis of transcripts and open-ended questions may introduce biases. To mitigate this, two researchers collaboratively reviewed all results. Additionally, the answers to the open questions and their analysis are part of the additional material [16] to be reviewed by all readers of this paper. We also combined qualitative findings with quantitative survey data to address the research questions. Finally, one game goal is to foster the discussion about TD, which we did not analyze specifically. We derived the openness for discussion by analyzing participants' willingness to change their behavior and added some citations proving that discussions indeed happened. However, this is subjective or anecdotal knowledge and may be evaluated further by analyzing team discussion after playing the game.

c) ***External Validity****:* First, we played the game with participants from more than ten different industry domains and included various stakeholders of many experience levels. However, like most industry research, our sample is accompanied by selection bias, as mainly players interested in the TD topic were willing to participate. Second, we would have preferred to include more business participants in the evaluation. Yet, in real life, one project manager or product owner manages multiple developers, which is a ratio that is also exhibited in our game sessions. Third, none of the games was played without a knowledgeable game master. Thus, we cannot ensure that the game instructions document is sufficiently self-explanatory. However, the players used the manual to examine rules, and we optimized the manual throughout the various iterations, resulting in an additional short manual and an overview on the game board.

d) ***Conclusion Validity****:* Our conclusions are only short-term related, i.e., we only surveyed the player's situation right after they played the game. This is aligned with our goal of starting discussions on TD management. We did not systematically analyze whether long-term changes resulted from the change.

IX. CONCLUSION

In this paper, we present a board game for conveying TD concepts, making experiences with TD tangible, and fostering discussions on managing TD. The game was designed based on a thorough theoretical foundation presented in Section III. We evaluated the game by playing it with 46 practitioners in 13 game sessions using a questionnaire after the game to gather feedback on the conveyed *aha*-moments. For qualitative feedback, we added open questions to the questionnaire and analyzed transcripts from the games' recordings.

By answering our RQs, we established that the game is fun to play, realistic, and can be used to harmonize the level of knowledge of all stakeholders. Many players changed their attitude toward TD and plan to change their behavior.

Practitioners, can use the game to experiment with various strategies in a safe environment ("graceful failure"). The game provides an unbiased and rational basis for further TD discussions. For long-term effectiveness, however, the game must be followed by a discussion on embedding TD management in the current software development life cycle.

For researchers, we presented that game-based learning is not only a method to impart knowledge but also a means to foster discussions and spark behavior changes. The game might also be used as a starting point when training practitioners on methods to manage TD.

Future work might include researching how educational programs and training might help embed TD management in an existing software development life cycle. The game might then be used as a first step in those trainings. Moreover, the company of one game session's participants considers using the game company-wide to foster a change of mindset throughout the company, which could be evaluated. In our study, players only played the game once. A further research topic might be to evaluate how experimenting in this emulated environment might be profitable.

Acknowledgment and Funding: We thank the game players for their participation, their trust, and the insights they provided. In memory of Prof. Dr.-Ing. André van Hoorn and Prof. Dr.-Ing. Matthias Riebisch. The project on which this report is based was sponsored by the Federal Ministry of Education and Research of Germany under the funding code 01IS24031. Part of this work was supported by Funetec-PB – Ct.: Phoebus 01/24. Responsibility for the content of this publication lies with the authors.

Authorized licensed use limited to: Bibliothekssystem Universitaet Hamburg. Downloaded on August 26,2025 at 10:01:46 UTC from IEEE Xplore. Restrictions apply.

References

[1] P. Avgeriou, P. Kruchten, I. Ozkaya, and C. Seaman, "Managing Technical Debt in Software Engineering," *Dagstuhl Reports*, vol. 6, no. 4, pp. 110–138, 2016.

[2] P. Kruchten, R. Nord, and I. Ozkaya, *Managing Technical Debt: Reducing Friction in Software Development.* Software Engineering Institute, Carnegie Mellon University, 2019.

[3] N. Ernst, R. Kazman, and J. Delange, *Technical Debt in Practice.* London, England: The MIT Press Cambridge, Massachusetts, 2021.

[4] Z. Li, P. Avgeriou, and P. Liang, "A systematic mapping study on technical debt and its management," *Journal of Systems and Software*, vol. 101, pp. 193–220, 2015.

[5] N. Rios, M. G. de Mendonça Neto, and R. O. Spínola, "A tertiary study on technical debt: Types, management strategies, research trends, and base information for practitioners," *Information and Software Technology*, vol. 102, pp. 117–145, 2018.

[6] H. J. Junior and G. H. Travassos, "Consolidating a common perspective on Technical Debt and its Management through a Tertiary Study," *Information and Software Technology*, vol. 149, no. October 2020, p. 106964, 2022.

[7] M. Wiese and K. Borowa, "IT managers' perspective on Technical Debt Management," *Journal of Systems and Software*, vol. 202, p. 111700, Apr. 2023.

[8] "InsighTD Project – InsighTD Project." [Online]. Available: http://www.td-survey.com/

[9] N. Rios, R. O. Spínola, M. Mendonça, and C. Seaman, "The practitioners' point of view on the concept of technical debt and its causes and consequences: a design for a global family of industrial surveys and its first results from Brazil," *Empirical Software Engineering*, vol. 25, no. 5, pp. 3216–3287, 2020.

[10] R. Ramač, V. Mandić, N. Taušan, N. Rios, S. Freire, B. Pérez, C. Castellanos, D. Correal, A. Pacheco, G. Lopez, C. Izurieta, C. Seaman, and R. Spinola, "Prevalence, Common Causes and Effects of Technical Debt: Results from a Family of Surveys with the IT Industry," *Journal of Systems and Software*, 2021.

[11] W. Cunningham, "The WyCash portfolio management system," in *Proceedings of the Conference on Object-Oriented Programming Systems, Languages, and Applications, OOPSLA*, vol. Part F1296, no. 2, 1992, pp. 29–30.

[12] R. Verdecchia, P. Kruchten, P. Lago, and I. Malavolta, "Building and evaluating a theory of architectural technical debt in software-intensive systems," *Journal of Systems and Software*, vol. 176, p. 110925, 2021.

[13] T. Besker, A. Martini, and J. Bosch, "The use of incentives to promote technical debt management," *Information and Software Technology*, vol. 142, p. 106740, 2022.

[14] P. Avgeriou, I. Ozkaya, A. Chatzigeorgiou, M. Ciolkowski, N. A. Ernst, R. J. Koontz, E. Poort, and F. Shull, "Technical Debt Management: The Road Ahead for Successful Software Delivery," in *2023 IEEE/ACM International Conference on Software Engineering: Future of Software Engineering (ICSE-FoSE)*, May 2023, pp. 15–30.

[15] M. Wiese, P. Rachow, M. Riebisch, and J. Schwarze, "Preventing technical debt with the TAP framework for Technical Debt Aware Management," *Information and Software Technology*, p. 106926, 2022.

[16] M. Wiese, A. Heinrichs, N. Rusieshvili, and K. B. Rebouças de Almeida, Rodrigo, "Additional material for teh techdebt game - enabling discussions a," 2024. [Online]. Available: https://doi.org/10.5281/zenodo.14205656

[17] A. Baker, E. Navarro, and A. van der Hoek, "Problems and Programmers: an educational software engineering card game," in *25th International Conference on Software Engineering, 2003. Proceedings.*, May 2003, pp. 614–619, iSSN: 0270-5257.

[18] J. M. Fernandes and S. M. Sousa, "PlayScrum - A Card Game to Learn the Scrum Agile Method," in *2010 Second International Conference on Games and Virtual Worlds for Serious Applications*, Mar. 2010, pp. 52–59.

[19] V. T. Heikkilä, M. Paasivaara, and C. Lassenius, "Teaching university students Kanban with a collaborative board game," in *Proceedings of the 38th International Conference on Software Engineering Companion*, ser. ICSE '16. New York, NY, USA: Association for Computing Machinery, May 2016, pp. 471–480.

[20] R. C. De Boer, P. Lago, R. Verdecchia, and P. Kruchten, "DecidArch V2: An Improved Game to Teach Architecture Design Decision Making," in *Proceedings - 2019 IEEE International Conference on Software Architecture - Companion, ICSA-C 2019*, 2019, pp. 153–157, iSBN: 9781728118765 Publisher: IEEE.

[21] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice: Software Architect Practice*, 4th ed. Addison-Wesley, 2021.

[22] H. Cervantes, S. Haziyev, O. Hrytsay, and R. Kazman, "Smart decisions: an architectural design game," in *Proceedings of the 38th International Conference on Software Engineering Companion*, ser. ICSE '16. New York, NY, USA: Association for Computing Machinery, May 2016, pp. 327–335. [Online]. Available: https://dl.acm.org/doi/10.1145/2889160.2889184

[23] L. Ganesh, "Board game as a tool to teach software engineering concept - Technical debt," in *Proceedings - IEEE 6th International Conference on Technology for Education, T4E 2014*, 2014, pp. 44–47, iSBN: 9781479964895 Publisher: IEEE.

[24] "Technical Debt Game." [Online]. Available: https://www.tha.de/en/Computer-Science/THA-ias/Technical-Debt-Game.html

[25] "Technical Debt Game – for non-technical people." [Online]. Available: https://ri-level.de/technical-debt-game/

[26] B. Suits, "What is a Game?" *Philosophy of Science*, vol. 34, no. 2, pp. 148–156, Jun. 1967.

[27] M. Pivec, O. Dziabenko, and I. Schinnerl, "Aspects of GameBased Lerning," in *Proceedings of 3rd International Conference on Knowledge Management*, Graz, Austria, 2003.

[28] J. L. Plass, B. D. Homer, and C. K. Kinzer, "Foundations of Game-Based Learning," *Educational Psychologist*, Oct. 2015, publisher: Routledge. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/00461520.2015.1122533

[29] J. C. Huizenga, G. T. M. ten Dam, J. M. Voogt, and W. F. Admiraal, "Teacher perceptions of the value of game-based learning in secondary education," *Computers & Education*, vol. 110, pp. 105–115, Jul. 2017.

[30] Y.-R. Shi and J.-L. Shih, "Game Factors and Game-Based Learning Design Model," *International Journal of Computer Games Technology*, vol. 2015, no. 1, p. 549684, Jan. 2015, publisher: John Wiley & Sons, Ltd.

[31] R. Garris, R. Ahlers, and J. E. Driskell, "Games, Motivation, and Learning: A Research and Practice Model," *Simulation & Gaming*, Dec. 2002, publisher: Sage PublicationsSage CA: Thousand Oaks, CA.

[32] "Tabletop TechDebt Game on GitHub." [Online]. Available: https://github.com/TechDebtGame/TechDebtGame

[33] "Digital TechDebt Game on Tabletopia." [Online]. Available: https://tabletopia.com/workshop/games/techdebts/1-7players/test

[34] S. S. Freire, N. Rios, B. Pérez, C. Castellanos, D. Correal, R. Ramač, V. Mandić, N. Taušan, G. López, A. Pacheco, D. Falessi, M. Mendonça, C. Izurieta, C. Seaman, and R. Spínola, "How Experience Impacts Practitioners' Perception of Causes and Effects of Technical Debt," in *2021 IEEE/ACM 13th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*, 2021, pp. 21–30.

[35] N. A. Ernst, S. Bellomo, I. Ozkaya, R. L. Nord, and I. Gorton, "Measure it? Manage it? Ignore it? Software practitioners and technical debt," *10th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, ESEC/FSE 2015 - Proceedings*, pp. 50–60, 2015.

[36] T. Besker, J. Bosch, A. Martini, and J. Bosch, "Technical debt cripples software developer productivity: A longitudinal study on developers' daily software development work," in *Proceedings - International Conference on Software Engineering.* ACM, 2018, pp. 105–114.

[37] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in software engineering*, 2012, vol. 9783642290442.

[38] A. Martini and J. Bosch, "The Danger of Architectural Technical Debt: Contagious Debt and Vicious Circles," in *Proceedings - 12th Working IEEE/IFIP Conference on Software Architecture, WICSA 2015*, 2015, pp. 1–10.

[39] R. Ramač, V. Mandić, N. Taušan, N. Rios, S. Freire, B. Pérez, C. Castellanos, D. Correal, A. Pacheco, G. Lopez *et al.*, "Prevalence, common causes and effects of technical debt: Results from a family of surveys with the it industry," *Journal of Systems and Software*, vol. 184, p. 111114, 2022.

Authorized licensed use limited to: Bibliothekssystem Universitaet Hamburg. Downloaded on August 26,2025 at 10:01:46 UTC from IEEE Xplore. Restrictions apply.

Chapter 15

Establishing Technical Debt Management – A Five-Step Workshop Approach and an Action Research Study

	6 *(Establishing TDM)*
Title	Establishing Technical Debt Management – A Five-Step Workshop Approach and an Action Research Study
Authors	Wiese M., Serwa K., Besier A., Marion-Jetten A., Bittner E.
Year	2026
Journal	Journal of Systems and Software, Volume 231 Special Issue on Managing Technical Debt in Software-intensive Products and Services
Reference	[Wie+26]

TABLE 15.1: Publication No. 6 *(Establishing TDM)*

The Journal of Systems and Software 231 (2026) 112606

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

Establishing technical debt management — A five-step workshop approach and an action research study☆,☆☆

Marion Wiese [a,*], Kamila Serwa [a], Anastasia Besier [a], Ariane S. Marion-Jetten [b], Eva Bittner [a]

[a] University of Hamburg, Department of Informatics, Hamburg, Germany
[b] University of Hamburg, Department of Health Science, Hamburg, Germany

ARTICLE INFO

Dataset link: https://zenodo.org/records/14916678

Keywords:
Technical debt
Technical debt management
Action research
Technical debt awareness
Technical debt activities

ABSTRACT

Context: Technical debt (TD) items are constructs in a software system providing short-term benefits but hindering future changes. TD management (TDM) is frequently researched but rarely adopted in practice.
Goal: This study aimed to establish a TDM process in an IT company based on a predefined workshop concept. We analyzed which research approaches practitioners adopted for each TD activity and the TDM's long-term effect on TD awareness.
Method: We used action research (fiveaction cycles in 16months) with an IT team that creates IT solutions for signal processing. To examine TD awareness, we (1) analyzed questionnaires completed during each workshop, (2) observed team meetings, (3) adopted a method from psychology for measuring awareness in decision-making situations called TD-SAGAT, and (4) evaluated the backlog data.
Results: Practitioners preferred TD repayment and prioritization based on the system's evolution and cost calculations, i.e., repayment of so-called low-hanging fruits. Reminders in the backlog items, such as checkboxes or text templates, led to a sustainable rise in TD awareness.
Conclusions: We showed that a workshop-based approach is feasible and leads to sustainable process changes. New ideas for TDM applicable to other IT teams emerged, e.g., using a *re-submission date*, using a *Talked about TD* checkbox, and using visualizations for TD prioritization.

1. Introduction

In 1992, Cunningham (1992) introduced the metaphor of technical debt (TD) to explain that code rewrites might be needed if the software is implemented under time pressure. Currently, a single instance of TD, called TD item, can be described as a construct that is beneficial in the short term but hinders future development (Avgeriou et al., 2016; Kruchten et al., 2019). Multiple causes and consequences of TD have been identified and categorized (Rios et al., 2020; Ramač et al., 2022; Wiese and Borowa, 2023). Also, various TD types, like code, architectural, test, or infrastructure debt, have been uncovered (Li et al., 2015; Ernst et al., 2021). A conceptual model has been developed and gradually improved (Avgeriou et al., 2016; Junior and Travassos, 2022; Wiese and Borowa, 2023).

Research on TD management (TDM) is manifold (Wiese et al., 2022; Guo et al., 2016a; Yli-Huumo et al., 2016a; Ramasubbu and Kemerer, 2019) and typically organized around so-called TD activities, such as TD identification, TD prioritization, or TD monitoring (Li et al., 2015). However, two tertiary studies on TD management (TDM) found that holistic approaches for TDM, i.e., approaches including various TD activities and various TD types, are still missing (Rios et al., 2018; Junior and Travassos, 2022). In their vision paper on TDM, Avgeriou et al. specified that TD research currently focuses on single TD types, mostly code debt, but should include various TD types (Avgeriou et al., 2023). Additionally, research on systematically establishing a TDM process is missing. Our previous study showed that IT managers considered TDM to be essential and their duty, but they were usually unable to establish a TDM process. Attempts to set up a TDM process were often half-hearted, ended prematurely, and led to "cluttered backlogs" (Wiese and Borowa, 2023). Our goal was to analyze existing research efforts and, together with new research findings, tailor it to the needs of practitioners. Moreover, we explored issues hindering a TDM establishment

☆ This article is part of a Special issue entitled: 'TechDebt' published in The Journal of Systems & Software.
☆☆ Editor: Prof. Raffaela Mirandola.
* Corresponding author.
E-mail addresses: marion.wiese@uni-hamburg.de (M. Wiese), kamila.serwa@uni-hamburg.de (K. Serwa), maria.besier@uni-hamburg.de (A. Besier), ariane.marion-jetten@uni-hamburg.de (A.S. Marion-Jetten), eva.bittner@uni-hamburg.de (E. Bittner).

https://doi.org/10.1016/j.jss.2025.112606
Received 30 August 2024; Received in revised form 8 July 2025; Accepted 21 August 2025
Available online 13 September 2025
0164-1212/© 2025 The Authors. Published by Elsevier Inc. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

in practice. These issues and intentions led to our first research question (RQ):

RQ 1: Which TDM approaches are useful for the practitioners of our case company?

RQ 1.1.: Which TDM approaches do practitioners adopt?
RQ 1.2.: Which new approaches do practitioners develop?

A frequently mentioned cause for TD is time pressure (Ernst et al., 2015; Martini and Bosch, 2015; Soliman et al., 2021; Wiese et al., 2022). Thus, establishing and maintaining a TDM process should not take much additional time. The amount of time that has to be invested on top of the time for actually repaying TD is also essential information. For example, it is relevant when further IT teams negotiate a TDM process establishment with their IT management. This led us to our next RQ:

RQ 2: How much effort are the practitioners of our case company willing to invest in a TDM process?

Avgeriou et al. also identified that there is a need to study TD's "social- technical and multi-disciplinary aspects", which includes expertise from various domains such as "human aspects of software engineering" (Avgeriou et al., 2023). One of those social-technical aspects is a missing awareness of TD, a frequently mentioned cause for incurring TD (Martini and Bosch, 2015; Tonin et al., 2017; Rios et al., 2020; Ramač et al., 2021; Wiese et al., 2022; Wiese and Borowa, 2023). In some studies, TD awareness is mainly mentioned as a necessity for TD management (Holvitie et al., 2018). Further, multiple research studies report that establishing a TDM process led to a raised TD awareness but fall short in proving this claim, e.g., Russo et al. (2022). Other studies present a simplified perspective on what TD awareness entails, e.g., that TD awareness is simply the knowledge of the own system's TD items, e.g., Besker et al. (2017), Ocker et al. (2019). Some studies assess TD awareness, but they usually employ surveys asking the participant whether they are aware of TD in a one-time format lacking longitudinal observations, e.g. Dong et al. (2019), Wiese et al. (2022). However, to avoid and regularly repay TD, participants must be aware of TD in their daily routine, e.g., when making decisions on designs or architectures. Our goal was to analyze whether the establishment of a TDM process improved this kind of TD awareness, leading to RQ3: **RQ 3: Is the team members' awareness of TD in decision-making situations improved sustainably by the TDM process's establishment?**

To analyze these three RQs, the action research method (Staron, 2020) provides a suitable research approach, combining an academic approach with practical insights. Five action research studies established TDM processes in companies (Oliveira et al., 2015; Yli-Huumo et al., 2016b; Borup et al., 2021; Detofeno et al., 2021, 2022). However, they each focused on a few TD activities and did not provide holistic approaches to establishing a TDM process that not only handled TD but also prevented it. Moreover, these studies rarely evaluated the long-term effects on the team's TD awareness during their decision-making regarding TD.

Our study aimed to establish a holistic TDM process and analyze the results in a long-term action research study. For this purpose, we created a workshop concept comprising the presentation of various TD research results to practitioners, structured by TD activities. Then, we let the practitioners decide which results and suggestions they wanted to adopt or adjust to their needs. This self-determined process aligns best with the agile mindset of a self-organized team rather than providing a pre-defined TDM process. We employed the agile change management method (Führing, 2021) from the economics domain because we identified that establishing a TDM process emulates an organizational change process. From this method, we adopted ideas such as conducting workshops to change a corporate mindset, employing agile change management sprints to support self-management, or diagnosing the current situation to identify the needed changes. In addition, this approach fits well with the action research method, as both are iterative approaches, and the "evaluation", "learning", "diagnosing", and "action planning" phases of an action research cycle can be represented within workshops of an agile change management. In this setting, the practitioners were also able to specify their own ideas for further approaches. They could also identify the effort each approach takes and whether they were willing to invest this effort to gain the resulting outcome. Such an agile workshop approach is highly flexible and takes the specifics of each team and company into account. Accordingly, we assume that it is less context-specific than other approaches and can be adopted in multiple settings.

During the workshop discussions, we evaluated which approaches developed or identified by academia were feasible in this practical environment and which research results were not applicable. We used questionnaires, interviews, and focus group discussions on the TDM topics during five action cycles. Moreover, we assessed the team's backlog and observed discussions for 16 months in 35 meetings comprising 25 meeting hours. Finally, we used the expertise of a psychology researcher to adopt a measurement method for *Situation Awareness*, i.e., to determine TD awareness in decision-making situations. This collaboration resulted in a new measurement method we call TD-SAGAT (Technical Debt's Situation Awareness Global Assessment Technique). We concluded the study when the team members were content with the resulting process and determined that the process did not need any further significant changes.

By answering the research questions, we contributed insights into the long-term effects the establishment of a TDM process had in a practical context and which research approaches practitioners have chosen. Additionally, we identified new approaches supplementing a TDM process, i.e.,

1. using a *re-submission date* for backlog items to recall them regularly
2. adding a *Talked about TD* checkbox to all backlog items to support TD prevention
3. using an *educated guess* based on TD item attributes
4. calculating a *return on investment* to optimize the *educated guess*
5. using visualizations to support the prioritization of TD items, e.g., identifying *low-hanging fruits*

1.1. Outline

The following Section 2 introduces the background information that forms the basis for understanding this study and the workshop presentations. We compare our study with related work in Section 3 and demonstrate our research methodology in Section 4. To disclose the workshop design for replication, we explain the workshops in Section 5. In Section 6, we outline the progress of each action cycle in terms of actions taken and learnings. We present the research results in Section 7, which comprises the action research results to answer RQ 1 and RQ2 in Sections 7.1 and 7.2 and the evaluation of the team's TD awareness to answer RQ3 in Section 7.3. Afterward, we discuss the results in Section 8, including a discussion of the TDM establishment with actionable recommendations for practitioners in Section 8.1, a summary of the TD awareness effects in Section 8.2 and a critical review of the study design in Section 8.3. We address the threats to the validity of our study in Section 9, and conclude our paper in Section 10. The additional material is available at Zenodo (Wiese and Serwa, 2025). It comprises all workshop slides and material, photographs of flip charts, evaluations (incl. questionnaire questions), and an introduction to the TD-SAGAT method.

2. Background

This section introduces the TDM topics we similarly introduced to the practitioners during the action cycle workshops.

2.1. Technical debt metaphor and definition

Cunningham introduced TD as a metaphor for financial debt (Cunningham, 1992). He described that prioritizing time over quality in software development can lead to the need for code rewrites. The suboptimal implementation is interpreted as *debt*, the resulting problems as *interest* rates, the refactoring as *repayment*, and the refactoring cost as *principal*. Later works described that the interest of TD is subject to a probability of occurrence, i.e., there is a certain *interest probability* (Schmid, 2013a). For example, when poorly written code needs extra time to be understood, the overall interest also depends on how often this code fragment must be read or how probable it is that it must be read.

According to the widely accepted "Dagstuhl definition", TD items are constructs that provide short-term benefits but hinder the long-term development and maintenance of the software and mainly refer to internal software qualities (Avgeriou et al., 2016). A single instance of TD, i.e., one construct, is referred to as a *TD item* (Kruchten et al., 2019). Kruchten et al. introduced the "colors in a backlog" determining the category of a backlog item, i.e., ticket (Kruchten et al., 2012a). They differentiate between whether objects are visible or invisible to customers and whether they increase a system's value or reduce its negative effects. Functional and architectural items increase the system's value, but the results of architectural items are invisible to the customer. Bugs and TD items shall reduce negative effects; however, TD is invisible to the customer. Since it is still difficult to determine whether a backlog item is TD, Avgeriou et al. simplified the decision. They specified all constructs as TD that generate interest when being changed (Avgeriou et al., 2023).

2.2. Technical debt types

Various works addressed the challenge of describing different TD types based on the specific part of the software or aspect of its development that the technical debt item affected (Li et al., 2015; Ernst et al., 2021). In their tertiary studies, Rios et al. (2018) found 15 different TD types, those being design, code, architectural, test, documentation, defect, infrastructure, requirements, people, build, process, automation test, usability, service, and versioning TD. Moreover, new types like security debt (Ahmadjee et al., 2021) or update debt (Wiese and Borowa, 2023) are still emerging. Sometimes, TD type names are not consistent across different works, e.g., Rios et al. (2018) mention "people debt", which is similar to "management and social debt" defined by Ernst et al. (2021).

2.3. Technical debt causes and consequences

McConnell and Fowler distinguish between TD caused intentionally or unintentionally, i.e., unconsciously (McConnell, 2008; Fowler, 2009). Intentional TD items result from a conscious decision, e.g., accepting a workaround to reach a deadline. Unintentional TD items result from missteps like poor design at the code level or architectural decisions that, over time, turn out to have been unfavorable.

Regarding the consequences, Schmid presents the distinction between potential and effective TD (Schmid, 2013a). Effective TD is TD that is currently creating interest, e.g., poorly readable code that must be frequently changed. Potential TD is TD that is rarely or never changed, e.g., poorly readable code in a legacy system that will be rewritten soon.

The main causes and consequences of TD have been extensively studied and categorized by the InsighTD project (Rios et al., 2020; Ramač et al., 2021). The InsighTD project describes eight categories of causes (e.g., "lack of knowledge", "external factors", "development issues", or "planning and management") and six categories of consequences (e.g., "external quality issues", "internal quality issues", "organizational", or "planning and management") (Ramač et al., 2021). Our study on the IT manager's perspective on TD found that cause and consequence categories are often identical and form vicious cycles (Wiese and Borowa, 2023). We found the following categories for causes and consequences equally: "time and budget", "business", "management", "technologies", and "personnel/human resources". Besides, we identified two more categories for causes, i.e., "decision-making" and "awareness". Moreover, causes and consequences often form causal chains, e.g., time pressure leading to bad decision-making (cause) or not keeping a given deadline, leading to customers leaving the company (consequence).

2.4. Technical debt management and activities

TDM has been intensively researched over the last decade, which resulted in two tertiary studies on this topic (Junior and Travassos, 2022; Rios et al., 2018) and two comprehensive management guidelines (Kruchten et al., 2019; Ernst et al., 2021) intended for software practitioners. Most TDM studies describe their approaches based on the so-called *TD activities*, which were first described in a mapping study by Li et al. (2015).

TD identification. This comprises methods to detect TD such as (1) static code analysis, e.g., SonarCloud (Sonar, 2025), (2) analyzing self-admitted TD, e.g., in "todo" tags in code comments (Li et al., 2022), or (3) recording TD in a TD backlog (Kruchten et al., 2019; Junior and Travassos, 2022). While automatic TD detection is repeatedly researched, automation tools often have limitations regarding the TD types, e.g., only identifying code debt but not architecture debt (Avgeriou et al., 2023). Thus, manual TD identification is still frequently used, e.g. Oliveira et al. (2015).

TD documentation. This is the term for representing TD items and their attributes in lists, e.g., as a supplementary item type in a backlog of work items. In backlog tools, work items are commonly referred to as tickets, i.e., one TD item is represented by one TD ticket. TD backlogs and lists lead to more visibility, make TD explicit, raise the perception of TD, and by this, raise the overall awareness for TD (Kruchten et al., 2012a; Eliasson et al., 2015; Guo et al., 2016b; Besker et al., 2019).

TD measurement. This activity tries to quantify TD, e.g., by estimating costs for interest and principal or a cost-benefit ratio (Tom et al., 2013; McConnell, 2008; Ramasubbu and Kemerer, 2019). In their recent systematic mapping study on quantification approaches, Perera et al. identified that most research focuses on the cost aspects, and benefit is rarely analyzed (Perera et al., 2024). Moreover, they found that research focuses on the cost retrospectively, i.e., on evaluating TD items after they are incurred. Research rarely focuses on cost estimations to guide decisions on incurring TD (Perera et al., 2024).

TD prioritization. This activity ranks TD items depending on their importance. TD can be prioritized against functional requirements or each other (Lenarduzzi et al., 2020). Prioritizing TD can be a one-time activity, but it is often an iterative activity, as a TD item's priority might depend on the current situation (Lenarduzzi et al., 2020). Multiple factors influencing a TD item's priority have been uncovered by academia, some of which we present in Table 3. However, in their systematic literature review on TD prioritization, Lenarduzzi et al. uncovered 53 prioritization factors organized into the categories: business, customer, evolution, maintenance, system qualities, quality debt, productivity, project factors, social factors, other factors, and "not specified" (Lenarduzzi et al., 2020).

TD prevention. This activity aims to avoid TD incurrence. The InsighTD project identified various TD prevention strategies used in the industry (Rios et al., 2020; Freire et al., 2020; Pérez et al., 2021). We listed these and further prevention strategies from academia in Table 3.

TD monitoring & visualization. This is concerned with observing a system's TD development over time (Li et al., 2015). Visualizations of TD development based on lists of TD items can support monitoring TD. Various tools for TD visualization have been developed: DebtFlag (Holvitie and Leppänen, 2013), MultiDimEr (Silva et al., 2022), VisminerTD (Mendes et al., 2019), MIND (Falessi and Reichel, 2015), and TEDMA (Fernández-Sánchez et al., 2017). They visualize self-admitted TD, bug reports, static analysis metrics, or collections of TD items of a proprietary tool.

TD repayment. This is the term for resolving the TD item. Buschmann et al. introduced three repayment categories: ignore, refactor, and rewrite (Buschmann, 2011). Various repayment methods are described in research as presented in Table 3. Ignoring includes only repaying TD if it impedes a functional requirement (Freire et al., 2023) or paying the interest (Buschmann, 2011). Rewriting involves waiting for the discontinuation or migration of a software system (Schmid, 2013a,b). Refactoring includes repaying TD (1) if the cost-analysis proves the positive effect (McConnell, 2008; Tom et al., 2013), (2) by assigning a repayment quota (McConnell, 2008; Wiese et al., 2022), (3) before a project if TD might hinder the development (Schmid, 2013a), or (4) after a project if TD was incurred due to its project deadline (Wiese et al., 2022).

3. Related work

Our study aimed to establish a TDM process utilizing action research. We analyzed not only the resulting process but also the change in practitioners' TD awareness. Hence, our related work section focuses on action research studies on TDM processes and the topic of TD awareness.

3.1. Establishing a technical debt management process

Two recent tertiary studies on TDM by Rios et al. (2018), and by Junior and Travassos (2022) give an overview of the TDM research. They both found that TDM is viewed as a set of TD activities and supporting strategies and tools, but they did not document any research on the systematic establishment of a TDM process. Both agreed that a holistic approach to TDM, including various TD activities and types, is missing. Similarly, the vision paper by Avgeriou et al. identified that TD research needs to focus on multiple TD types instead of only focusing on code debt and should include more industrial datasets (Avgeriou et al., 2023). This coincided with our finding that IT managers lack suitable guidelines for establishing TDM (Wiese and Borowa, 2023). A general guideline for establishing TDM was presented in the book of Kruchten et al. (2019), mainly in the "What can you do today?" sections. Yet, the feasibility of this guideline has not been evaluated.

3.2. Action research studies on technical debt management

Five action research studies established TDM processes in companies (Oliveira et al., 2015; Yli-Huumo et al., 2016b; Borup et al., 2021; Detofeno et al., 2021, 2022).

Detofeno et al. performed two action research studies to implement a TDM process in a Brazilian company (Detofeno et al., 2021, 2022). The first study (two cycles in two years) established a TDM guild, i.e., a community of practice, on the TD topic (Detofeno et al., 2021). The guild contributed to the identification, monitoring, prevention, prioritization, and repayment of TD. However, the actions were limited to code debt since they focused on static code analysis of PHP code. In the second study (two cycles in two years and eight months), they developed a detailed TD prioritization mechanism but still limited their work to code debt (Detofeno et al., 2022).

Oliveira et al.'s study (Oliveira et al., 2015) (three action cycles in five months) intended to confirm the feasibility of the TDM framework presented by Seaman and Guo (2011). Their approach focused on identifying, documenting, and prioritizing existing TD items. Similarly, Yli-Huumo et al. (2016b) focused their work on identification, documentation, and prioritization. They concentrated on uncovering the risks and benefits related to these TD activities. They performed five subsequent research steps: problem identification, TD identification, TD documentation, TD evaluation (i.e., measurement), and TD prioritization. However, they both missed the activities of TD prevention and monitoring (Li et al., 2015) as well as the multiple ideas on TD repayment (Li et al., 2015; Buschmann, 2011).

Borup et al.'s study (Borup et al., 2021) (two interventions in ten months) focused on TD deliberation, i.e., communicating about, weighing, and reflecting on TD. Therefore, they might have filled some of the gaps in the previous studies, particularly in TD prevention, but they did not focus on TD Management, e.g., documenting or prioritizing.

These action research studies show shortcomings that we address in our study. First, our study focused on all major activities and did not differentiate TD types other than tagging backlog tickets with the respective TD type. Second, we performed long-term research (five action cycles in 16 months) to evaluate whether processes are established and the team's TD awareness is improved sustainably. Only the works of Detofeno et al. performed studies longer than ours; however, they had fewer action cycles and, therefore, fewer possibilities for intervention. Third, we observed team discussions as valuable supplemental data sources, which was not done by any other study. Finally, we focused on developing a TDM process together with practitioners, reinforcing an agile mindset rather than simply evaluating a pre-defined approach developed by researchers. This is specifically relevant in agile teams to create acceptance for the TDM process among the practitioners.

3.3. Technical debt awareness

To analyze the current state of research on TD awareness and identify potential measures, we conducted a literature review and explored five established digital libraries: IEEE Xplore, ACM Digital Library, SpringerLink, ScienceDirect, and Wiley. We used **("technical debt") AND (aware OR awareness)** as a search string and searched titles, abstracts, and keywords to identify literature about TD Awareness. We did not do full-text searches as the terms "aware" and "awareness" are frequently used in everyday language. We may have missed some papers, but we aimed to gain an overview of the academic discussions on TD Awareness and assumed that we found all the literature focusing on TD Awareness. After removing ten duplicates, we studied 59 unique references (see additional material (Wiese and Serwa, 2025)).

We found out that TD awareness is an ambiguously used term. Some studies interpreted TD awareness as knowing what TD is Apa et al. (2020), Besker et al. (2017), Holvitie et al. (2018), Baars and Meester (2019) and Zampetti et al. (2021); other studies used it as knowing the TD items in their own system (Besker et al., 2017; Liu et al., 2020), and some interpreted TD awareness as not unintentionally incurring TD (Rocha et al., 2017; Wiese et al., 2022). Many papers mentioned TD awareness as a prerequisite for TDM (Besker et al., 2017; Kruchten et al., 2019). Some papers have identified visualization (Eliasson et al., 2015), gamification (Crespo et al., 2022), and TDM (Baldassarre et al., 2020; Wiese et al., 2022) as feasible methods for raising TD awareness. Tonin et al. conducted a study specifically focusing on the effect of elevated TD awareness by regularly discussing this topic during IT students' development projects (Tonin et al., 2017).

Unfortunately, few papers explicitly defined what they meant by the term TD awareness or described their measurements. We define TD awareness for decision-making situations in Section 4.3.1. To the best of the authors' knowledge, TD awareness in decision-making situations such as team meetings has not yet been researched.

To evaluate TD awareness, researchers commonly use questionnaires (Berenguer et al., 2021; Wiese et al., 2022), which have the disadvantage of being subjective measurements. Others deduced TD

awareness from indicators derived by static analysis tools (Crespo et al., 2022), which only indirectly point to TD awareness. Kruchten et al. (2019) introduced five levels of TD awareness (from not knowing the term TD to having control over their TD). However, these levels indicate the maturity of the TDM process, and likewise, can only indirectly point to a team's TD awareness.

In our study, we addressed these shortcomings by supplementing meeting observations, which, to the best of the authors' knowledge, have not been used previously. Additionally, we identified potential measuring methods from the field of Psychology. We found conceptual similarities to TD among three awareness concepts: contingency awareness, bounded awareness, and situation awareness. We finally adopted a measuring method for situation awareness, which we further explain in Section 4.3.1. This allowed us to provide more objective measurements and quantitative data in addition to the qualitative data.

4. Method

Our study aimed to create a TDM process in cooperation with our industry partner based on previous academic findings. From a business perspective, this resembled a small change management project as work processes were changed (Führing, 2021). For such a change process, it was essential to include the employees who have to work with these processes (Führing, 2021). Moreover, the agile work concepts adopted in most IT companies heavily rely on a team's self-organization and do not promote process changes that are enforced by higher hierarchical levels (Meyer, 2014; Maximini, 2018).

Action research (Staron, 2020), an approach in which both the practitioners from a company and the research group work together to understand a problem and develop a solution, seemed especially suitable for the purpose. Therefore, we chose this method to answer RQ 1&2, determining the adopted approaches and the time required to establish the TDM process. We collected data from the workshops and subsequent discussions in the action cycles, workshop questionnaires, and, on rare occasions, interviews. Additionally, we analyzed the backlog development.

Along with the action research, we analyzed the effect the TDM establishment had on the participants' awareness of TD to answer RQ 3. For this purpose, we collected three kinds of data throughout the whole action research. The participants completed questionnaires during the workshops, we observed the team during their meetings, and we adopted an approach from psychology to assess the participants' TD situation awareness during meetings, which we call TD-SAGAT.

In the following sections, we introduce our industry partner, outline the basics of the action research, explain the analysis of the team's backlog, and introduce the interviews we used. Subsequently, we introduce the three data-gathering methods used to analyze the participants' awareness of TD. The questions used for the action research and awareness analyses were included in the same questionnaires and are, thus, explained together in Section 4.3.3.

4.1. Industry partner

For this research, we were cooperating with a German company with over 2000 employees that had been in operation for over 70 years. The company produces signal-processing products, which means the software development of this company is located in the embedded, i.e., the mechatronic domain. The company chose not to disclose its name, so the information in this article and the additional material were anonymized.

4.1.1. Team

For our study, we worked particularly with one IT team that was developing and maintaining a central framework used by other IT teams in the same company. Informally, the team was divided into two subteams, the *algorithm subteam* focusing on the signal processing algorithms and the *framework subteam* on the supporting framework. While the algorithm subteam did not further differentiate different roles, the framework subteam had a team architect, a quality assurance specialist, and one specialized developer with an engineering background to bridge the gap between the disciplines. As the algorithm subteam did not suffer from much TD, we reported on their insights during the action cycles (RQ1, Section 7.1) but focused our awareness evaluation on the framework subteam (RQ3, Section 7.3). Table 1 shows an overview of all team members.

Table 1
Study participants with their role, subteam assignment to Framework (F) or Algorithms (A), and Action research (AR) team assignment to action team (A) or reference team (R).

Participant	Role	Subteam	AR team	Years in IT	Gender
P1	Team manager	–	A	3–5	f
P2	Team architect	F	A	6–10	m
P3	Developer and engineer	F	R	>10	m
P4	Quality assurance	F	A	0–2	m
P5	Developer	F	R	>10	m
P6	Developer	A	A	6–10	m
P7	Developer	A	R	6–10	f
P8	Developer	A	R	0–2	m

The team was newly assembled but worked on an existing framework with lots of TD, which, therefore, were the first focus of this team. During the study, this focus shifted to implementing more new functionalities. Further, the team continuously improved the processes it had just established.

4.1.2. Meetings & Backlog

The team used an adapted Scrum process with three-week sprints ending with a retrospective. The team had daily stand-up meetings and organized its work tickets in the backlog tool Microsoft Azure DevOps (Microsoft, 2025a). Their backlog was configured as an agile process based on a "portfolio backlog" defined by Azure DevOps (Microsoft, 2023, 2024). The team used the following hierarchically ordered work item types for functional requirements defined by that process: initiatives, epics, features, user stories, and tasks. Additionally, they employed a particular ticket type for bugs. During the study, they added another specialized type for TD. Their tickets could have had six different statuses, i.e., their Scrum board comprised six columns typical for Scrum processes: backlog, in proposal, planned, in progress, blocked, testing, and done.

At the beginning of our study, the team had three meetings to organize their work:

- *Grooming*: Recording necessary attributes for new tickets, e.g., description, effort, or acceptance criteria. After being "groomed", the tickets were not given a specific status other than "backlog".
- *Refinement*: Selecting and preparing the tickets for the next sprint and finalizing attribute information. After being refined, the tickets transitioned from the status "backlog" to "in proposal".
- *Review & planning*: Reviewing the tickets of the past sprint and planning the tickets for the next sprint. After being "planned" the tickets transitioned from the status "in proposal" to "planned".

During our work with the team, the meeting structure has changed. The two subteams decided to separate their meetings from each other and reduce the number of meetings. The framework subteam decided on two meetings: A refinement meeting for analyzing tickets and recording their attributes (leading to the ticket status "in proposal") and a planning meeting to review the old and organize the next sprint (leading to the ticket status "planned"). The algorithm subteam decided on one meeting combining refinement & planning.

4.2. Action research

Multiple research studies identified the adoption of TDM processes in the industry as insufficient, e.g., Martini et al. (2018) and Wiese and Borowa (2023). Introducing a TDM process aims to change work processes, and action research is a research method that aims to create and improve work processes. The research is conducted in iteratively repeating action cycles, which comprise five phases each:

- Diagnosing: identifying which problem should be solved
- Action planning: identifying a set of actions to solve the problem
- Action taking: execution of the identified actions
- Evaluating: analyzing the actions' effects and whether they solved the problem
- Learning: Identifying improvements to enhance the action's efficacy in effectively addressing the problem.

The action research terminates when the problems are solved, which makes it crucial to define a termination goal before starting the action research. We decided to end our action research when the participants noted their satisfaction with the developed process in a questionnaire.

Our iterative action research design, including all five workshop topics and major actions taken, is presented in Fig. 1. A detailed overview of how we conducted our action cycles is presented in the following section.

Action research allows the researcher to interact with and influence the study's participants (Staron, 2020). Accordingly, action research is interpreted as a field experiment with regard to the ABC framework (Stol and Fitzgerald, 2018), which means the study is conducted in a natural, pre-existing setting. In this regard, it is similar to design science research. Yet, in contrast to changing work processes, the primary aim of design science research is to develop an artifact that is suitable for practitioners. The downside of field experiments is that the precision of measurement is affected by "confounding contextual factors", and measurements cannot be statistically generalized (Stol and Fitzgerald, 2018). For action research, *ACM Special Interest Group on Software Engineering*'s empirical standards accept this lack of generalizability or reliability in favor of, e.g., reflexivity, credibility, resonance, usefulness, and transferability (SIGSOFT, 2025). Nonetheless, we took measurements to counteract biases. For example, we followed Staron's (2020) proposition to separate the team into an *action team* to execute the actions and a*reference team* to monitor the soundness of the action team's work (see Table 1). We informed the team members about their roles and their roles' tasks at the beginning of each workshop. To further reduce biases, the researchers did not decide on the actions that were taken. For example, researchers could hint that the current amount of meetings and their structure lead to irritations, but it is the team's task to choose if and in which way to change the meeting structure.

In the following, we present our action cycles, the accompanying backlog analysis, and the two interviews we employed during the action research. We present the questionnaires used during the workshops in Section 4.3.3 together with the questionnaires and questions for the awareness analysis.

4.2.1. Action cycles

We conducted our action research in action cycles, each consisting of the five phases as outlined in the previous section. An overview of the cycles is presented in Fig. 1.

Each cycle started with a one-day workshop. A more detailed agenda of the workshops and the employed workshop methods are explained in Section 5. In the morning, the participants were presented with the results of the last cycle's evaluations (results of "evaluating"). They discussed these results and developed ideas for improving the process ("learning" and "action planning"). The first workshop started with a general introduction and motivation for TD in the morning instead of the "evaluating" and "learning" parts (see Section 5). In the afternoon, the researchers presented a new TD activity with its respective research results and approaches to the participants. The team discussed these approaches ("diagnosing") and decided which approach to adopt, adapt to their needs, or dismiss ("action planning"). Between workshops was the time for "action taking". Some actions were ongoing, e.g., documenting TDs when they occur, while others were one-time actions, e.g., consolidating the backlog by tagging TD tickets or reviewing the existing TD tickets to record new attributes. The action team performed one-time actions, including making decisions on details, e.g., the content of attribute combo boxes. The results were presented to the complete team during the regular meetings or the following workshops and discussed if necessary.

The first five workshops each had a focus on one or two TD actions, these being: TD identification & definition, TD prevention, measurement & documentation, TD prioritization, TD repayment, and TD visualization & monitoring. After that, the workshops were reduced to retrospectives that focused on the "evaluating", "learning", and "action planning" parts. The cycles and, therefore, the study ended when all participants found the process sufficiently defined and working except for minor changes, which we analyzed with a progress questionnaire (see Section 4.3.3).

4.2.2. TD backlog

In the following, we explain how the team changed their backlog during the action research by introducing a TD ticket type and corresponding attributes. These changes were part of the action research, as employing a TD backlog is a significant step in establishing a TDM process. The analysis of the TD backlog was part of the "evaluating" phase of each action cycle. The illustrations helped the participants gain insights into their backlog data and evaluate the usefulness of having and monitoring TD items and their attributes. As such, the changing backlog is primarily part of the actions (and explained in Section 7.1) and only secondarily a method to analyze the action research's success. Furthermore, we used the number of TD items as an indicator to support RQ3.

Data collection. For each workshop, we gathered the TD tickets from the backlog by exporting and analyzing them using Microsoft Excel (Microsoft, 2025b). In the consolidation phase, the team tagged 65 items as TD. For the last cycle, i.e., the retrospective, we used the visualizations directly from the visualization tool developed during the action research (see Section 6.6).

The TD ticket type was established after the first workshop. It was extended by the following attributes after the second workshop, which we used for our evaluations:

- *Interest*: The five elements of the interest scale were: "very low (less than a minute)", "low (several minutes)", "medium (one hour)", "high (one day)", and "very high (more than one day)".
- *Interest probability*: The elements of the interest probability scale were: "very low (less than once a year)", "low (once a year)", "medium (once a month)", "high (once a week)", and "very high (once a day)".
- *Contagiousness*: Contagiousness, i.e., whether the principal might grow over time, was initially measured on a five-step scale from very low to very high. In the fifth cycle, the team decided to simplify this measurement to a simple "yes/no" check box.
- *Breaking change*: A simple "yes/no" checkbox recorded whether the implementation incurred breaking changes, i.e., effects on other components.
- *Component (Area)*: Components were structured in a hierarchy utilizing Azure DevOps' standard attribute "area".
- *Effort (principal)*: This value ranged from one to five, where one to four means one to four person days, and five means more than five days. In the fifth action cycle, the team decided to denote the effort on a continuous scale in person-days.

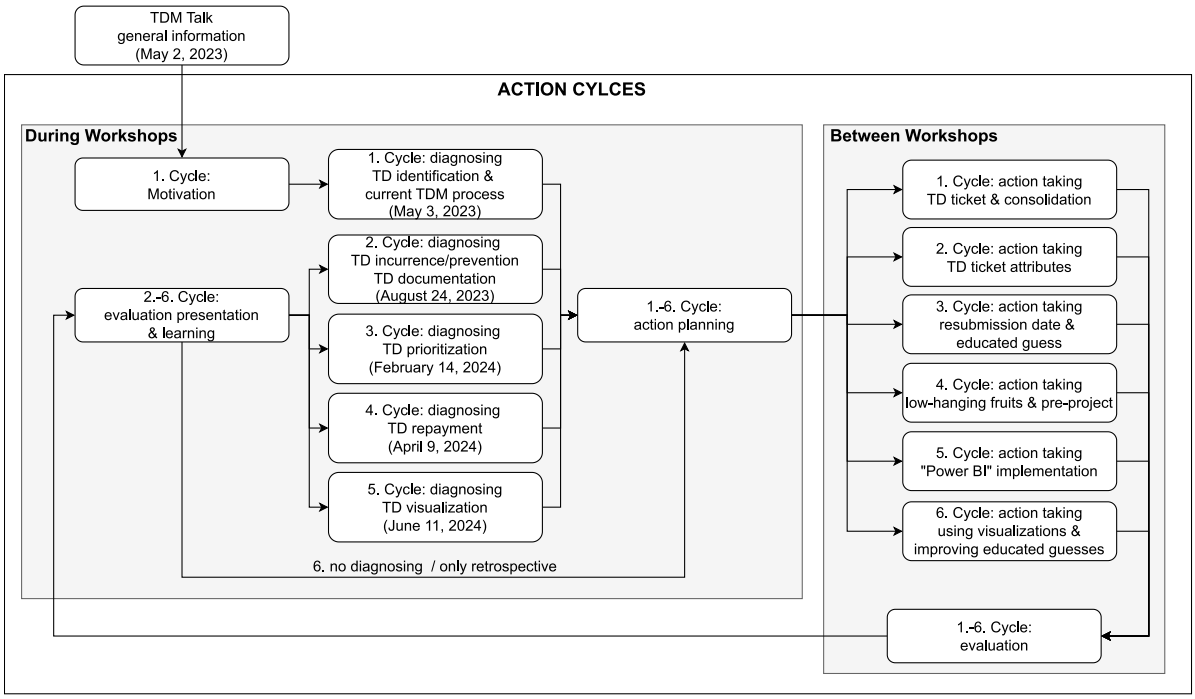


Fig. 1. Study design - iterating action cycles including five workshops on TD activities.

Data analysis. For the first workshop, we did not evaluate the backlog items as TD had not been tagged yet.

For the second and third workshops, we evaluated the number of opened and closed TD tickets per month and illustrated the number of TD versus functional and bug tickets.

From the fourth workshop on, we evaluated the new attributes as follows:

- number of opened and closed TD tickets per month,
- share of tickets per TD type,
- share of tickets per estimated priority,
- share of tickets per *interest risk*, i.e., the interest ID (from 0 to 4) times the interest probability ID (from 0 to 4),
- share of tickets per estimated contagiousness,
- share of breaking changes, and
- share of tickets' *interest risk* versus their effort.

For the fifth workshop, we added a calculation based on the idea of a *return on investment* (ROI) value to experiment with the idea of calculating a priority value. Due to the experimental setting, we made some assumptions:

- For the interest, we assumed "several minutes" to be 15 min and "more than one day" to be two days and normalized all values to *interest in minutes*. This led to the values of 1, 15, 60, 480, or 960 min (very low to very high) for the interest.
- For interest probability, we normalized the selected values to *times per month*, leading to the values of 30, 4.5, 1, 0.0027, or 0.0013 times per month (very low to very high).
- For the effort, we assumed 10 days for a given five days, as long as the team did not estimate their effort on a continuous scale. After this change, we used the given effort for calculations. We multiplied the effort by 60 to determine the *effort in minutes* (*EFFORT*).

Using these values, we identified the *interest in minutes per month* (*INTEREST*) for each TD item by multiplying *interest in minutes* and *times per month*. We set the *interest in minutes per month* against the *effort in minutes* to calculate the number of months it takes to achieve an ROI (ROI in months – *ROIM*) as follows:

$$ROIM \text{ months} = \frac{EFFORT \text{ min}}{INTEREST \text{ min/months}}$$

We acknowledge the limitations of this calculation, i.e., (1) that we made assumptions, (2) the underlying values were all estimations, and (3) the values on the scale were not equidistant, i.e., the distance between "once a year" and "once a month" is not the same as the distance between "once a week" and "once a day". We separated the results into the following categories so that the items were distributed evenly for the most part, and large gaps between the ordered single values also led to different categories:

- $ROIM < 1$ month → very high,
- $ROIM < 2$ months → high,
- $ROIM < 1$ year → medium,
- $ROIM < 3$ years → low,
- $ROIM \geq 3$ years → very low.

We only used these values experimentally as a basis to discuss a calculated priority with the participants as part of the action cycle. The limitations and ways to counteract these limitations, as well as correct assumptions, were part of the discussion as presented in Section 6.6.

For the sixth workshop, i.e., after implementing the backlog's visualization in the fifth cycle, we used the visualization directly, and no additional analysis was necessary.

4.2.3. Interviews

We used an informal interview to get initial insight into the team's members and processes. Further, we conducted semi-formal interviews to develop the visualizations.

Data collection. At the beginning of the study, we consulted the team's IT manager for information about the current TDM status, the current working processes, including ticket types and status, and the stakeholders, including other IT managers and team members.

For the visualization, we used semi-structured interviews to gather use cases for a visualization. We interviewed the action team members, asking the following questions:

Table 2
Overview of all questions from the kickoff (K), progress (P), and TD-SAGAT questionnaire (S), their answer types, and their reference to the RQs.

No.	Question	Sub-Question or answer-options if applicable	Answer type	Questionnaire	Used to answer
1a	When I decide for/against	What is an alternative solution option to my favorite solution option?	y/n	KP	RQ3
2a	an architecture, a design,	What are the advantages of my favorite solution option?	y/n	KP	RQ3
3a	or a way to implement a	What are the disadvantages of my favorite solution option?	y/n	KP	RQ3
4a	requirement, I know ...	What are the risks of the supposedly faster solution?	y/n	KP	RQ3
5a	(questions: 1, 2, 3, 4, 5, 8)	How much effort is required for a supposedly ideal solution?	y/n	KP	RQ3
6a	OR When I decide	How much will the repayment (the refactoring) cost?	y/n	KP	RQ3
7a	for/against a refactoring	How much effort will it take to keep the current solution?	y/n	KP	RQ3
8a	(TD repayment), I know ...	What competing functional (technical) requirements are there?	y/n	KP	RQ3
9a	(questions: 6, 7, 9, 10	Which components are affected by the repayment (the refactoring)?	y/n	KP	RQ3
10a		What are the requirements for the quality attributes (e.g. availability, performance, usability, maintainability) for the affected component?	y/n	KP	RQ3
1b	Regarding the topic	What alternatives are there to the solutions discussed?	y/n	S	RQ3
2b	currently under discussion,	What are the advantages of the solutions discussed?	y/n	S	RQ3
3b	I considered ... (Please	What are the disadvantages of the solutions discussed?	y/n	S	RQ3
4b	answer spontaneously! If	What are the risks of the solutions discussed?	y/n	S	RQ3
5b	you have to think long and	How much effort is required for the solutions discussed?	y/n	S	RQ3
6b	hard about whether you	How much effort (if any) will it take to convert the discussed solutions into the ideal solution (repayment of TS)?	y/n	S	RQ3
7b	have (just) thought about something, the answer is probably – No)	How high, if any, is the (additional) effort required to permanently maintain the solutions discussed? (Interest from TS)	y/n	S	RQ3
8b		What competing functional (technical) requirements are there?	y/n	S	RQ3
9b		Which components are affected by the solutions.	y/n	S	RQ3
10b		Which quality attributes are influenced.	y/n	S	RQ3
12	How much time (in person-hours) did you invest in initializing the technical debt management process (e.g., workshop participation or backlog consolidation) in the last cycle?		Number	P	RQ2
13	How much time (in person-hours) did you invest in the process of managing technical debt in the last cycle (excluding initial efforts such as workshop participation or backlog consolidation)?		Number	R	RQ2
14	Do you consider the amount of technical debt in the current backlog to be a consistently appropriate amount?		y/n	P	RQ1
15	Do you think the amount of technical debt taken on in the last cycle was appropriate?		y/n	P	RQ1
16	Do you think the amount of technical debt repaid in the last cycle was appropriate?		y/n	P	RQ1
17	Are you satisfied with the process, backlog and technical debt situation? Or should we further optimize the process?		y/n	P	RQ1
18a	What are your goals related to the process? What should change in your everyday life? How should technical debt be handled at the end of the project?		Text	K	Supports action
18b	What are your goals related to the process? What is going “really well” now? What should we improve in the next step?		Text	P	Research (RQ1)
19a	What expectations do you have for the project? How should the project work? What should not be forgotten? What should be given less importance? What should definitely not happen?		Text	K	Supports action
19b	What expectations do you have for the project? What is going “really well” in the project? What should change?		Text	P	Research (RQ1)

- Could you please name two to three use cases in which the visualization tool may be useful?
- Do you have ideas for the content of potential visualizations?
- Why do you want to see this specific content?
- Which ticket attributes would be particularly useful for a visualization? Which ones would not?
- Are there any non-functional requirements, e.g., security, for the application regarding your team/company?
- Do you have any other comments on the tool or the visualizations?

Data analysis. The interviews were analyzed informally. The first interview was used for understanding and reporting the team’s processes. The results of the semi-formal interviews on potential visualizations were assembled to create the mock-ups discussed during the “diagnosing” phase of the fifth workshop as presented in Section 6.6 and Table 4. The answers to the question regarding non-functional requirements were used to specify the visualization environment and supported the decision to use Microsoft Power BI (Microsoft, 2025c).

4.3. Evaluation of TD awareness

To answer RQ3, we continuously evaluated the team’s awareness of TD in decision-making situations using three different measurement methods: Questionnaires during the workshops for a self-assessment, observations of team meetings by researchers for an external assessment, and TD-SAGAT for an immediate assessment in decision-making situations. For comparability, all three methods used the same ten questions presented in Section 4.3.1. However, additional questions were evaluated for observations and questionnaires. As mentioned previously, the additional questions were related to the action research. An overview of all questions is presented in Table 2.

We explain each method in the following three sections.

4.3.1. Situation Awareness Global Assessment Technique for TD (TD-SAGAT)

As TD awareness is ambiguously used in TD research (see Section 3), we searched for a way to define TD awareness more precisely for our study and determine a fitting measurement method. Together with a researcher from the psychology domain, we searched a database of scientific papers that is used predominantly in psychology (i.e., PsycInfo Association, 2025) for awareness concepts and subsequent measuring methods. We identified 17 distinct awareness terms and their corresponding definitions. We dismissed 14 terms, e.g., ambient or phonemic awareness, that did not conceptually overlap with TD awareness (see Wiese and Serwa (2025)). We identified conceptual similarities to contingency awareness, bounded awareness, and situation awareness.

Contingency awareness refers to the relationship between two events, which appears to fit the TDM problem, where the causes and consequences of TD are often not linked by stakeholders (Wiese and Borowa, 2023). Within the context of affective learning, contingency awareness is defined as the relation between conditional and unconditional stimuli (Dawson and Schell, 1987). This refers to whether people

are aware that one stimulus (e.g., a neutral image, such as a balloon) has been influenced by a simultaneous stimulus (e.g., an electric shock). However, this measurement method is poorly adaptable to TD Awareness because the stimuli (TD incurrence and TD consequences) have to be shown to participants, yet are too abstract in the case of TDM.

Bounded awareness refers to the fact that people often focus on a limited portion of the information available and make decisions based on that rather than considering it in full (Chugh and Bazerman, 2007). This bias often stems from the information people are requested to focus on. This means that if a developer's goal is to finish an assignment within a certain time frame, they will focus on software that works, not whether they have optimized it to limit TD. Unfortunately, there is no metric specifically designed to measure bounded awareness. Instead, measurements have been designed for specific examples (Bazerman and Sezer, 2016).

We found that **Situation Awareness (SA)** (Endsley, 1988) fits our concern as it focuses on decision-making, which we consider to be the underlying concern of TDM. IT stakeholders must make two essential decisions in TD: Whether to incur TD or not, and whether to repay it or not (Wiese and Borowa, 2023; Buschmann, 2011). Consequently, we deemed the corresponding Situation Awareness Global Assessment Technique (SAGAT) (Endsley, 2000) to be the most appropriate research method for our concern. Moreover, this was the approach that most objectively measured awareness.

Situation Awareness (SA). *Situation awareness (SA)*, as proposed by Endsley (Endsley, 1988), "refers to the perception of the elements of the environment within a volume of time and space, the comprehension of their meaning and the projection of their status in the near future" (Zhou et al., 2023). Going with this definition of SA, we defined TD awareness:

> **TD awareness** is the (1) **perception** of the prerequisites (e.g., alternatives, benefits, drawbacks) of a decision (e.g., incurring TD) during decision-making situations (e.g., refinement meetings), (2) the **comprehension** of these prerequisites' meaning, and (3) the **projection** of their status in the future (e.g., TD's consequences).

With the workshops and information we give the participants, they get a primary perception of the prerequisites (1). By researching this TD awareness over the course of the study, we want to analyze whether the participants comprehend those prerequisites (2) and whether they are able to use them in decision-making, i.e., for the projection of future effects (3).

Situation Awareness Global Assessment Technique (SAGAT). The idea behind the *Situation Awareness Global Assessment Technique (SAGAT)* is to interrupt participants in their work or a simulation of their work and ask them questions about their situation (Endsley, 2000). Interrupting participants with questions is still a subjective assessment. Yet, participants asked about their current thought processes are more likely to provide objective answers as opposed to answering retrospectively (e.g., at the end of the week) (Schwarz et al., 2007).

Moreover, this approach includes a process for systematically breaking down the specific awareness to its prerequisites, i.e., pieces of information needed to make informed decisions. The main *goals* and *subgoals* for the situation lead to *decisions* that have to be made to achieve the (sub)goal. The *SA prerequisites* are then derived for these decisions. Finally, the appropriate *queries* are designed from these prerequisites, questioning the participants about whether they considered the prerequisites during the decision-making before interruptions (Endsley, 2000). In the original papers on this technique, these "prerequisites" are also called "requirements". To avoid misunderstandings regarding "(non-)functional requirements", we use the term "prerequisites" in this paper.

TD-SAGAT questionnaire development. We adopted the SAGAT approach and named this specific method *TD-SAGAT*. First, we utilized the systematic query-creation aspect of the SAGAT approach to collect TD awareness prerequisites and subsequent queries. We defined the following as the main goal and inferred subgoals from the TD definition from Avgeriou et al. (2016) and the most common cause for TD according to the InsighTD project, which is time pressure (Rios et al., 2019):

> **GOAL:**
> Have a system that is easy to change and fulfills the users' (functional) requirements.
> **SUBGOALS:**
> (1) Develop a new system that is easy to change within a given timeframe.
> (2) Implement a new functionality in a system so that it is still easy to change within a given timeframe.
> (3) Refactor (incl. architectural refactorings) a system so that it is easy to change after the refactoring with the least amount of time.
> (4) Maintain a system (update its components) so that it is easy to change with the least amount of time and effort.

From these subgoals, we derived various decisions and respective prerequisites (see additional material Wiese and Serwa, 2025), e.g.: *Could the preferred solution option incur TD?* The resulting SA prerequisites were knowing (1) alternative solutions, (2) the benefits, and (3) the drawbacks of each solution. Another decision was: *Should we incur this TD item or not?* The resulting SA prerequisites were knowing (1) the effort for the assumed best and the TD-incurring solution, (2) the principal and interest, (3) the risks of both solutions, and (4) the competing functional requirement. Finally, this led us to the following ten prerequisites:

> **PREREQUISITES:**
> (1) solution alternatives, (2) solution's benefits, (3) solution's drawbacks, (4) solution's risks, (5) effort for implementing a solution, (6) TD item's principal, (7) TD item's interest, (8) component affected by TD, (9) functional requirement potentially competing with a TD item, and (10) quality attributes affected by TD.

To validate the resulting prerequisites, we surveyed six experts gathered from our personal network and had them create their own hierarchy of goals, subgoals, decisions, and prerequisites. Three experts were software engineers with over 20 years of IT experience, each from a different domain. The other three experts were researchers who had all published peer-reviewed papers on TD. We did not discuss the topic of TD awareness with them prior to this study. Comparing their input with our initial design, we found that the goals and subgoals were sometimes defined slightly differently. However, we did not receive further input regarding the prerequisites, which we interpreted as confirmation of our prerequisites.

Data collection. We identified meetings in which tickets were discussed, evaluated, and estimated, such as agile grooming/refinement meetings, as situations when to use our TD-SAGAT method. We applied this method starting in August 2023 and used the TD-SAGAT interruptions only once a month to minimize the Hawthorne effect, i.e., observing behavior can influence the behavior (McCambridge et al., 2014; Chiesa and Hobbs, 2008). We interrupted the participants at two to three randomly chosen points in time during a one-hour meeting using a random alarm clock app (RandomTicker, 2025). The participants were then invited to complete the TD-SAGAT questionnaire (see additional material Wiese and Serwa, 2025), asking whether they

considered the prerequisites during the previous ticket's discussion with "yes" or "no" as answer options. All questions are presented in Table 2 (also see Section 4.3.3). We ordered all questions randomly to avoid systematic and contextual effects, e.g., that subsequent questionnaire items are more likely to be answered similarly (Royal, 2016). Only participants familiar with the discussed topic answered the TD-SAGAT questionnaire, which sometimes resulted in few answers. However, we assume the overall results are still valid, as we conducted multiple interruptions per meeting and only evaluated the results as a sum. Overall, we used the TD-SAGAT interruptions during 12 meetings, resulting in 90 datasets, i.e., an average of 7.5 per meeting. A step-by-step introduction to the TD-SAGAT method is part of our additional material (Wiese and Serwa, 2025).

Data analysis. For each prerequisite, we counted the number of participants considering this prerequisite per meeting, i.e., we summarized the "yes" answers across all interruptions, i.e., tickets. We then calculated the percentage of "yes" answers relative to all answers. We arranged this on a timeline to see whether the number of participants per prerequisite increased during the action cycles. In the end, we compared these results with the observations' results to see whether there is a difference in what participants think about and what they talk about.

We presented these results in each workshop's "evaluation" phase, comparing them with the observation results. For this paper, we furthermore compare them to the results of the workshop questionnaire. To enable a better overview, we split the prerequisites into three categories:

- *comparing solutions:* comprising solution alternatives, solution's benefits, solution's drawbacks, and solution's risks
- *evaluating costs:* comprising effort for implementing a solution, TD item's principal, and TD item's interest
- *considering potential consequences:* comprising the affected components, a functional requirement potentially competing with a TD item, and the affected quality attributes

4.3.2. Observations

Data collection. For the first three months, we observed the review & planning meetings. Initially, many tickets were still debated in these meetings. However, discussions became less frequent as the structure of the meetings became more established over time, and only tickets for the upcoming sprint were selected. Thus, from August on, we also participated in refinement and grooming meetings, during which much more discussion on the single backlog items occurred. Consequently, we focused our observation on these two meeting types for the rest of the study.

For the observations, we used a predefined observation protocol (see Wiese and Serwa (2025)) where the researchers marked the following observations per ticket, i.e., discussion item:

1. Ticket ID and subject to enable discussion in the next workshop's "learning & evaluation" phase
2. Ticket type given by the team (e.g., TD, bug, user story) to reveal potential deviations between backlog and discussion
3. Checkboxes to indicate whether the team versus the researchers consider a ticket to be a TD repayment to determine if there were any different assessments on what is considered a TD item
4. Checkbox to indicate whether the team runs the risk of incurring TD unconsciously, e.g., when someone was already implementing a ticket even though it has not been refined and, thus, alternatives have not been analyzed, or when a ticket was planned for the next sprint even though crucial information for the implementation was still missing
5. Checkbox to indicate whether TD was incurred intentionally to determine instances of conscious TD incurrence
6. Checkboxes for each TD-SAGAT prerequisite to indicate whether it was discussed

We explained the researcher's evaluation in a *remark* column for discussion in the next workshop's "learning" phase.

The first and second authors of this paper performed the observations. We analyzed five meetings with both researchers to align our working methods. Then, we alternated the observations between us. Overall, we observed the discussion of 415 items in 35 meetings watching 25 h of video recordings.

Data analysis. For data analysis, we counted the number of tickets (i.e., ticket discussions) for which the respective observations occurred per meeting. For May to July 2023, these numbers were based on the review & planning meetings. For the rest of the study, the values are based on the refinement and grooming meetings. To better compare the results with other methods, we calculated the percentage of counted tickets relative to the total number of discussed tickets per meeting. For the TD-related values (interest, contagiousness, and competing functional requirements), we used the number of TD tickets as the basis.

We presented these results in each workshop's "evaluation" phase and this paper, comparing the prerequisites (observation no. 5) with the TD-SAGAT results. To enable a better overview, we split the prerequisites into the same categories as mentioned in Section 4.3.1.

The additional observation items were presented separately on a timeline, focusing on the risk of unconscious TD incurrence (observation no. 3) and dispute on whether an item is TD or not (observation no. 2).

Finally, we clustered the "remarks" by common topics and used the single items as examples to support the discussion in the following workshop's "learning" phase. Similarly, information on observations no. 1 (backlog deviation) and no. 4 (intentional TD incurrence) was presented as discussion topics with examples. We did not present the values in a figure, as the instances were extremely rare, e.g., only two instances of intentional TD incurrence during the study's duration.

4.3.3. Questionnaires

Throughout the study, we used three different questionnaires: one questionnaire for the first, i.e., kickoff workshop, one to analyze the progress made so far during the following workshops, and one during the TD-SAGAT interventions.

Data collection. Each of the three questionnaires comprised the ten questions about the TD-SAGAT prerequisites in a "yes" or "no" format. However, for the kickoff and process questionnaire, we separated this evaluation into the three situations where they (a) might potentially incur TD, (b) discuss repaying TD, and (c) discuss updating the existing system, e.g., libraries or infrastructure (resulting in 30 questions).

The kickoff questionnaire additionally included questions on whether the participants had heard of TD and its related concepts, whether they were familiar with the various TD types, and how they currently manage TD. All these questions had predefined values to simplify the completion of the questionnaire. We did not add these questions to this paper as they were not used for scientific evaluation, but to get to know the participants and team. Still, we added them to the additional material (Wiese and Serwa, 2025) as they might help other researchers when replicating this study.

The process questionnaire comprised additional questions on the progress of the study, enabling us to analyze the termination point for the action research. Moreover, we asked for the time invested in the management of TD, which was separated into the initial effort and the continuous effort to answer RQ2. Finally, the kickoff and the process questionnaire additionally asked for the goals and expectations of the participants in two open questions to give participants an opportunity to express their opinions and take these into account in the further development of the TDM process.

All questions that were evaluated for this study are presented in Table 2. In some cases, a slightly different wording of the questions was necessary. In these cases, we added both questions and numbered them as a/b. For creating the questions on the prerequisites, we followed the

SAGAT guidelines presented in Section 4.3.1. The questionnaires were created using LimeSurvey (Limesurvey, 2025). They were completed electronically via smartphones or laptops, depending on each participant's preferences. All questionnaires were pre-tested with two senior developers who were not part of the research team. In addition, the questionnaires were all completed while the researchers were present, allowing any uncertainties regarding the wording of the questions to be discussed immediately.

Data analysis. During the evaluation, we recognized the complexity of asking for the prerequisites in three different scenarios and the hardship of comparing them with the results from other methods (also see Section 8.3). Therefore, we merged the separated results to reduce them to the ten TD-SAGAT prerequisites. For this, we used the results from the questions on TD incurrence (a) if the prerequisites were not TD-related, i.e., alternatives, benefits, drawbacks, risks, effort, and competing functional requirement (1a -5a and 8a of Table 2). We used the answers to the TD repayment questions (b) if the prerequisites were TD-related, i.e., principal, interest, affected component, and quality attributes. (6a, 7a, 9a, and 10a of Table 2). We created visualizations for the prerequisites' results and presented the progress in each workshop's "learning" phase and in this paper, comparing the questionnaire's answers to the results of the observations and TD-SAGAT.

We used the questions from the kickoff questionnaire regarding the current TDM process to understand and report the team's current processes in Section 6.1. We presented the results to the team in the second workshop. However, we did not analyze these questions scientifically, nor did we present the visualizations in this paper due to their limited significance compared to the textual description.

We utilized questions 12 and 13 (Table 2) from the progress questionnaire to answer RQ2. Questions 14 to 17 indicated the right time to terminate the action research. Furthermore, we compared these backlog-related questions with the actual amount of TD in the backlog. Finally, the open-ended questions (18 and 19) gave qualitative feedback on the TDM process and the project's progress. We used the answers to address emerging topics in the following workshop or separate meetings, e.g., special issues the algorithm subteam identified (see Section 6.3).

5. Workshops

In this section, we describe the five workshops in detail. Each workshop comprised the "evaluation", "learning", "diagnosing", and "action planning" phases of the respective action research cycle. The first workshop followed a unique structure as there was no "evaluation & learning" part but a need to get to know each other, analyze the team's current TD processes, and organize the upcoming work. Workshops two to five followed a similar structure, with a slight deviation for the second workshop, as two activities were introduced. The first and second workshops were done on-site, while the subsequent workshops were held online following the team's preferences.

5.1. Recurring intervention formats

We repeatedly employed two intervention formats typical for workshops: the "Love it – Change it – Leave it" and the "1-4-all" format.

In the retrospective "Love it – Change it – Leave it" format, participants are asked to write down what they like about the current process, what needs to be changed, and what should be abandoned. Each participant writes down each thought on one moderation card and presents their results one after the other by sticking them on a flipchart or whiteboard. The results of all participants are then clustered by the team. In the end, possible actions are derived, discussed, and potentially taken.

In the "1-4-all", questions given by the moderator are thought through briefly by each participant on their own (1). Then, the questions are discussed in detail in groups of four members each (4). Finally, each group presents its result to the other groups to find commonalities and discuss discrepancies until a common answer is formed (all).

5.2. First workshop/Kickoff workshop

We started the first workshop by motivating the TD topic. Throughout the entire first workshop, we collected topics relevant to the practitioners on an additional flip chart (i.e., a topic cache), which we then used at the end of the study to determine whether all topics were adequately addressed.

First, we informed the team about the details of the upcoming study. After that, we analyzed the team's motivation using an adaptation of Führing's "organizational energy template" (Führing, 2021) where each participant was asked to identify their opinion on the topic's importance versus their willingness to invest effort into that topic. The participants marked their self-assessment, their assessment of their team, and of their company, on a matrix with four quadrants on a flip chart (see Wiese and Serwa (2025)). We combined this with a short introduction of each participant, including their name, position, IT experience, time of company affiliation, and hobby. Additionally, we motivated the TD topic by playing a game developed to enable discussions on TD (Wiese et al., 2025).

After a lunch break, the participants completed the workshop (kickoff) questionnaire before the actual action cycles started. We diagnosed the current TD management process in three subsequent steps:

1. We collected concrete examples of TD, i.e., TD stories of TD incurrence/causes, TD consequences, and TD repayment, to get a first insight and to give the participants some room to present their experiences with TD. This also supported changing a potentially frustrated basic mood into a more constructive mood.
2. We identified the current approaches to managing TD by discussing the following questions in a "1-4-all" format.
    - What are you currently doing after you have identified a TD item? (e.g., document, communicate, repay, ignore)
    - How do you currently identify TD items?
    - In which situation do you consciously incur TD? On which information do you base this decision?
3. We gathered requirements for process improvement using a "Love it – Change it – Leave it" format

Eventually, we discussed the organization of the change project and action research. We identified meetings where TD items were typically discussed to observe these meetings for evaluation, and we named one team member to fulfill the role of a *TD manager*, similar to a "TD champion" mentioned in Jaspan and Green (2023). The tasks for the TD manager were:

- to be part of the action team and function as the link between the research team and practitioners,
- to implement the approved actions in cooperation with the researchers and potentially other teams, e.g., backlog administrators,
- to support other teams of the company in adopting and establishing a TDM process, and
- to be the "voice of reason" in meetings, continuously reminding everyone of the TD topic, i.e., to ensure the process would still be followed after the study ended.

We ended the workshop by reusing the organizational energy matrix to identify changes in motivation and allowing the participants to give feedback on the first workshop.

5.3. Second to fifth workshop

The second to fifth workshops were structured in similar ways. Each workshop started by reminding the participants of the workshop's goal and their role (action/reference team) in the change process and

Table 3
Overview of presented approaches with references and their adoption/use.

TD activity	Presented approaches	Used	Remark
TD organization	Establishing a TD champion (Jaspan and Green, 2023)	y	Called TD manager
TD identification & definition	Automatically by static analysis tools	(y)	Only as supporting method
	Manually by Dagstuhl definition (Avgeriou et al., 2016)	(y)	But other TD causes exist
	Manually by backlog colors (Kruchten et al., 2012a)	n	No external customers exist
	Manually by identifying whether TD interest occurs (Avgeriou et al., 2023)	y	
TD prevention	Well-defined requirements, code evaluation/standardization, following well-defined project processes, well-defined architecture/design, using good practices, improving tests, and reviews (Rios et al., 2020; Freire et al., 2020; Pérez et al., 2021)	(y)	The team generally optimized their work processes; not all strategies were followed
	Educational sessions on how to avoid TD (Besker et al., 2020)	n	Workshop worked as an educational session, but no other sessions were planned
	Incorporating a TDM approach into project management (Wiese et al., 2022)	n	
	Debiasing decision-making by listing at least one solution alternative, one drawback, and one risk per alternative (Borowa et al., 2021, 2022)	y	
	Better decision-making by considering quality attributes (ISO/IEC, 2023; Bass et al., 2012)	n	
	Counteract TD inheritance between disciplines in mechatronics (Dong et al., 2019; Vogel-Heuser and Bi, 2020)	n	
TD measurement	Interest, interest probability, effort/principal for prioritization (Tom et al., 2013; McConnell, 2008)	y	To identify *low-hanging fruits*
	Interest, interest probability, effort/principal for decision-making (Perera et al., 2024)	y	
TD documentation	Recording TD in a backlog (Kruchten et al., 2019; Junior and Travassos, 2022)	y	
TD prioritization	Prioritizing TD against functional requirements (Lenarduzzi et al., 2020)	n	
	Prioritizing TD against each other (Lenarduzzi et al., 2020)	y	
	Prioritizing once (Lenarduzzi et al., 2020)	n	
	Prioritizing constantly (Lenarduzzi et al., 2020)	y	By using a *re-submission date*
	Based on the systems evolution (Schmid, 2013a)	y	
	Based on the business value (Reboucas De Almeida et al., 2018)	n	
	Based on cost attributes (TD measurements) (Tom et al., 2013)	y	To identify *low-hanging fruits*
	Based on the affected quality attributes (Rachow and Riebisch, 2022)	n	
TD repayment	Repay TD if it impedes a functional requirement (Freire et al., 2023)	y	Similar to refactor if it impedes development
	Pay the interest (Buschmann, 2011)	y	
	Wait for discontinuing or migrating of the affected software system (Schmid, 2013a,b)	y	Use *re-submission date* to recall TD items
	Repay TD if the positive effect is proven by analyzing the costs (McConnell, 2008; Tom et al., 2013)	y	Repay *low-hanging fruits* if time allows
	Repay TD by assigning a quota (McConnell, 2008; Wiese et al., 2022)	n	
	Repay if TD might hinder the system's evolution in the near future (Schmid, 2013a)	y	
	Repay if TD was incurred due to a project deadline (Wiese et al., 2022)	n	
TD monitoring & visualization	DebtFlag (Holvitie and Leppänen, 2013), MultiDimEr (Silva et al., 2022), VisminerTD (Mendes et al., 2019), MIND (Falessi and Reichel, 2015), TEDMA (Fernández-Sánchez et al., 2017)	n	Not for backlog visualization, or only for self-developed tools
	Self-developed Microsoft PowerBI TD visualization (Section 6.6)	y	Interfaces to Azure DevOps and Jira exist

study. Then, the researchers presented the evaluation results of the previous action cycle ("evaluating"). During this presentation, the team discussed the findings if needed. Before lunch, the participants were asked for their evaluation of the actions taken during the previous cycle in a "Love it – Change it – Leave it" format ("learning"). After the break, the participants completed the workshop questionnaire. Then, the researchers introduced the new topics, namely TDM approaches, for the respective workshops in presentations that each lasted no more than 45 min. An overview of all workshop topics is given in Table 5, and the respective research approaches are presented in Table 3. The details of each workshop's topics are explained in the following "topics" subsections. After the introduction, we asked the participants to discuss the new topics and decide which approaches to adopt ("diagnosing" and "action planning") utilizing the "1-4-all" format. At the end of each day, we determined the topics that were raised during the day and the measurements that the team decided on. We assigned at least one responsible participant for each measurement.

While all workshops were planned this way, slight deviations regarding the breaks were possible. A generic agenda for these workshops is part of the additional material (Wiese and Serwa, 2025).

The third to fifth workshops were done online utilizing Microsoft Teams (Microsoft, 2025d) due to the participants' preferences. In these cases, we used breakout rooms for the "1-4-all" discussion in "groups of four" and Mural (Mural, 2025b) for the "Love it – Change it – Leave it" part, employing its "Retrospective Template" (Mural, 2025a).

5.3.1. Second workshop – Topics

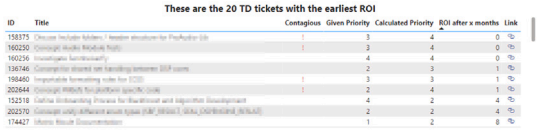
The second workshop marginally deviated from this procedure as two activities were introduced (TD prevention and TD measurement & documentation), and two discussions were conducted: We presented the works shown in Table 3 about TD in mechatronics (Dong et al., 2019; Vogel-Heuser and Bi, 2020), quality attributes (ISO/IEC, 2023; Bass et al., 2012) and cognitive biases (Zalewski et al., 2017; Borowa

Table 4
Overview of visualizations with underlying uses cases/questions, assignment to a dashboard (M - manager, T - team), final implementation details, the corresponding figure (add. mat. = additional material) and their adoption by the team.

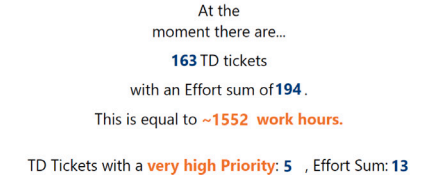
No.	Dash-board	Use case question to be answered	Chart type	Implemented visualization	Fig.	Adoption /Usage
V1	M + T	Which TD tickets can influence the development of a specific user story?	List	We list the TD tickets in the component (areaPath) and its sub-components where the user story will be implemented. (The original mock-up was not adopted but changed to a list.)	2(a)	Used
V2	M + T	How big is the current amount and effort of all TD tickets?	Text	We display the number of TD tickets and the sum of the effort as text.	2(c)	Rarely used
V3	M	Which TD item's repayment will be quickly amortized?	List	We list the 20 TD tickets with the lowest ROI in months. (The original mock-up was not adopted but changed to a list.)	2(b)	Usage planned
V4	M	How many costs can we save by repaying this TD item?	Line	We show the accumulating interest costs (calculation see Section 4.2.2) per TD item.	add. mat.	Not used
V5	M	How does the amount of TD develop over time?	Line	We display the effort sum of all TD tickets per month that were still open in the respective month.	2(d)	Rarely used
V6	M	Which TD tickets can only be repaid by person X?	–	No visualization: From the works council's point of view, it is critical to present the work performance of a single person.	–	Not used
V7	T	How accurately are the priorities estimated?	Bar	We show the *educated guess* and the calculated *ROI-based* priority (calculation see Section 4.2.2) as overlaid bars.	2(e)	Used
V8	T	How many TD tickets will be resubmitted for a given meeting?	Dot	We show the number of TD tickets per *re-submission date* versus the *educated guess* or the calculated priority (customizable).	add. mat.	Not used
V9	T	Which tickets are *low-hanging fruits* (i.e., high priority with low effort/principal)?	Dot	We show the number of TD tickets per effort versus the priority. The upper left and lower right regions are dyed green/red and marked with *Low-hanging fruits/Wait*.	2(f).	Used
V10	T	Which tickets are contagious and should be repaid soon?	List	We represent contagious TD tickets as a column containing a red exclamation mark in the detailed table views.	2(b)	Used



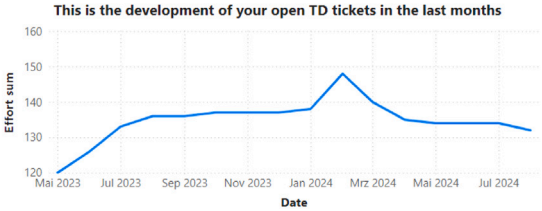
(a) V1: TD tickets (TD ID) that influence a given user story (US ID) with component or sub-component (areaPath) and the sub-component's level



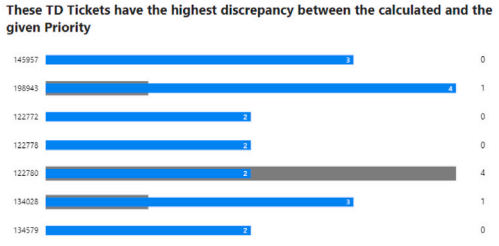
(b) V3: TD tickets (ID) ranked by the ROI in months, with *educated guess* (given) and *ROI-based* (calculated) priority and the information on contagiousness (V10)



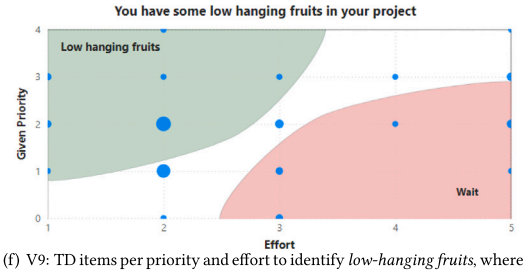
(c) V2: Current amount and effort over all TD tickets and for very high and very low priority tickets



(d) V5: Effort summarized for open TD items per month over time



(e) V7: Priority discrepancies between *educated guess* and *ROI-based* priority



(f) V9: TD items per priority and effort to identify *low-hanging fruits*, where the size of the dot represent the amount of TD tickets

Fig. 2. Visualizations created to manage TD items in the team's backlog and their adoption/usage by the team.

et al., 2021, 2025) and their impact on decision-making. Using the "1-4-all" format, the participants explored optimizations to the TD ticket by adding TD attribute fields to provide the respective vital information for decision-making of managers and developers. Further, we asked them to determine key performance indicators (KPIs) and ways to calculate them. For the (additional) discussion on quality attributes, each participant was asked to rank the attributes in a priority that would be typical for their system. Everyone presented their ranking, and the team was then asked to discuss and find a ranking that each participant was satisfied with. The resulting discussion about trade-offs and the hardship of deciding between quality attributes was on purpose and one of the learning goals. In the "1-4-all" discussion part of the second workshop, we asked the participants to explore optimizations to the TD ticket, i.e., the TD documentation, by adding TD attribute fields.

5.3.2. Third workshop – Topics

We informed the practitioners that TD could be prioritized against functional requirements or each other and that prioritization could be a one-time activity or iterative (Lenarduzzi et al., 2020). Additionally, we introduced various prioritization factors uncovered by academia (Schmid, 2013a; Reboucas De Almeida et al., 2018; Tom

et al., 2013; Rachow and Riebisch, 2022), including the comprehensive list from the SLR on TD prioritization by Lenarduzzi et al. (2020) (see Section 2.4). Using the "1-4-all" format, we asked the participants to discuss the prioritization factors and formulate additional ones if appropriate. The participants discussed which factors are most relevant to their work and which ones they can evaluate or estimate in a reasonable amount of time. Finally, we wanted to know the participants' opinions on whether a priority calculation might be feasible or if priorities should be set by the participants' *gut feelings*.

5.3.3. Fourth workshop - Topics

We introduced the repayment categories: ignore, refactor, and rewrite (Buschmann, 2011) and the methods explained in Section 2.4 and referenced in Table 3 (Yli-Huumo et al., 2016a; Freire et al., 2023; Buschmann, 2011; Schmid, 2013a; McConnell, 2008; Tom et al., 2013; Wiese et al., 2022). For the discussion part, we asked the participants to examine if they could uncover further repayment methods, and the participants discussed in which contexts they might use which approach.

5.3.4. Fifth workshop - Topics

In contrast to the other workshops, we did not use the related work on visualizing TD as inspiration, as they do not visualize backlog tickets but bugs, self-admitted TD, or static analysis metrics, or they visualize the tickets from a proprietary tool. However, we presented five TD visualizations found in research studies, as shown in Table 3.

Instead of using the visualizations from research, we developed a TD visualization based on Microsoft PowerBI (Microsoft, 2025c) that can easily be adapted to individual backlog needs. Moreover, PowerBI provides pre-implemented interfaces for Azure DevOps (Microsoft, 2025a) and Jira (Jira, 2025), which are commonly used backlog tools in the industry. For this purpose, we interviewed the action team, i.e., the two TD managers (P2, P6) of both subteams, the team manager (P1), and the person responsible for quality assurance (P4) before the fifth workshop. From these interviews, we gathered use cases for visualization and structured them into two categories, resulting in dashboards for the IT management and the development team. We found that the questions presented in Table Table 4 should be addressed through visualizations.

We introduced the concept of storytelling with visualizations to exemplify how data-driven storytelling might support decision-making (Gershon and Page, 2001; Kosara and Mackinlay, 2013; Lezcano Airaldi et al., 2021). Subsequently, we presented the visualization mock-ups designed by a researcher based on the previous interviews and their resulting use cases. We included one visualization idea based on calculating an ROI value as explained in Section 4.2.2. With this ROI approach, we intended to get feedback from the team on whether calculating a priority would be suitable, assuming the team would make the assumptions rather than the researchers.

In the discussion part, we questioned whether the presented mock-ups answered the questions comprehensibly and sufficiently. The team also discussed additional use cases and whether some questions or use cases and their corresponding visualizations were superfluous. Finally, the participants discussed in which situations, e.g., meetings, they might use visualizations. The discussion about and final decision on the adoption of those visualizations is presented in the result Section 6.7. In Fig. 2, we illustrate all visualizations that were evaluated as suitable after a four-week testing period.

5.4. Workshop material

More details on each workshop, including the researcher's presentation, the actions taken, and the learnings, are discussed in Section 6. For reproduction purposes, we have added all slides prepared for the workshops and the initial talk, as well as photographs of flip charts, to the additional material (Wiese and Serwa, 2025). They include further details on concrete evaluations, actions, measurements, and learnings, and can be exploited to replicate this study.

6. Action cycles

In the following, we introduce the action cycles, which provide details on the adoption of the approaches and the development of new approaches to manage TD. Each action cycle starts with a workshop day, which includes the "evaluation's presentation", "learning", "diagnosing", and "action planning" phases of the action research method, as shown in Fig. 1. The workshop day is followed by the "action taking" and "evaluation" phases between two workshops.

In this section, we briefly reference the *diagnosing* phase, i.e., the discussed topics during the workshops as explained in Section 5.3 for each cycle. Subsequently, we focus on the *actions taken* phase in terms of changes to the backlog and changes to the processes. Moreover, we present the *learnings & emerging issues* of each cycle, providing research insights that did not lead to actions. The workshop topics, actions taken, and learnings are outlined in Table 5. The table can be used to get an overview of all cycles and facilitates navigating to the most engaging topics.

6.1. Initial situation

The team started their TDM establishment without having followed a structured approach beforehand. According to the questionnaire, all participants agreed they did not manage TD; they had no approaches to avoiding, monitoring, prioritizing, or repaying TD in a structured manner.

In the kickoff discussion, the participants explained various situations where TD consequences hit them. For instance, the company opted to use an outdated CPU because waiting for the new version would have delayed the project by two additional months. Meanwhile, the demand in the market for the new feature was intense. Consequently, the team is now facing numerous performance and maintenance issues due to the choice of CPU. The team mentioned that they had set up SonarQube to identify code debt, but they had no process for dealing with the results. Still, most of the TD identification was done by the developers who were stumbling across TD-related issues, e.g., *"P2: . . . during implementation or when a complaint or a request actually comes in from some project. . . . I have a ticket, and I'm supposed to expand or implement something, and . . . it's working, but then I look again, [and] I'd have to incur TD to get it working, and I don't like doing that. And then I look at why, and then I might realize, yes, there is [already] TD somewhere"*.

In these cases, they either repaid the TD immediately or discussed it with colleagues and created tickets in their backlog, but they had no structured process for TD documentation.

P8 noted: *"That's a general question about how big the TD is. If it's a small problem that you can fix easily, then you would probably fix it yourself directly, but if it's really something bigger, then it should definitely be brought up for discussion"*. The manager mentioned: *"So I think we do discuss it and maybe even write a ticket, but we don't do it systematically. We don't have a specific type of ticket where we say what alternative there is, how much effort would it take to remove it and so on"*. Finally, P5 remarked: *". . . P2 has already done that somewhere. . . . We discussed this in the group, that this is somehow recorded separately [by each participant]"*. P3 and P6 complained about TD prioritization: *"P3: 'Prioritization is not always ideal, but'P6: 'I also find it really difficult when it comes to prioritization because I don't think we can prioritize a lot of TD on our own. So, a lot of TD that comes to us may or may not have an impact on others, and then you have to talk to other teams first"*.

Regarding conscious TD incurrence, the manager explained: *"so without judging it negatively now - so that we then have the pressure from other project teams from outside . . . we have to see: Ok, do we have the time to build it in properly and build it in properly together with them and do we have to take on this technical debt, because there's this time pressure from outside"*. Concerning the wishes for the project, the team particularly mentioned the importance of communication and better documentation

Table 5
Systematic overview of action cycles, including workshops, approaches, actions taken, and learnings.

No.	Date	Workshop topics	Approaches from TD activity	Actions taken - backlog	Actions taken - processes	Learnings & Emerging issues
Introduction Talk (1 h)	May 2, 2023	TD definition & TD types Causes & consequences Cost & benefit estimation Prioritization factors TD backlog & TAP framework	TD definition TD identification TD measurement TD prioritization TD repayment			
1	May 3, 2023	Study introduction Recap of the previous presentation TD game Previous TD instances Current TDM process Wishes and requirements for a TDM process TD manager & meetings	TD identification & definition (Multiple for game)	New ticket type Ticket consolidation	TD identification by customer visibility TD manager	TD vs. architecture trade-offs
2	August 24, 2023	TD in mechatronics Quality attributes Cognitive biases TD tickets attributes	TD prevention TD measurements & documentation	TD ticket attributes Talked about TD	Quality requirements' relevance	Backlog administration TD in the algorithm subteam
3	February 2, 2024	Prioritization vs. TD or functional requirements Prioritize one-time vs. recurring Prioritization factors Calculate a priority vs. gut feeling	TD prioritization	Re-submission date	Educated guess for priority Reorganize meetings	
4	April 9, 2024	TD repayment strategies	TD repayment	Benefits, drawbacks, risks	Pay interest Low-hanging fruits Repay before a project Repay after a project	"Central tendency bias"
5	June 11, 2024	Visualization from research Story-telling with visualization Visualization ideas based on interviews Visualization ideas based on ROI	TD visualization & monitoring	Valuable visualizations	Situations to use visualizations ROI-based priority Factors for the *educated guess* Worst-case scenarios Estimate effort on a continuous scale Knowledge silos.	Work processes are TDM's basis
6 Retrospective	August 8, 2024			Revising the template Relevant visualizations	Expectations for the future	Satisfaction with the backlog Simplyfied visualizations

The hyperlinks to the workshops', actions', and learnings' explanations in the text can be used to simplify navigating this paper's results.

of TD. P8: *"We had just discussed in our team that communication is always a very important point, and it's also very important here because you're often so deep in your framework and then discuss with your colleagues what really makes sense or are you actually already too deep in your mess . . . "* P2: *"I would also like detailed descriptions [of TD], I already mentioned that earlier, that you can really... that you can then see what the consequences are if you can estimate them at the time; that you give a bit of context around why this is technical debt at all"*. The topic cache on the flip chart stated similar topics, e.g., general TD decision-making (incl. a guideline for management decisions), the TD ticket quality, and continuous communication, e.g., with colleagues that function as their customers, enterprise architecture management, and other TDM stakeholders.

6.2. First cycle - TD identification & Current TDM process

The interventions started on May 2nd, 2023, with a one-hour talk to inform the team about the basic intricacies of TD. This talk included inspiration on TD from various studies (Cunningham, 1992; Avgeriou et al., 2016; Li et al., 2015; Wiese and Borowa, 2023; Wiese et al., 2022; Schmid, 2013a; Tom et al., 2013; Reboucas De Almeida et al., 2018; Rachow and Riebisch, 2022; Kruchten et al., 2012b) introduced in Section 2. The talk was directly followed by the first one-day workshop on May 3rd, 2023, which started the action research and served as the kick-off regarding the change management process. The agenda of the first workshop is explained in Section 5.2.

Diagnosing. In this workshop, we diagnosed the current TD management process in three subsequent discussions: We asked the participants to describe concrete TD instances, elaborate on how they react to TD items, and discuss their wishes and requirements for a TDM process as explained in Section 5.2.

Actions taken – Backlog. New ticket type. The team created a new ticket type for TD tickets in the backlog. This ticket type did not yet incorporate any TD-specific attributes. Initially, it was simply a means of identifying and tagging TD items in the backlog.

Ticket consolidation. The team identified all TD items in the backlog and created TD tickets for all TD items they had in mind but had not yet recorded. The advantage of this approach was that at least all TD items that had been hindering the team at one point, i.e., all effective TD items (Schmid, 2013a), were recorded. At this point, 127 tickets were in the backlog, of which 109 had the status "new", i.e., they were not yet refined. The action team identified 65 tickets as TD items.

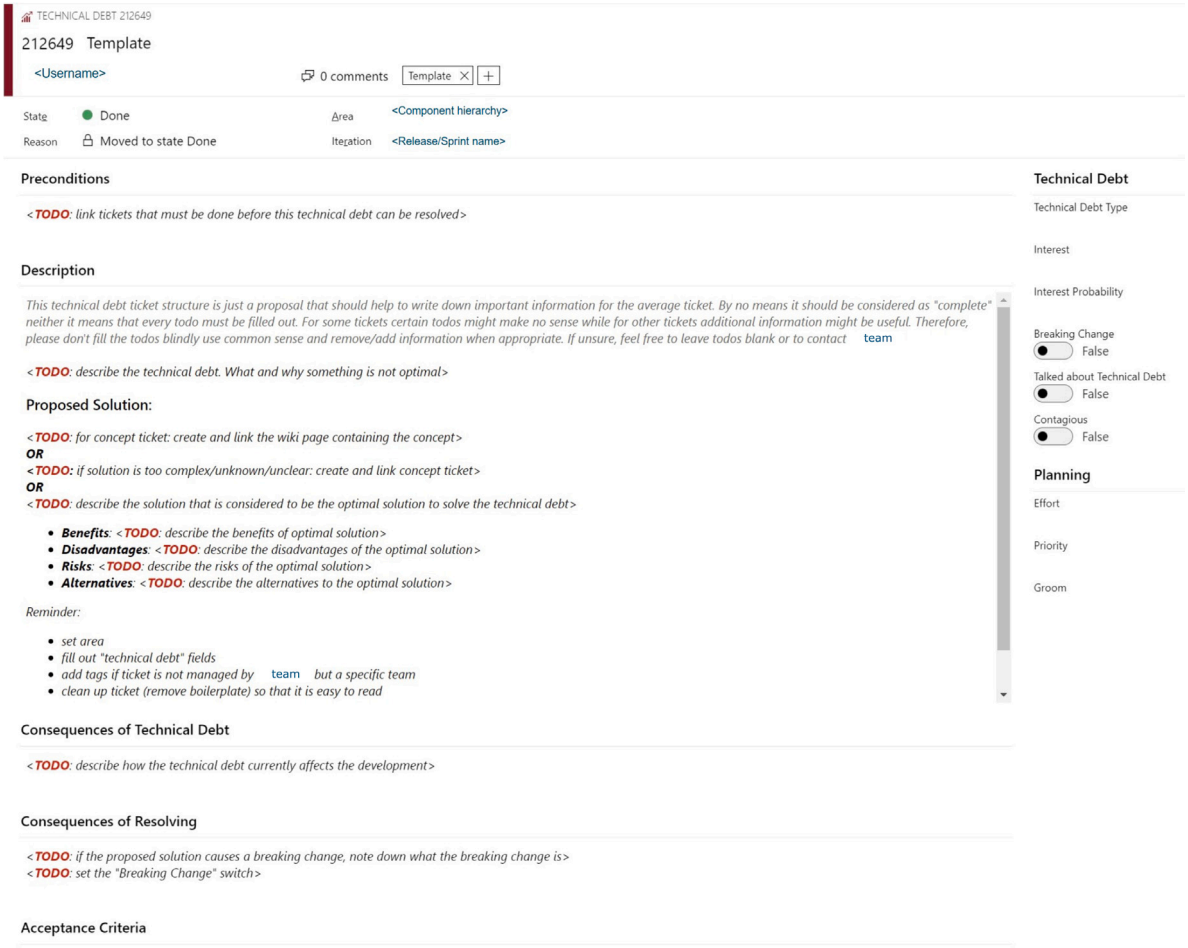


Fig. 3. Screenshot of the team's TD ticket template.

Actions taken – Processes. TD Manager. During the first workshop, we established a TD manager, as explained in Section 5.2.

TD identification by customer visibility. We tried to distinguish TD tickets from functional or bug tickets using the four colors of the backlog established by Kruchten et al. (2012a). However, the team created software for other IT teams within the company. Thus, the question arose whether an item is TD when it is not visible to the other teams or only if it is not visible to the end customer.

For example, the following discussion with their manager arose: *"P1: 'But if it's not working as expected right now, isn't that a bug, or am I misunderstanding something here?" P2: 'Well, it's a bug in their [the other team's] application then, but I mean, we have made life difficult for them; why aren't we doing that [avoiding the bug] already? We can do that implicitly when we set up our controller . . . ' P1: 'Ah, and they have to make a workaround somehow so that it works properly for them, right?'"* The team first decided to adopt the product owner's viewpoint and interpret the other teams as their customers. Yet during the study, the team embraced a simplified method by inspecting whether the TD items created interest for the team as described by Avgeriou et al. (2023). Furthermore, they considered whether negotiations with management are necessary for the implementation of this ticket.

Learnings & Emerging issues. TD vs. architecture trade-offs. Initially, the practitioners had problems distinguishing between TD and architecture trade-offs and identified all trade-offs as TD. We explained the difference, and the team decided to classify an issue as TD only if they had a more suitable solution alternative in mind.

6.3. Second cycle - TD documentation and prevention

The second workshop on August 24th, 2023, was on the topics of TD incurrence & prevention by optimizing TD decision-making and TD documentation to improve the TD ticket.

Diagnosing. In this workshop, we discussed methods to prevent TD (e.g., use of quality attributes, cognitive biases) and goals for TD documentation discussions as explained in Section 5.3.1.

Actions taken – Backlog. TD ticket's attributes. The TD ticket was enhanced by adding attributes for the TD item's interest, interest probability, contagiousness, whether the repayment results in a breaking change, the affected component, the consequences of the TD item, and the consequences of repaying it as explained in Section 4.2.2. The final version of the team's TD ticket can be seen in Fig. 3. This figure and an excerpt from the team's wiki describing these attributes are included in the additional material (Wiese and Serwa, 2025).

Talked about TD Checkbox. The practitioners determined that creating a TD ticket type is not sufficient. Instead, changing all ticket types and adding necessary information for decision-making is essential to avoid TD. As a first approach, they added a *Talked about TD* checkbox to all ticket types, i.e., user story tickets, bug tickets, features, epics, etc. This checkbox should remind the team to evaluate the risk of incurring TD when implementing the respective ticket.

Actions taken – Processes. Quality requirement's relevance. The workshop highlighted the relevance of quality attributes, especially for the algorithms subteam. For them, the quality requirements were essential but

generally not provided from the outset. Instead, they were assumed by single developers, which led to misunderstandings with the customers and, consequently, to TD in previous implementations. The team chose to discuss quality requirements with each other and the customers before starting the development process from now on.

Learnings & Emerging issues. Backlog administration. The team had to discuss changing the ticket design, e.g., adding the attributes, with the team responsible for the backlog software. Particularly, adding the *Talked about TD* checkbox to all ticket types took another two months of organization and negotiation. Our previous research showed that not adding another tool for TDM is important (Wiese and Borowa, 2023). Yet, adapting existing tools to TDM might be a problem that many companies face. If the whole company uses the backlog tool, there might be restrictions on its adaptation. Besides, adding value calculations or rule-based pre-filling of attributes might be challenging for proprietary tools.

TD in the algorithm subteam. The algorithms were developed in an unrestricted environment, often as part of research and development work, e.g., by experimenting and developing prototypes. Accordingly, the amount of effective TD in the algorithm team seemed very low. We decided to focus the research study on the framework subteam and use the algorithm team's insights as supplementary input. However, we also established a dedicated TD manager for this subteam.

6.4. Third cycle - TD measurement & Prioritization

The third workshop took place on February 14, 2024. We initially planned to focus this cycle on TD visualization (now the fifth cycle). However, prioritizing TD was already a frequent discussion topic in the previous cycles, which led us to adjust the initial plan according to the practitioners' needs.

Diagnosing. We informed the practitioners about TD prioritization factors and discussed the relevant factors and processes for the team as presented in Section 5.3.2.

Actions taken – Backlog. Re-submission date. The team recognized that tickets got lost in the backlog, i.e., some tickets, particularly TD tickets, have a low priority and stay at the end of the backlog list until they are forgotten. The team decided to add a *re-submission date* to all ticket types, which represents an iterative prioritization approach (Lenarduzzi et al., 2020). This date ensured no ticket was lost in the backlog and also indicated the ticket's urgency. Urgent tickets would be rescheduled soon, while less urgent tickets would be postponed for longer periods. This also had the advantage of separating priority from urgency, following the idea of the Eisenhower matrix (Baer, 2014; Wikipedia, 2025; Miro, 2025), creating a quadrant of urgency and importance to decide a task's future. The team really valued this approach as mentioned in one of the meetings: *"P4: 'One really notices that there's at least a bit of a flow now, right? It's getting better every time'. P3: 'And as a result, nothing is forgotten, which I think is nice because they [TD tickets] come up every now and then. P4: 'There's still room for improvement, but we're on the right track".'*

Actions taken – Processes. Educated guess for priority. In the workshop's discussion regarding TD prioritization, we asked the team whether they would like to calculate the priority based on the recorded attributes. The team, particularly the algorithm developers, who all had a mathematical background, did not think it was possible to calculate a priority that builds on all relevant attributes. Some attributes, e.g., the developers' availability and the requirements of the teams using their framework, change quickly or cannot be recorded. The team decided the priority should be an *educated guess*, i.e., an estimation based on knowledge of the underlying ticket's attributes. This should be sufficient and much better than the mere *gut feeling* they used before. Fittingly, the *resubmission date* was used for continuous re-evaluation of this guess.

Reorganize meetings. In the "learning" phase, it became apparent that many TD-related problems originated from poorly designed work processes, such as communication issues or the number and rationale of meetings. Consequently, another significant change was to reduce the three meetings (planning, grooming, and refinement) and separate the meetings of both subteams starting in March 2024. The framework subteam decided to have two meetings: refinement and planning. Initially, they had three half-hour refinement meetings each week to review the TD backlog and discuss each ticket and its attributes, including the new *re-submission date*. The algorithms subteam decided to have one meeting that combined refinement & planning. They also decided not to use the *re-submission date*, as their TD backlog was smaller and organized hierarchically.

6.5. Fourth cycle - TD repayment

In the fourth workshop, held on April 09, 2024, we informed the participants about the different methods of addressing TD.

Diagnosing. We introduced the repayment categories: ignore, refactor, and rewrite (Buschmann, 2011) and the respective approaches and discussed the adoption of these as explained in Section 5.3.3.

Actions taken – Backlog. Benefits, drawbacks, and risks. Our observations uncovered that the *Talked about TD* checkbox led the team to discuss alternative solutions. However, discussions about the benefits, drawbacks, and risks of the solutions were rare. In contrast, the TD-SAGAT evaluation showed that these aspects affected many team members' minds (see Fig. 7). As the discussion might help share all information across a team, they decided to add a reminder to each ticket type to discuss these attributes by creating a text template providing space to elaborate on each attribute (functionality provided by the backlog tool).

Actions taken – Processes. Pay interest. The participants decided to pay the interest if the priority and urgency were low and the effort was high. However, they did not want to close these tickets but to keep them open in the backlog with a far-off *re-submission date*.

Low-hanging fruits. The team planned to repay *low-hanging fruits*, i.e., TD tickets with a high priority and low effort, as bridging tickets, i.e., if a developer found some spare time to bridge between functional requirements.

Before a project. The participants decided to repay TD before a project, which was their most valued repayment method. In their backlog's ticket hierarchy, they considered a feature, epic, or user story ticket to be a project. They planned to use this method by reviewing the TD tickets related to the respective ticket's affected component.

After a project. The team intended to adopt the project-related TD repayment after a project. As the team does not utilize projects, they considered initiatives (see Section 4.2.2) as projects for this repayment method. This means the team planned to repay all TD tickets incurred during an initiative's implementation before it was closed. However, as initiatives were rarely used, they were not able to use this method before the study's end.

Learnings & Emerging issues. "Central tendency bias". We presented a calculated priority based on *interest risk* as explained in Section 4.2.2. Comparing this priority with the *educated guess* priority demonstrated that team members seldom gave very low or very high priorities. We attribute this to the "central tendency bias" of preferring medium values instead of extreme ones (Pimentel, 2019). We discussed this observation with the team members, who determined to use "very low" and "very high" more boldly and frequently. However, the tendency remained until the end of the study.

6.6. Fifth cycle - TD monitoring & Visualization

The fifth workshop took place on June 11, 2024, and was on the topic of visualization.

Diagnosing. We presented TD visualizations found in research studies as shown in Table 3 and visualization mock-ups, which we created for the use cases that emerged from the interviews. In the discussion part, we examined the suitability of the mock-ups as explained in Section 5.3.4.

Actions taken – Backlog. Valuable visualization. We presented mock-ups for all visualizations mentioned in Table 4 (visualizations see additional material Wiese and Serwa, 2025). The team decided that a button to switch between *educated guess* and *ROI-based* priorities should be included for all visualizations that are based on priority. Notably, they discarded V6 (TD tickets that must be repaid by a specific person) from the outset because a visualization for a single team member would not be compliant with the works council. Based on the outcomes of the discussion, a researcher further improved all visualizations to make them better understandable. The implementation was completed on July 23, 2024, and the visualizations have been used in team meetings since then to assess their suitability.

Actions taken – Processes. Situations to use visualization. The algorithm subteam decided not to use visualization as they had too few TD tickets (two at that time) for a visualization to be valuable. The framework subteam decided to use the visualization at the beginning of each refinement meeting. However, they changed their decision during the "action taking" to using the visualization in more specific situations, e.g., when looking for *low-hanging fruits* or TD items that affect a functional requirement.

ROI-based priority. As intended, the *ROI-based* priority was intensely discussed, and its limitations were laid out. While the team members initially were reluctant, they acknowledged during the discussion that a calculated value might have benefits, e.g., summarizing at least some relevant attributes. However, the underlying values, e.g., interest and interest probability, had to be chosen more carefully, keeping in mind their effects on the priority. They did not consider these effects during the previous refinements. Thus, they planned to re-evaluate the tickets with a significant difference between the *educated guess* and the *ROI-based* priority. They decided to use visualization V7, i.e., Fig. 2(e), for this purpose.

Analyzing the differences between *educated guess* and *ROI-based* priorities led to three additional learnings and changes in their processes to improve their *educated guesses*:

(1) Factors for the educated guess. While the calculation of the *ROI-based* priority included the effort of a ticket, the team members usually did not include the effort in their *educated guess*. They decided that including the effort was unnecessary as the visualization V9 of *low-hanging fruits* would give them the required result without changing their estimations. In contrast to that, the team included the information on contagiousness and breaking change in the *educated guess* while these factors were not included in the calculation of the ROI. Moreover, the team discussed that attributes such as interest and interest probabilities might change over time, e.g., due to a component being used more frequently. They included these expectations for the future in their *educated guess*.

(2) Worst-case scenarios. Initially, the team always estimated a worst-case scenario for interest probability, which might have led to higher values in the calculated priority. However, they did not specify whether the *educated guess* should be based on a median or worst-case scenario. We identified this issue by observing the team comparing the *educated guess* and *ROI-based priority* using visualization V7 (priority discrepancies). P2 and P3 discussed a TD item where the documentation should be updated, including an example code snippet: *"P2: Is that really painful? P3: . . . Yes, it's actually painful - especially when people are new to it, . . . it's not easy to understand: why we do it and why– P2: It can take a day if people don't understand it, can't it? P3: Yes, it's complicated. P2: And that happens a lot? P3: . . . That's the daily work when developing algorithms. . . . P2: Yes, but is that high or medium? Because once they've done that, it's no longer so painful for them. That's the worst-case scenario here. [pause] P2: Yes, we can set the high [previously medium] for my sake. I mean, at least then it's done. Otherwise, we'll just push this [TD item/backlog item] around".* In this case, the team reprioritized the TD item based on the worst-case scenario aligning it with the worst-case scenario used for the interest. Confronted with these observations, the team decided to align their estimations and (re-)evaluate all values using medium expectations.

(3) Estimate effort on a continuous scale. The participants realized that the effort scale, where an effort of five meant anything above five, was too coarse and unsuitable for a calculation. This problem was supported by the meeting's observations: *"P5: I've still added '5' to the effort, I don't know exactly how long it will actually take. But it will probably take a week or so. I don't know exactly what the '5' is somehow".* They decided to still interpret the numbers for the effort as person-days, but to allow any values, including values above five.

Knowledge silos. Comparing TD-SAGAT results with the observations, it became apparent that participants considered many prerequisites but did not talk about them. The participants explained that this might have been because only persons knowledgeable about a ticket completed the survey. They conceded that knowledgeable persons tend to expect other participants to know the same or consider knowledge sharing unimportant, which is known as the "curse of the knowledge bias" (Zalewski et al., 2017). Thus, they did not discuss prerequisites while still considering them. One striking example was the moderator of most of the meetings, i.e., the TD manager, who wrote down every piece of relevant information in the backlog item and, by this, learned details about every ticket and became a kind of a *knowledge silo*. In contrast, the non-moderating team members might not have put enough thought into tickets. Thus, the team decided to rotate the moderation of Scrum meetings to counteract the creation of such *knowledge silos*.

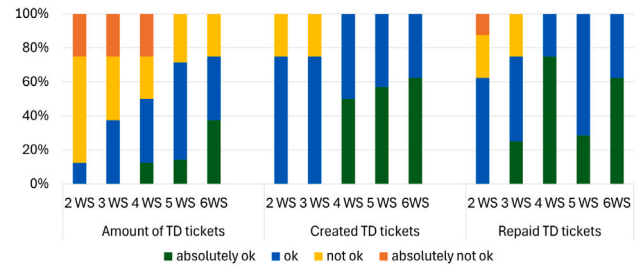
Learnings & Emerging issues. Work processes are TDM's basis. During all cycles, it became increasingly apparent that all work processes influenced the establishment of a TDM process, and, in turn, establishing TDM affected all work processes. On the downside, establishing a TDM process required much effort, which included identifying and discussing all work processes, e.g., to avoid TD, they improved the quality of all their backlog items by adding reminders to discuss alternatives, drawbacks, and risks. On a very positive note, however, the team improved all their work processes, not only TD-related process artifacts, e.g., they optimized their meeting structure and avoided *knowledge silos*.

6.7. Sixth cycle - Retrospective

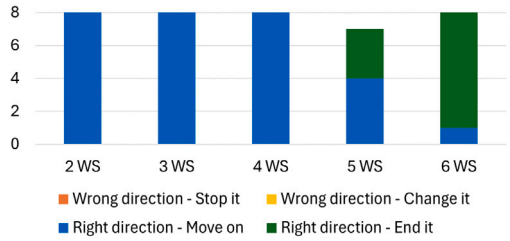
The first and last retrospective meeting took place on August 8, 2024. In this format, we did not present new ideas but focused on discussing the evaluations from the previous cycle.

Actions taken – Backlog. Revising the template. The reminder for "alternatives, benefits, drawbacks, and risks" was valuable for the discussion, but filling out the information in the ticket was deemed micro-management. The benefits of a ticket were usually obvious and part of the regular description. Evaluation of alternatives was already done when using the *Talked about TD* checkbox or as part of a concept ticket. Similarly, explaining the effort was deemed micro-management: *"P3: 'But [we're] getting into micromanagement now.' P2: 'Yes, that's the case now – I don't know – I wouldn't actually write such obvious things as effort. That it's an effort".'* Thus, the team decided to focus the discussion on drawbacks and risks as they provide essential information often underestimated or ignored, e.g., *"P2: The disadvantage is that we'll probably end up with dependencies. Because you would probably come to the conclusion that you want to take [tool name] command line arguments to do that".*

Relevant visualizations. After the team had used the improved visualization for four weeks, we discussed each visualization's suitability. The team identified that V1 (Fig. 2(a) – TD tickets influencing a user story) and V9 (Fig. 2(f) – Low Hanging Fruits) were the most valuable



(a) Number of participants satisfied with the TD items in the backlog



(b) Number of participants satisfied with the TDM process

Fig. 4. Participant's satisfaction with the study's results per workshop.

visualizations. However, the team had difficulties understanding the original V1 visualization (see additional material Wiese and Serwa, 2025) and, thus, decided to use a list of TD tickets for a given user story with a filter for the user story instead (see Fig. 2(a)). The original versions of V3 and V4 were similarly hard to understand (see additional material Wiese and Serwa, 2025). Again, the team decided to use a simplified visualization, i.e., a list of TD items whose repayment is highly profitable (see Fig. 2(b)). This visualization has not yet been used, but the manager plans to use it for argumentation in the future. V7 (Fig. 2(e) – Priority discrepancies between *educated guess* and *ROI-based priority*) was used extensively to optimize the *educated guess* but might not be used in the long term. V8 (TD items per priority and re-submission date) was not beneficial at all and was removed from the dashboard. V2 (Fig. 2(c) – Amount and effort over all TD items) and V5 (Fig. 5(a) – Effort of TD tickets over time) are used for orientation but not as a part of any TDM process steps.

Actions taken – Processes. Expectations for the future. We confronted the team with our thought that including future expectations in their *educated guess* contradicts the basic idea of the *re-submission date*. When using a *re-submission date*, it seems more reasonable to base all attributes on the current situation, set an appropriate *re-submission date*, and change attributes when the *re-submission date* is reached. We noticed this issue while watching the team compare the *educated guess* and *ROI-based priority* using visualization V7 (priority discrepancies). For example, the team discussed whether a component should be able to use more than two kernels, which is currently not possible. P2 and P3 rediscussed the priority: *"P2: In any case, I think we've prioritized it correctly. It needs to be high because we're going to need it soon. Yes? P3: Hmmm P2: That means we're not going to reprioritize it now"*. In this case, the team included the future expectations in their educated guess. In the same meeting, they also discussed the inclusion of a block size parameter in a component. In this case, a workaround exists that all developers are aware of. Again, P2 and P3 discussed this: *"P2: Yes, so it doesn't hurt at the moment, because people are used to setting it for every module. P3: Yes, yes, exactly. Otherwise, we usually did it . . . with a wrapper . . . P2: Well, yes, I would still say that it remains medium. I wouldn't downgrade it now or anything. P3: No, we still have the other priority with the groom date [= re-submission date] . . . "* In this case, the team used the *re-submission date* to account for the future expectations. The team regarded our thoughts as valuable, revised their previous decision, and included future expectations only through a shorter *re-submission* period.

Emerging issues/learning. Satisfaction with the backlog. While Figs. 5(a) and 5(b) show that the number of TD items increased over time, the satisfaction with the backlog improved (Fig. 4(a)). The participants realized that the improved satisfaction values and the success of the change process are not rooted in the amount of TD but in the ability to manage the backlog and distinguish relevant from irrelevant TD.

Simplyfied visualizations. We identified that visualizations, even though they were built on the basis of the participant's requirements, were sometimes hard to understand. In multiple situations, the participants preferred more straightforward visualizations, e.g., simple lists for V1 and V3 and a simple pictogram for the contagiousness of a ticket in the list presentation. It would be interesting to further analyze whether this is due to the stakeholders being technical stakeholders and whether business stakeholders might prefer other visualizations.

6.8. Terminating the action research

Fig. 4(b) shows that the participants wanted to end the action research after the fifth cycle, i.e., the sixth workshop in August 2024. At this point, the participants were content with the incurrence and repayment of TD (Fig. 4(a)). The two participants who were dissatisfied with the overall TD amount in the backlog reported that they saw no reason to change the process further but to utilize the process to reach an acceptable amount of TD in the long term. Furthermore, we revisited the topic cache from the kickoff workshop and examined whether relevant topics remained unresolved. The team concluded that all significant topics had been addressed.

7. Results

In the following subsections, we present the answers to our three RQs on the adopted and newly created approaches (Section 7.1), the effort establishing a TDM process takes (Section 7.2), and the effects the establishment has on TD awareness (Section 7.3).

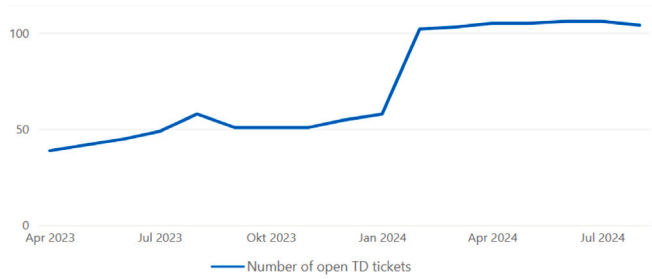
7.1. RQ1: Which TDM approaches are useful for the practitioners of our case company?

The action research determined the useful approaches, i.e., the approaches adopted in the "action taking" phases. We presented the details on and rationales for the adoption of the approaches in the previous Section 6. Table 3 gives an overview of all approaches and their adoption (usage) or rejection by the practitioners for each TD activity answering RQ1. Below, we summarize the useful approaches (Section 7.1.1) and provide an overview of the quantitative development of the team's backlog (Section 7.1.2).

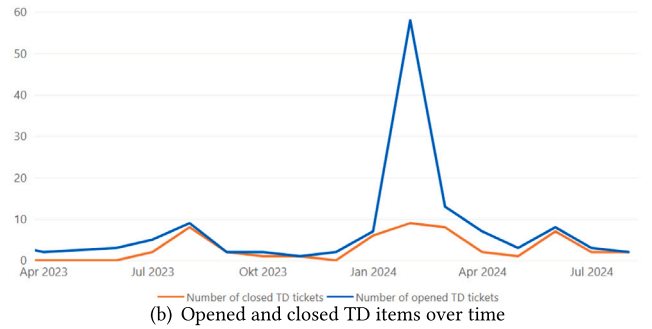
7.1.1. Adopted and developed approaches

TD identification. The team struggled to identify what items should be considered TD based on their visibility (Kruchten et al., 2012a). Also, the "Dagstuhl definition" (Avgeriou et al., 2016) did not entirely fit their needs. The most feasible definition was to identify an item's interest and specify it as TD if interest occurred (Avgeriou et al., 2023). In addition, they used the question of whether this ticket's implementation needs to be negotiated with the management as guidance. While the team had set up a SonarQube installation, they only used this as a supplementary tool for TD identification.

TD documentation/TD backlog (qualitative). For TD documentation, the team used its existing backlog. TD items were recorded as tickets using a newly created ticket type for TD as proposed by Kruchten et al. (2012a) and Rubin (2012). We found that improving the templates for



(a) Open TD items over time



(b) Opened and closed TD items over time

Fig. 5. Backlog development.

backlog items and adding new ones is a favorable way to establish a TDM process, as the team already organized all its work around these backlog items. Thus, this approach achieved a high level of acceptance among all team members. Additionally, the backlog items can be used to enable communication with business stakeholders by providing information necessary for their decision-making and subsequently visualizing the items in an aggregated way.

A new approach was using a *re-submission date* to iteratively adapt the attributes to new circumstances. By this, the team ensured that no items got lost in the backlog. They could safely set *re-submission dates* far away, e.g., when using the repayment method "waiting" when they expected a potential TD item to sort itself out over time.

TD prevention. Regarding TD prevention, the team's first action was to record and refine tickets before starting their implementation, i.e., they were "following well-defined project processes" as mentioned by the InsighTD project (Freire et al., 2020; Pérez et al., 2021; Rios et al., 2020). Another new approach was using a checkbox they named *Talked about TD* in every ticket type to remind them of the danger of incurring TD. Additionally, the team added a text template to all ticket types to record alternatives, benefits, drawbacks, and risks, as presented in the work of Borowa et al. (2022).

TD prioritization. The most relevant prioritization approaches were the evolution-based prioritization (Schmid, 2013a) and a cost-based prioritization (Tom et al., 2013; McConnell, 2008) to identify *low-hanging fruits*. These considerations led to a priority based on an *educated guess* rather than pure *gut feeling*.

The new approach of calculating an ROI for priority based on the underlying attributes was met with resistance from the participants. They conceded that using an *ROI-based* priority might support the priority estimation if the underlying factors are set carefully and with their effect on the ROI in mind. However, they deemed the ROI insufficient to determine the overall priority, as information on contagiousness, risks (breaking change), and developer availability are not included in the ROI but are essential factors for priority estimation. Finally, the participants used the *ROI-based* priority to analyze and improve their *educated guesses* in the fifth cycle. Notably, the participant did not adopt the idea of prioritizing TD against functional requirements (Lenarduzzi et al., 2020).

TD repayment. The chosen TD repayment approaches were strongly related to the prioritization approaches. The team identified TD items that might impede a functional requirement by examining TD tickets in the same component. This approach equals a mix of ignoring tickets that do not impede functional requirements (Freire et al., 2023) and evolution-related repayment before an initiative, feature, or user story's implementation (Schmid, 2013a). An initiative would be their equivalent of a project, but they rarely used initiatives for their daily work. Thus, repaying TD incurred by a project after that project's deadline (Wiese et al., 2022) was not adopted. Additionally, they repaid TD tickets they deemed as *low-hanging fruits*, which equals the cost-based approach (Tom et al., 2013; McConnell, 2008). Finally, they used the *re-submission date* to wait for a planned rewrite or a system's discontinuation without losing information about the TD item. Remarkably, the team did not adopt the idea of a repayment quota (McConnell, 2008; Wiese et al., 2022).

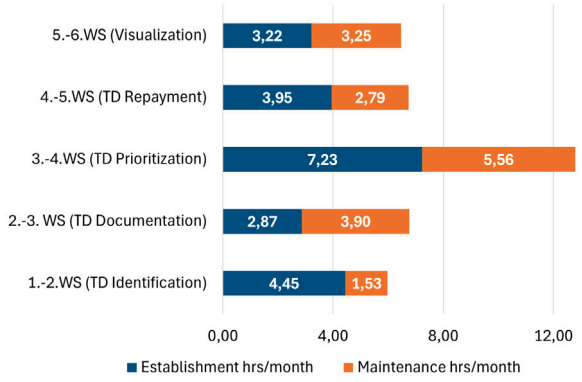
TD monitoring/visualization. The most relevant visualizations were the ones that support the chosen repayment methods, i.e., the *low-hanging fruits* (V9, Fig. 2(f)) and the TD items influencing a user story (V1, Fig. 2(a)). The visualization presenting the difference between *educated guess* and *ROI-based* priority (V7) was used to fine-tune the *educated guess*. The *ROI-based* visualization (V3) was deemed the most relevant from an IT management perspective, but it still has to prove its practical usability as, unfortunately, no situations for its use arose during the study.

TD manager. The researchers proposed a new work role for a *TD manager*, which was essential for the study to have a contact person within the team. More importantly, this person's task was to continue the process sustainably after its establishment. This is similar to the idea of a "TD champion" (Jaspan and Green, 2023).

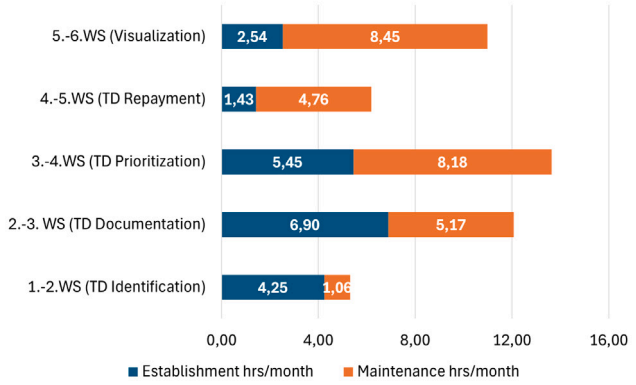
7.1.2. TD backlog (quantitative)

After the consolidation of the backlog, the backlog comprised 109 backlog items, of which 65 were identified as TD items. During further work with the backlog, the team deleted (not closed!) 50 items that were duplicates or related to technology they no longer supported in November 2023. This led to a reduced total of 40 instead of 65 TD items at the beginning of our study. Fig. 5(a) shows the amount of TD tickets over time and Fig. 5(b) the number of opened and closed TD tickets. The spike in the number of opened tickets in February 2024 was due to the refactoring initiative at that time. The team recorded 61 small refactorings as TD tickets. However, they were only able to repay a few of them in the given time. At the end of our study, this led to an increased total of 102 open TD tickets. However, apart from the spike from the refactoring initiative, the number of opened and closed TD tickets is similar. This means that new TD tickets were created, but nearly the same number of TD tickets were continuously repaid.

> **RQ1.1:** Using a backlog for TDM, particularly in combination with a *re-submission date*, is immensely valuable not only for monitoring but also for preventing TD. The approach by Schmid (2013a) to prioritize and repay TD based on the affected system's evolution seems to be the most intuitive for practitioners. A TD manager, i.e., a TD champion (Jaspan and Green, 2023), plays an essential role in maintaining the TDM process. To answer RQ 1.1. Table 3 presents an complete overview.



(a) Mean time investment (hrs.) over all participants



(b) Time investment (hrs.) of the TD manager

Fig. 6. Participant's effort for establishment and maintenance of the TDM process (excl. TD repayments during this time) (WS – workshop).

RQ1.2: Using a *Talked about TD* checkbox and a *resubmission date* in the backlog were approaches newly developed and employed by the participants. Using a *low-hanging-fruits* visualization and employing an *educated guess* for prioritization were new additions to their TDM process. Using an ROI calculation (*ROI-based* prioritization) was a new approach but only adopted partly, i.e., to optimize the team's process but not as part of the process.

Supplementary finding: While the amount of TD items in the backlog did increase during the study, the participants were more content with the backlog at the end of the study. Presumably, this shows that the ability to manage TD items is more relevant to practitioners than actually paying them back.

7.2. RQ2: How much effort are the practitioners of our case company willing to invest in a TDM process?

In the questionnaires, we asked the participants to estimate their effort invested in the initial establishment and the continuous maintenance of a TDM process per cycle. Notably, the requested effort does not include the effort to repay TD but just the overhead that a process and the establishment of this process creates. We broke the provided estimations down per month for each cycle, and the results are shown in Fig. 6. Across all cycles, the participants invested 5.1 h per month for the establishment and 4.67 h per month to maintain the TDM process. This is less than 3% of the working hours for each, assuming an 8-h workday and 22 workdays per month.

We found that TD prioritization required the most effort, which is understandable, as all tickets were reviewed to record the cost and prioritization attributes. Similarly, the visualization cycle consumed more time because the *educated guess* and the *ROI-based* priority were compared. We assume that some participants evaluated the effort for the review as maintenance as it took place during the regular refinement meetings, explaining the higher maintenance share in this cycle.

Notably, the invested effort highly depends on the participant. In our case, the algorithm developers did not invest much time, while, e.g., the *TD manager* invested a high amount of time for obvious reasons. In the end, the TD manager needed less than 5% of their time, which, nonetheless, should be taken into account when assigning other work packages to the TD manager.

RQ2: In the mean, the participants invested less than 3% of their time in establishing and maintaining the process, respectively. The TD manager continuously invested less than 5% of their time in maintenance.

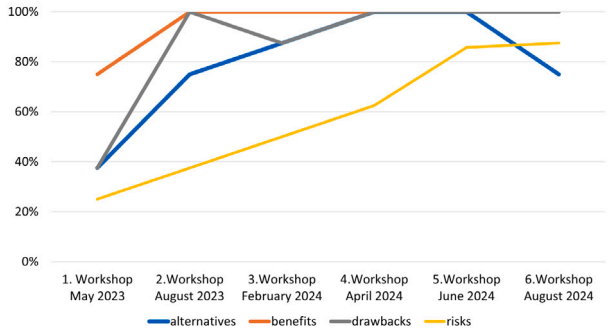
7.3. RQ3: Is the team members' awareness of TD in decision-making situations improved sustainably by the TDM process' establishment?

This section presents the evaluation results we collected to assess the participants' TD awareness. For this purpose, we analyzed whether the participants considered relevant information (prerequisites) when making decisions (Section 7.3.1), and the unconscious incurrence and correct identification of TD (Section 7.3.2).

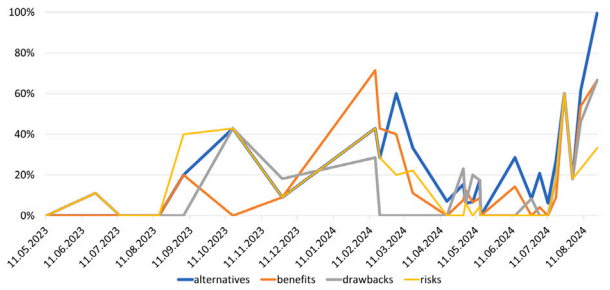
7.3.1. Requirements for decision-making

We evaluated whether the participants assessed the ten prerequisites (identified by the SAGAT approach) when making decisions, utilizing three methods, i.e., the self-assessment surveys Section 4.3.3, the observations Section 4.3.2, and TD-SAGAT Section 4.3.1. The ten prerequisites are organized into three categories (*Comparing alternatives*, *Evaluating costs*, and *Considering potential consequences*) to simplify their analysis as explained in Section 4.3.1. The results for each category are depicted in Figs. 7 to 9 and separated into charts for each method.

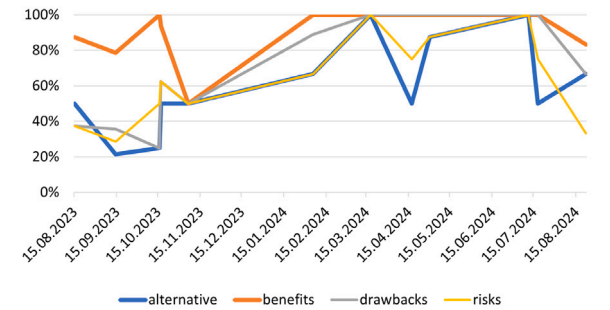
Comparing alternatives. Fig. 7 shows that the awareness for comparing alternatives, including considering benefits, drawbacks, and risks, rose after the second workshop on TD incurrence (including the debiasing techniques) in August 2023. This positive development is particularly evident in the results of the self-assessment survey in Fig. 7(a). The team discussed these prerequisites, even though most of the time, they still deemed their initial idea to be the better alternative: *"An alternative to [framework name] would be, maybe you can do it natively, but that's a lot of implementation work"*. Figs. 7(b) and 7(c) show that the awareness rose even further after the introduction of the *Talked about TD* checkbox at the beginning of October 2023. The *Talked about TD* checkbox also introduced discussions on the risks of incurring TD: *"P2: 'Do you also have to think about TD here? Because that could entail organizational TD.' P1: 'That's actually true, yes. Do you have any ideas?' P3: 'Well, if we have to simulate it and there's no one available, then we have a problem. Then, it will take its toll. Then we have to pay interest".'* Fig. 7(b) shows that after a drop in November 2023, the observation values further increased between January and March 2024. We assume that the drop was due to a focus on a competing functional requirement (see Fig. 9(b)). The increase was presumably due to the implementation of the TD ticket type and the review of all TD tickets to record all attributes for documentation and prioritization. In parallel, the team worked on a refactoring initiative, increasing the number of new tickets as visible in Fig. 5(b). This figure also shows that new tickets were rarely created from April 2024 onward. Consequently, Fig. 7(b) shows fewer discussions on alternatives for this time as primarily old tickets



(a) Questionnaire: % of participants stating to having considered prerequisite during the last action cycle

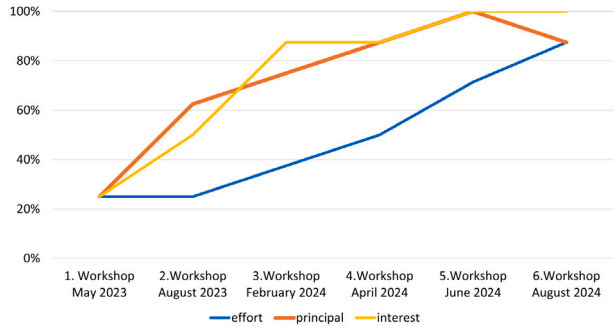


(b) Observations: % of ticket where prerequisites were mentioned

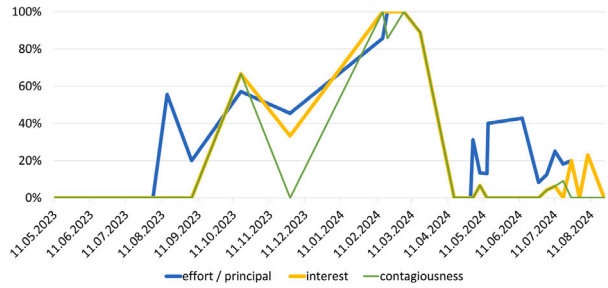


(c) TD-SAGAT: % of participants stating to having considered prerequisites during the previous ticket's discussion

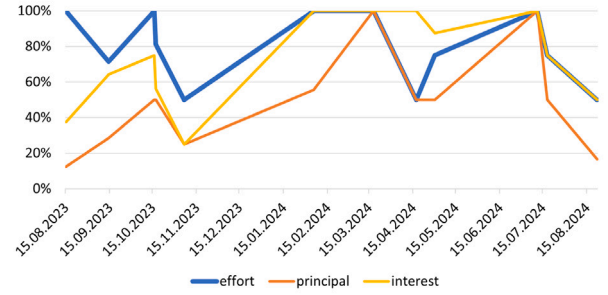
Fig. 7. Considering prerequisites regarding comparing alternatives.



(a) Questionnaire: % of participants stating to having considered prerequisite during the last action cycle



(b) Observations: % of ticket where prerequisites were mentioned



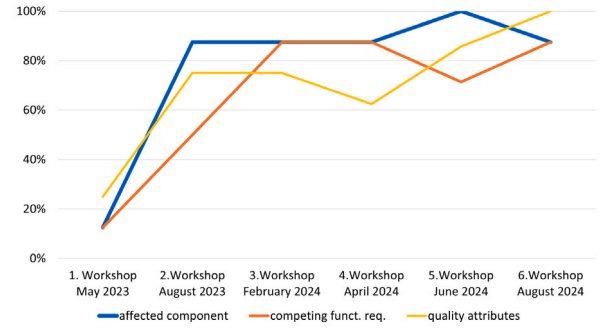
(c) TD-SAGAT: % of participants stating to having considered prerequisites during the previous ticket's discussion

Fig. 8. Considering prerequisites regarding evaluating costs.

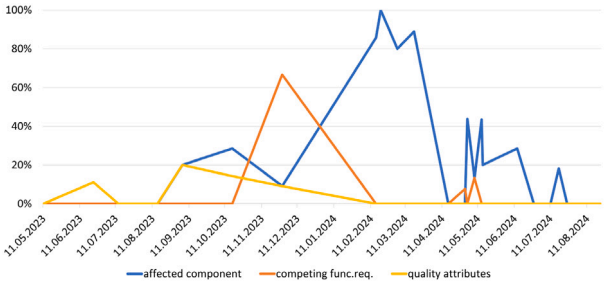
that were already refined were re-discussed, or only the *re-submission date* was postponed. From July 2024, the values in Fig. 7(b) began to rise again after introducing a reminder to discuss and document alternatives, benefits, drawbacks, and risks to each ticket. This led to a re-discussion and documentation of this information for old tickets. This positive development can similarly be witnessed in Fig. 7(c), i.e., the TD-SAGAT results for the situation-specific assessments.

Evaluating costs. A positive trend regarding the assessment of costs, i.e., effort, principal, and interest, is evident in Fig. 8(a). The values increased continuously over the course of the study, i.e., more and more participants declared they would consider these prerequisites. A similar direction of the values can also be seen in the TD-SAGAT assessment in Fig. 8(c). However, there was a notable decrease at the end of the study. We regularly observed discussions on the interest: *"P2: 'Interest is, you sayP̌3: 'Disturbing, it's annoying because you have to send everything back and forth to [name of function], and then there are restrictions again and so on".'* Particularly, the discussion on interest probability often led to astounding results: *"P2: Interest probability is kind of the exact opposite for me now because the average case is that people don't look into it [documentation] . . . Nobody misses it, and probably nobody needs it".* Fig. 8(b) shows that the trend regarding observations was less linear. For Fig. 8(b), we were able to combine effort and principal in one value because the effort for repaying a TD item is the principal, and we had the information on the ticket type for each observation. Moreover, we observed and marked discussions on a TD item's contagiousness. We included it as a prerequisite of the costs, as it illustrates the increase in principal over time. Unfortunately, this value was not covered in the surveys. The progression of the observation graph can be explained in the same way as Fig. 7(b). The significant increase between January and March 2024 probably resulted from reviewing the backlog and recording all attributes in the new ticket type. The drop after that was most likely due to the lack of new tickets, meaning that no discussions about tickets took place. During this time, only the effort was readjusted due to the initiative to estimate the effort on a bigger scale.

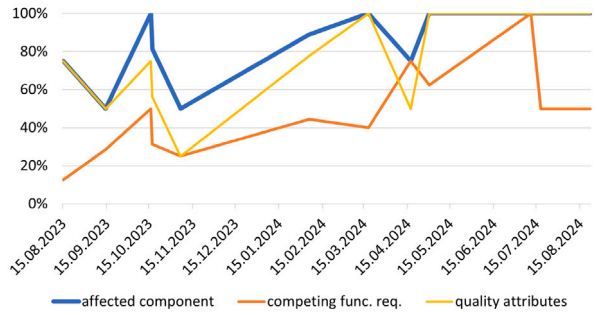
Considering potential consequences. Considering TD consequences included considering the component affected by a TD item, the potentially competing functional requirements when prioritizing a TD repayment, and the TD item's effects on quality attributes. As presented in Fig. 9(a), the self-assessed awareness for those prerequisites increased mainly at the beginning of the study, i.e., most participants declared to consider these factors since the second workshop. A similar development can be observed in Fig. 9(c) for TD-SAGAT, even if fluctuations occur. Small fluctuations might be attributed to the TD-SAGAT



(a) Questionnaire: % of participants stating to having considered prerequisite during the last action cycle



(b) Observations: % of ticket where prerequisites were mentioned



(c) TD-SAGAT: % of participants stating to having considered prerequisites during the previous ticket's discussion

Fig. 9. Considering prerequisites regarding TD's potential consequences.

method, which provides snapshots of the situation instead of a constant long-term development. Fig. 9(b) shows that quality attributes, despite their relevance, are rarely discussed. The team often discussed this issue and had plans to address it, as mentioned by P2 in February 2024: *"P1 and I actually wanted to ask all the stakeholders what they expect from our team, because we don't actually know exactly who our stakeholders are and what their expectations are, and the idea was to derive quality attributes from this"*. Nevertheless, the team did not manage to alter their behavior regarding quality attributes during this study. Fig. 9(b) shows a peak for considering the affected component at the beginning of 2024. From this category, the affected component is the only attribute recorded in the TD tickets. Thus, this peak and the slight decrease after it resulted most probably from the reviewing and documentation similar to Figs. 7(b) and 8(b).

Notably, in November 2023, the team focused on a functional ticket, so the value for competing functional requirements increased in Fig. 9(b). At this time, a decrease can be noticed in all other values, including the comparison of alternatives (Fig. 7), the assessment of costs (Fig. 8), and TD's consequences (Fig. 9). Together, this leads us to the assumption that the focus on one functional ticket at this time might have influenced the whole TDM process negatively.

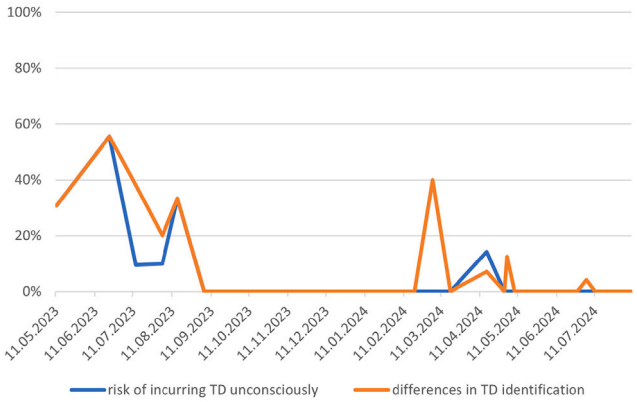


Fig. 10. Percentage of TD items per meeting, where different assessments on TD identification between researchers and team occurred, and where the team runs the risk of incurring TD unconsciously.

7.3.2. Unconscious TD incurrence and TD identification

In addition to the requirements, we observed whether there were differences between our and the practitioners' judgments as to whether a ticket could be a TD ticket and whether a functional ticket carried the risk of an unconscious TD incurrence (see Section 4.3.2). In Fig. 10, we can see many assessment differences at the beginning of the study, which we explained in Section 6.2 about the action cycles. Again, in the phase of TD prioritization between February and May 2024, there are some assessment controversies.

Situations with a potential risk of unconscious incurrences are reduced after the second workshop (in August 2023), which focused on TD prevention. The knowledge that TD can be incurred consciously or unconsciously led the team to classify their own TD items: *"This is also TD that we didn't consciously incur . . . We have a timer class, but there are other places where this timer class is not used, but [instead] some other code, which is why it reduces maintainability"*. Further, they were able to realize and discuss conscious TD incurrence: *"But that could be discussed. So I think it would be legitimate to say that we're taking on a lot of technical debt now because she [the customer] doesn't have that much time. And then later, when we want to use it again, we'll have to make sure that it's straightened out"*.

RQ3: The self-assessment shows an increase in awareness of all prerequisites. TD-SAGAT supports these increases overall but shows fluctuations typical for this method. The observations show that, in particular, the reminders and documentation of attributes in the backlog template sustainably led to discussions on respective prerequisites. We were able to identify how a focus on one important functional requirement was able to affect the whole process negatively. During the study, the team learned to identify TD more reliably, and they reduced the risk of unconsciously incurring TD.

8. Discussion

In this section, we discuss the TDM establishment, including recommendations for practitioners in Section 8.1. In Section 8.2, we discuss the effects the workshops had on TD awareness. Finally, we examine the overall study design to enable researchers to replicate and improve the study (Section 8.3).

8.1. Workshop approach for TDM establishment

During the TDM establishment, we made some notable observations, which we discuss below, including ideas on the process's improvement. Additionally, based on our observations and analyses, we present a list of actionable recommendations for practitioners who plan to establish a TDM process.

8.1.1. Notable observations during the TDM establishment

First, the backlog tool already used by a team seemed to be the natural tool to manage TD. However, two issues occurred during this tool's adoption: These backlog tools are commonly administrated company-wide, which makes changes for single teams more unlikely or negotiating them tedious. As TDM is an issue in multiple teams, this can be remedied by explaining the goal and providing added value for multiple teams. Regardless, we advise calculating the additional time this might take when establishing a TDM process. As a second issue, adding value calculations or rule-based pre-filling of attributes might be challenging in a proprietary tool. We assume these technical issues might be one reason why practitioners prematurely abandon TDM establishment approaches. A solution would be for the backlog tool vendors to integrate TD ticket types by default.

Second, while the participants used the *ROI-based* priority to analyze and improve their *educated guesses*, we observed that this led them to change the attributes in various cases. Still, they rarely reevaluated their *educated guess*. This can be caused by an anchoring bias, i.e., the preference for the first given value (Borowa et al., 2022). Additionally, the participants remarked that they initially did not evaluate the underlying attributes with the ROI in mind. For future replication, we suggest determining the ROI from the beginning (i.e., during the second or third cycle) and proposing it as an additional prioritization factor.

Third, apart from the mentioned "anchoring bias", we identified further biases, such as the "central tendency bias", and "curse of the knowledge bias", that influenced the TDM process at salient points. We recommend researching this topic more profoundly and developing methods to mitigate the effects of those biases in software engineering.

Fourth, the TD manager in this team was the team architect, which supported this participant's role as a *knowledge silo*. While a senior developer should perform this role, we suggest not assigning the most important developer or team architect but distributing such roles within the team.

Fifth, workshops three to five were performed online due to the teams' preferences, and we did not detect any drawbacks from this context. Regardless, we advise performing the first two or at least the first workshop on-site to get to know each other, build trust between the participants, including workshop supervisors, and sufficiently motivate the participants.

Finally, we consider two instances of not-adopted approaches remarkable: This team did not adopt the idea of prioritizing TD against functional requirements, which might be because they were under little pressure to process functional requirements. Additionally, they did not adopt the idea of a repayment quota, which might be because the team was comparably free to decide which tickets to process. However, further research on the reasons for this lack of adoption might be appropriate. It is intriguing to further analyze whether other teams in other contexts might be more willing to adopt those approaches and which contextual factors might lead to adopting which approaches.

8.1.2. Recommendations for practitioners

Our proposed workshop approach is flexible and does not provide a strict process to follow. Instead, it offers a way to determine a process that fits the respective team or company. Thus, we assume that the workshop approach, as well as the establishment of the TDM processes, might be adopted by multiple other companies. The approach is mainly fitted for teams that are organized in an agile way and are using some form of work items, e.g., user stories or task lists. Furthermore, it might be beneficial if some adaptable backlog tool is used. However, this process would similarly work for teams that use an Excel sheet as a backlog, e.g., by adding new columns per item. Analyzing whether this workshop approach would also work in less agile IT departments provides a fascinating area of future research.

To enable the adoption of this approach, we have collected some practical suggestions for practitioners when employing the workshops or establishing a TDM process in another way based on our experiences during the action research. For easy access, we provide the actionable recommendations in the form of a list:

- Do not try to implement the TDM process with all its intricacies from the beginning, but take the time to improve the process continuously.
- Start with identifying and tagging all the TD items in the backlog, then create a separate item type and successively add relevant attributes.
- Focus on the TD items identified by the developers (instead of putting too much focus on automatic detection) as those TD items are the effective ones, i.e., they have at least at some point already incurred interest.
- Motivate the developers to document TD items in the backlog that are on their minds but not yet documented.
- To distinguish TD from other ticket types: (1) Define your customer, e.g., would you define colleagues from other teams who use your system as customers? If they have requests regarding your system, would you determine this to be TD or a functional requirement? (2) Ask yourself whether interest occurs. (3) Ask yourself who would be willing to pay for the implementation.
- When your backlog contains too many items to identify all TD items manually: (1) Search for specific phrases like "refactoring", "quality", "test", "document", etc. (2) Start by focusing on the most recent items, e.g., items created in the previous year.
- Use the tools you are familiar with, e.g., your current backlog tool. (1) Include tool/backlog admins in the project early on and discuss adding a specific TD item template. (2) Analyze whether the backlog tool is flexible enough to allow for calculations (and not only rules).
- Install a TD manager, i.e., someone who feels responsible for the process. (1) Assign an experienced and respected developer who is willing to invest time. (2) Do not assign a team member as TD manager who already has many responsibilities.
- Plan the additional time to set up and maintain the process in your project/portfolio management, particularly including the time for the TD manager.
- When discussing new attributes, include discussing the selectable items of combo boxes, e.g., low, medium, and high for interest or interest probability, and their respective meanings and usage.
- If you plan to employ an ROI calculation: (1) Consider this when choosing the selectable items for the effort, interest, and interest probability. (2) Consider this when assigning those values to a TD item.
- Use reminders in the ticket templates to maintain awareness for certain topics (checkboxes are sufficient for this purpose).
- Use of a *re-submission date* for TD items. Make sure to define an appropriate process, i.e., define in which meeting a ticket that reached the date will be re-evaluated.
- When using an *educated guess* for priority estimation: (1) Discuss the various priority factors and decide which factors should be included with the team, e.g., whether to include the effort or not. (2) Determine in advance if you use worst-case scenarios or medium-case scenarios when evaluating priorities. (3) Make sure not to include a factor multiple times, e.g., do not include the future expectation in the priority if you have a *re-submission date*.
- Determine your system's architectural components. If you plan to use the process throughout the company, define a component hierarchy that fits into the company's hierarchy of products, systems, and components to be able to aggregate the results for the management.

- Discuss with the team what the term "quality" implies in terms of your system. Employing the task on quality attributes from workshop two might help your team to get to know quality attributes and experience the resulting hardships of deciding on trade-offs.
- Document the TD ticket's purpose and attributes, and explain them to persons outside the team who add (TD) tickets to your backlog.
- Structurally refining TD items will presumably lead the developers to acknowledge that not all TD items must be repaid. Hence, discuss suitable repayment methods after having some TD tickets prioritized with the team.
- Use visualizations if the number of TD tickets is too big to analyze otherwise.

During our study, we identified that issues in the general work process might incur TD. Thus, we recommend discussing the following questions regarding your meetings and backlog item's states:

- Which purpose has each meeting, and how does it differ from other meetings?
- In which meeting will you change which attribute?
- Is the meeting's purpose transparent to every participant? To answer this, do not rely on the fact that there is a written description, but ask each participant.
- Which status is changed in which meeting?
- In which (other) situation is each status changed?
- What does "refined" mean in your context? Which attributes have to be filled out?

8.2. Effects on TD awareness

In the second workshop in August 2023, we introduced the relevance of considering alternatives, benefits, drawbacks, and risks to avoid cognitive biases, which changed the team's observable behavior initially, though not sustainably (Fig. 7(b)). Introducing a TD ticket type and respective attributes in February 2024 led to a rise in all recorded attributes like interest, contagiousness, or component (e.g., Fig. 8(b)). Further, the *Talked about TD* checkbox implemented led to a rise in discussed alternatives. The reminder for alternatives, benefits, drawbacks, and risks added in July 2024 led again to a rise in Fig. 7(b) due to the participants re-discussing the tickets. Information not recorded in attributes, e.g., quality attributes or functional requirements competing with a TD repayment, was less discussed in meetings, even though it seems to have been on the developers' minds, as Fig. 9(c) indicates. From this, we conclude that the workshop information only had a short-term effect on TD awareness, and long-term awareness can be reached if the learned information is included in the daily process by adding respective attributes to the ticket types.

In the observations, we further analyzed how often the danger of unconscious TD incurrence occurred and how often we identified a ticket to be TD where practitioners did not. While there were various risks of unconscious TD incurrence and controversies on what items were TD at the beginning of the study, we did not identify these instances anymore at the end. This seemingly proves the success of our workshop approach. However, just employing the workshop concept would not include the action research's "evaluation" phases. We cannot determine whether this success can be reached without a TD expert evaluating the meetings. This implies that we cannot conclude whether this success results from the workshop concept or the action research. Thus, evaluating these effects without presenting the evaluation results to the participants might be an intriguing next research step.

8.3. Discussion of the study design

In this section, we briefly discuss what we learned regarding the study design, which might be helpful for replication. Furthermore, we provide an assessment of the adopted and tested TD-SAGAT approach.

8.3.1. Lessons learned

First, similarly to the first topics in the previous section, the study was delayed due to negotiations with the administrators of the backlog tool and the implementation of the visualization tool. We are aware that sometimes an essential trust must be established before getting access to the company's tools, and thus, delays might be unavoidable. Therefore, the study's timetable must be particularly flexible to accommodate these uncertainties. Change management processes (Führing, 2021) assume such impasses and have methods to counteract them, which is why they adequately supplement this study design.

Second, we reduced the number of observed prerequisites over time to simplify the evaluation, focus it on the most critical aspects, and make the evaluations of the different methods comparable. For example, while we started the initial questionnaire with some pretty particular questions, e.g., whether the evolution of the affected component or its business value was taken into consideration, we ended up only asking whether the affected component was generally considered.

Third, we had problems evaluating the observations, as many tickets were discussed more than once, but usually, not all attributes were discussed. Particularly at the study's end, tickets were often re-discussed due to new insights and the *re-submission date*. The normalization of attribute discussions to the number of discussed tickets proved problematic. It showed low values only because the tickets had already been discussed, and their attribute's discussions had already taken place in previous meetings. We explained this in our analysis of those results, and future adoptions of this evaluation must similarly keep this effect in mind. We found TD-SAGAT to be a valuable supplement to pinpoint this issue, as the values for TD-SAGAT increased while the values for observation seemed to decrease.

Fourth, the studied team was newly assembled and had only 120 backlog tickets, which is a small amount that can be manually searched for TD tickets with a small effort. Yet, analyzing more extensive backlogs, particularly with older items in them, might not be easy to do manually. We suggest researching how machine learning mechanisms or language processing might help cluster backlog tickets and identify TD tickets.

Fifth, as mentioned in the previous section, we cannot differentiate whether some of the outcomes result from the workshop concept or the interventions of the action research. Thus, repeating this study without performing the action research's "evaluating" phase or at least without presenting those results to the practitioners could, therefore, extend the evaluation of the workshop concept's success.

Finally, while the amount of TD items did increase over time (Fig. 5), the satisfaction with the backlog improved (Fig. 4(a)). We discussed this seeming contradiction with the practitioners and found out that the improved satisfaction values were not rooted in the amount of TD but in the ability to manage the backlog and distinguish relevant from irrelevant TD. This implies that the reduction of TD items might not be as relevant to a successful TDM process as is currently expected. For future research, we advise that to assess the success of a TDM process, researchers should not focus too much on the number of TD items and should always take the subjective view of team members into account.

8.3.2. Assessment of the TD-SAGAT evaluation

The TD-SAGAT method is a newly employed method that proved its feasibility for usage in research during this action research. The method discloses insights beyond the observable behavior, and it analyzes underlying prerequisites instead of bluntly asking for awareness. Due to the situational interruptions, the information is still in the participants' memory. This reduces bias compared to answering questions in hindsight, where memory loss can lead to more biased assessments. However, the method is not intended to be used by practitioners as it would create too much overhead for them due to the analysis and the continuing meeting interruptions.

The TD-SAGAT results mostly supported the results of the workshop questionnaires and the observations but also showed some deviation

(see Section 7.3). We analyzed deviations and discussed them with the participants to identify whether the method was feasible and if an explanation for the deviation could be found. We identified the issues illustrated in the previous subsection regarding our observation's analysis and adjusted our final evaluation based on this. Additionally, we discussed why benefits, drawbacks, and risks were on the participants' minds (TD-SAGAT) but were not discussed (observations), which led to two significant changes: First, the team added a reminder to discuss these issues to the TD ticket template. Second, the participants identified the issue of *knowledge silos* as explained in Section 6.6 and changed their behavior to rotating the meeting moderation.

However, this method also has some shortcomings, particularly the low number of data points in our situation. We did not want to disturb and influence the participants too much, so we conducted this method only once a month. Additionally, not all participants were able to complete the surveys due to *knowledge silos*. Overall, TD-SAGAT provides snapshots of situations resulting in fluctuations in the TD-SAGAT figures. For the analysis of TD-SAGAT results, it is consequently essential to identify the long-term direction of the prerequisites, rather than focusing on these fluctuations. An issue in the survey creation was that we neglected the prerequisite of a TD item's contagiousness, which we would suggest adding as a prerequisite to future TD-SAGAT usage.

Still, in our opinion, this method is a valuable addition to method triangulation as it researches the thought process in addition to visual behavior (observations) and participants' opinions (surveys). Further, we were able to identify issues that we would not have determined without this method.

Finally, we would like to encourage researchers to adapt and use this method in other situations. For example, developers could be interrupted when implementing code to determine their thought processes in those situations.

9. Threats to validity

As explained in the introduction to this research method (Section 4.2), *ACM Special Interest Group on Software Engineering*'s empirical standards state that "example criteria include reflexivity, credibility, resonance, usefulness, and transferability", i.e., they are decisive for action research studies (SIGSOFT, 2025). For the reflexivity, we systematically reflect on our potential biases in the following subsections (Sections 9.1 to 9.4) by analyzing the common potential threats to validity mentioned in Staron's book on action research in software engineering (Staron, 2020). Regarding credibility, we provided exhaustive additional material (Wiese and Serwa, 2025). The resonance is addressed through the retrospective parts of each workshop (Section 6). To address the usefulness and transferability, we provide details on the workshops in Section 5 for replicating the study and recommendations for practitioners in Section 8.1.2. Generalizability refers to the ability to draw conclusions from quantitative research with a representative sample size, such as when conducting a survey of 1000 randomly chosen participants to generalize the results to the entire population. In contrast, transferability is a key concept in qualitative research, referring to the potential for research results to be applied in other contexts. In our case, the transferability is offered by replicating the workshops to establish TDM processes in other companies. In contrast to providing a fixed TDM process, the flexible approach of our workshop concept, where participants choose approaches that best fit their contexts, enables this transferability. The empirical standards further note that "positivist quality criteria such as internal validity, construct validity, generalizability, and reliability typically do not apply" (SIGSOFT, 2025). Nevertheless, we explicate and discuss these criteria for our study as far as possible in fulfillment of the criterion of reflexivity. We provide information on the threats by the names given to them in the book by Staron (2020).

9.1. Construct validity

First, we fall for the "mono-operation bias" because we study only one team of a big company. As with every IT team, this team has some special characteristics, e.g., central development with no real customer contact or the subteam of algorithm developers. For future work, we plan to replicate this study in other teams and domains to identify generalizable aspects. However, the results of this study have to be considered with this limitation in mind. Second, "interactions of different treatments" may have occurred. The new composition of this team allowed it to adopt a TDM process while still deciding on the general working processes. However, this means that changes in general working processes might have influenced the results as well as the TDM establishment. This threat can be mitigated by replicating this study in other contexts, which we plan for future work. Third, the "interaction of testing and treatment" also influenced our study. The recording of the meetings and the researchers' presence in some meetings might have influenced the participants' awareness of the TD topic. We mitigated this issue by not attending every meeting and recording meetings instead of attending. Further, for some results, we were not able to pinpoint whether they resulted from the workshop concept or action research (Section 8.2). We were not able to mitigate this threat in this study, but we advise replicating the study with an adapted study design to focus on these issues. Similarly, our study might be influenced by "hypothesis guessing" and "evaluation apprehension", i.e., the practitioners know the study's goal and that they are the objects of our study, which might result in them adjusting their behavior. We separated the action and reference teams and repeated the relevance of the reference team and open criticism at each workshop's beginning to balance some of the effects.

9.2. Internal validity

First, the team's "maturation" might have had an influence, which we already described for "interactions of different treatments". Particularly, the changing meeting structure might have influenced the observations, which were based on review & planning meetings for the first three months and on the refinement meetings at the end. In the beginning, the team still discussed and refined tickets during the review & planning meetings. Therefore, we assume that the overall impression of the observations is still valid. Nonetheless, we had to be careful when interpreting the results from these meetings. Second, asking about problems beforehand can lead to the perception that there is a problem. Staron called this a "testing" threat. Our first questionnaire contained such questions about the current situation. However, while this made the problems public, the decision to become active and introduce TDM was already made before the first questionnaire, as the pressure resulting from TD consequences was already high. Third, a "biased selection of subject" influences our study. Commonly, only willing industry teams already suffering from a certain problem will collaborate with researchers. This implies that our results might assist other teams willing to invest in a TDM process, but might not assist teams unprepared to invest. Finally, we cannot negate the "John Henry effect", i.e., that participants compete with a given baseline. However, our baseline was exceptionally low, i.e., a TDM process was nearly nonexistent. Therefore, we assume that this effect is of little relevance to our results.

9.3. Conclusion validity

First, "missing the needle in the haystack", i.e., missing important observations, and "seeing things that aren't there", i.e., observations unrelated to our intervention, might have influenced our study. Both threats were countered by two researchers who coordinated their observations with each other. While not all observations were made by the two researchers, they regularly compared their observations

to ensure that they were valid and that relevant information was included. Regardless, we might have missed some observations, but we assume that the overall tendency is still valid. Second, a "low reliability of measures" might have affected the TD-SAGAT results. We could use TD-SAGAT only about once a month so as not to influence the participants too much. Often, only one or two participants could respond regarding a specific ticket. This led to a low number of data points per month. This method may be more effective in other teams where team members have a similar background. Nevertheless, TD-SAGAT provided valuable information. Comparing observations with TD-SAGAT has revealed a problem with the team's know-how transfer, leading to countermeasures like rotating moderation to avoid *knowledge silos*. Third, the "low reliability of measures" might also be relevant for our questionnaires. The questionnaires were sometimes misleading, leading to questions from the participants. Thus, we improved the questionnaire during our work to simplify the questions. Finally, regarding the "random heterogeneity of respondents", in rare situations, the questionnaires were initially understood differently. However, researchers were present when the questionnaires were completed and were able to clarify uncertainties.

9.4. External validity

First, we examined a team that consciously wanted to tackle the topic of TDM and that had the full support of the team manager, which is a "selection bias" as similarly discussed for internal validity. This limits the transferability to teams that are similarly willing to establish a TDM process. However, it was not the aim of this study to establish a TDM process in unwilling teams. Second, if we control the context too much, it becomes difficult to generalize to non-experimental settings, which Staron describes as "reactive effects of experimental arrangements". We intervened controllingly by presenting certain research results, but we could not present all the research results on TD. Nevertheless, the practitioners could choose for themselves which methods they would adopt, and the whole workshop approach's goal is to enable the participants to choose what fits best into their environment without limiting the scope to one specific process. Third, we decided to do the workshops with a distance of at least eight weeks to avoid a "multiple treatment interference", i.e., when two or more actions simultaneously make it difficult to control the effects. Finally, Staron mentions "constructs, methods, and confounding factors", i.e., that measurements and methods might not allow us to generalize outside the study's context. In our study, the method is limited to agile working teams, as we adopt the idea of iterations and team decisions. This approach would work less well in a context where the team is not allowed such decisions, e.g., when a central group like architects or team managers leads such process decisions.

10. Conclusion

We conducted an action research study with five action cycles in 16 months. During this study, we presented TDM approaches from research papers to an IT team in five workshops structured along the TD activities.

This study showed which approaches the IT team adopted and which they dismissed. Further, we showed the effects the establishment of the TDM process had on the participants' TD awareness in decision-making situations. To evaluate TD awareness, we observed 35 team meetings, i.e., 25 meeting hours, and evaluated questionnaires from each workshop. Additionally, we used TD-SAGAT, a method adopted from the psychology domain to analyze situation awareness.

We found that two methods for TD prioritization and repayment were preferred, which are (1) repayment based on the system's evolution (Schmid, 2013a) and (2) repayment of *low-hanging fruits*, i.e., based on cost calculations (McConnell, 2008; Tom et al., 2013). For TD prevention, we found that presenting information, e.g., during the workshops, is insufficient, but reminders in the backlog's ticket templates, e.g., a *Talked about TD* checkbox, lead to a sustainable change. Finally, various new ideas for TDM emerged from this study: (1) using a *re-submission date*, (2) using a *Talked about TD* checkbox, (3) using *educated guesses* for priority instead of calculations, (4) using a *ROI-based* priority to improve the *educated guess* and (5) using visualizations for TD prioritization, e.g., to identify the *low-hanging fruits*.

For practitioners, this study outlined a five-step workshop approach for establishing TDM in practice. We demonstrated which research approaches were feasible in an embedded environment and presented five additional (new) approaches. Practitioners can use these as inspiration for establishing their own TDM process. We further provided an actionable list of recommendations for establishing a TDM process.

For researchers, we provided insights into which methods are feasible in practice and which might be less helpful. In particular, the resistance to a calculated, e.g., *ROI-based* priority is striking. Research could address this issue by providing methods to improve the *educated guess* priority instead of researching and providing calculations. Further, we identified that attribute fields in a backlog ticket can sustainably support TD awareness, including sound decision-making. This realization raises the research questions of which attributes should be part of backlog tickets, including for purposes other than supporting TDM. Including a TD ticket type and the relevant attributes in commercial backlog tools might further support the TDM establishment in practice and could be promoted by researchers. Finally, we presented a new method – TD-SAGAT – to analyze TD awareness adopted from psychology and critically discussed its suitability.

For future work, the team wishes to continue the collaboration to distribute the TDM process to other teams. We might approach this topic similarly to our workshops by adopting existing research methods, e.g., TD Guilds (Detofeno et al., 2021). Further, we plan to evaluate the five-step workshop concept in other teams from other domains. By this, we plan to optimize the workshop concept, compile a guideline comprising approaches that have been demonstrated to be best practices, and supplement these with approaches that work in specific environments. This list could be further improved by replicating the study in less agile work environments. As this research comprised researchers' interventions, it would be interesting to evaluate if this approach also works with less intervention by researchers. Another intriguing research topic would be to compare the TD awareness evaluation of this study with the results of TDM establishments of pre-defined processes. This comparison could determine whether the more open approach of this workshop concept leads to higher TD awareness. e.g., because participants may work more intensively on this topic.

CRediT authorship contribution statement

Marion Wiese: Writing – review & editing, Writing – original draft, Visualization, Validation, Resources, Project administration, Methodology, Funding acquisition, Formal analysis, Data curation, Conceptualization. **Kamila Serwa:** Writing – original draft, Formal analysis, Data curation. **Anastasia Besier:** Writing – original draft, Visualization, Software, Data curation. **Ariane S. Marion-Jetten:** Writing – review & editing, Methodology. **Eva Bittner:** Writing – review & editing, Supervision, Project administration.

Funding and declarations

The project on which this report is based was sponsored by the German Federal Ministry of Research, Technology and Space under the funding code 01IS24031. Responsibility for the content of this publication lies with the authors.

During the preparation of this work, the authors used Grammarly to improve the readability and language of the manuscript. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the content of the published article.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Marion Wiese reports financial support was provided by Federal Ministry of Education and Research of Germany under the funding code 01IS24031. Kamila Serwa reports financial support was provided by Federal Ministry of Education and Research of Germany under the funding code 01IS24031. Anastasia Besier reports financial support was provided by Federal Ministry of Education and Research of Germany under the funding code 01IS24031. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

We thank the case company and its team members for their participation, trust, and the valuable insights they provided. In memory of Prof. Dr.-Ing. André van Hoorn and Prof. Dr.-Ing. Matthias Riebisch.

Data availability

I shared our additional material on Zenodo under the following link: https://zenodo.org/records/14916678.

References

Ahmadjee, S., Mera-Gómez, C., Bahsoon, R., 2021. Assessing smart contracts security technical debts. pp. 6–15. http://dx.doi.org/10.1109/TechDebt52882.2021.00010.

Apa, C., Jeronimo, H., Nascimento, L.M., Vallespir, D., Travassos, G.H., 2020. The perception and management of technical debt in software startups. Fundam. Softw. Startups: Essent. Eng. Business Asp. 61–78. http://dx.doi.org/10.1007/978-3-030-35983-6_4.

Association, A.P., 2025. APA PsycInfo. URL https://www.apa.org/pubs/databases/psycinfo.

Avgeriou, P., Kruchten, P., Ozkaya, I., Seaman, C., 2016. Managing technical debt in software engineering. Dagstuhl Rep. 6 (4), 110–138. http://dx.doi.org/10.4230/DagRep.6.4.110.

Avgeriou, P., Ozkaya, I., Chatzigeorgiou, A., Ciolkowski, M., Ernst, N.A., Koontz, R.J., Poort, E., Shull, F., 2023. Technical debt management: The road ahead for successful software delivery. In: 2023 IEEE/ACM International Conference on Software Engineering: Future of Software Engineering (ICSE-FoSE). pp. 15–30. http://dx.doi.org/10.1109/ICSE-FoSE59343.2023.00007.

Baars, S., Meester, S., 2019. CodeArena: Inspecting and improving code quality metrics using minecraft. In: 2019 IEEE/ACM International Conference on Technical Debt (TechDebt). pp. 68–70. http://dx.doi.org/10.1109/TechDebt.2019.00023.

Baer, D., 2014. Dwight eisenhower nailed a major insight about productivity. URL https://www.businessinsider.com/dwight-eisenhower-nailed-a-major-insight-about-productivity-2014-4.

Baldassarre, M.T., Lenarduzzi, V., Romano, S., Saarimäki, N., 2020. On the diffuseness of technical debt items and accuracy of remediation time when using SonarQube. Inf. Softw. Technol. 128, 106377. http://dx.doi.org/10.1016/j.infsof.2020.106377.

Bass, L., Clements, P., Kazman, R., 2012. Software Architecture in Practice: Software Architect Practice_c3. Addison-Wesley.

Bazerman, M.H., Sezer, O., 2016. Bounded awareness: Implications for ethical decision making. Organ. Behav. Hum. Decis. Process. 136, 95–105. http://dx.doi.org/10.1016/j.obhdp.2015.11.004.

Berenguer, C., Borges, A., Freire, S., Rios, N., Tausan, N., Ramac, R., Pérez, B., Castellanos, C., Correal, D., Pacheco, A., López, G., Falessi, D., Seaman, C., Mandic, V., Izurieta, C., Spínola, R., 2021. Technical debt is not only about code and we need to be aware about it. pp. 1–12. http://dx.doi.org/10.1145/3493244.3493285.

Besker, T., Martini, A., Bosch, J., 2019. Technical debt triage in backlog management. In: 2019 IEEE/ACM International Conference on Technical Debt (TechDebt). TechDebt '19, IEEE Press, pp. 13–22.

Besker, T., Martini, A., Bosch, J., 2020. Carrot and stick approaches when managing technical debt. In: Proceedings of the 3rd International Conference on Technical Debt. TechDebt '20, Association for Computing Machinery, New York, NY, USA, pp. 21–30. http://dx.doi.org/10.1145/3387906.3388619.

Besker, T., Martini, A., Bosch, J., Tichy, M., 2017. An investigation of technical debt in automatic production systems. In: Proceedings of the XP2017 Scientific Workshops. Association for Computing Machinery, New York, NY, USA, http://dx.doi.org/10.1145/3120459.3120466.

Borowa, K., Almeida, R.R.d., Wiese, M., 2025. Debiasing architectural decision-making: An experiment with students and practitioners. http://dx.doi.org/10.48550/arXiv.2502.04011, arXiv:2502.04011 [cs].

Borowa, K., Jarek, M., Mystkowska, G., Paszko, W., Zalewski, A., 2022. Debiasing architectural decision-making: A workshop-based training approach. In: Gerostathopoulos, I., Lewis, G., Batista, T., Bureš, T. (Eds.), Software Architecture. Springer International Publishing, Cham, pp. 159–166. http://dx.doi.org/10.1007/978-3-031-16697-6_11.

Borowa, K., Zalewski, A., Kijas, S., 2021. The influence of cognitive biases on architectural technical debt. In: International Conference on Software Architecture (ICSA). pp. 115–125. http://dx.doi.org/10.1109/ICSA51549.2021.00019.

Borup, N.B., Christiansen, A.L.J., Tovgaard, S.H., Persson, J.S., 2021. Deliberative technical debt management: An action research study. In: Wang, X., Martini, A., Nguyen-Duc, A., Stray, V. (Eds.), Software Business. In: Lecture Notes in Business Information Processing, Springer International Publishing, Cham, pp. 50–65. http://dx.doi.org/10.1007/978-3-030-91983-2_5.

Buschmann, F., 2011. To pay or not to pay technical debt. IEEE Softw. 28 (6), 29–31. http://dx.doi.org/10.1109/MS.2011.150.

Chiesa, M., Hobbs, S., 2008. Making sense of social research: How useful is the Hawthorne effect? Eur. J. Soc. Psychol. 38 (1), 67–74. http://dx.doi.org/10.1002/ejsp.401.

Chugh, D., Bazerman, M.H., 2007. Bounded awareness: What you fail to see can hurt you. Mind & Soc. 6 (1), 1–18. http://dx.doi.org/10.1007/s11299-006-0020-4.

Crespo, Y., López-Nozal, C., Marticorena-Sánchez, R., Gonzalo-Tasis, M., Piattini, M., 2022. The role of awareness and gamification on technical debt management. Inf. Softw. Technol. 150 (April), 106946. http://dx.doi.org/10.1016/j.infsof.2022.106946.

Cunningham, W., 1992. The WyCash portfolio management system. In: Proceedings of the Conference on Object-Oriented Programming Systems, Languages, and Applications, OOPSLA. vol. Part F1296, pp. 29–30. http://dx.doi.org/10.1145/157710.157715.

Dawson, M.E., Schell, A.M., 1987. Human autonomic and skeletal classical conditioning: The role of conscious cognitive factors.

Detofeno, T., Malucelli, A., Reinehr, S., 2022. PriorTD: A method for prioritization technical debt. In: Proceedings of the XXXVI Brazilian Symposium on Software Engineering. SBES '22, Association for Computing Machinery, New York, NY, USA, pp. 230–240. http://dx.doi.org/10.1145/3555228.3555238.

Detofeno, T., Reinehr, S., Andreia, M., 2021. Technical debt guild: When experience and engagement improve technical debt management. In: Proceedings of the XX Brazilian Symposium on Software Quality. SBQS '21, Association for Computing Machinery, New York, NY, USA, pp. 1–10. http://dx.doi.org/10.1145/3493244.3493271.

Dong, Q.H., Ocker, F., Vogel-Heuser, B., 2019. Technical Debt as indicator for weaknesses in engineering of automated production systems. Prod. Eng. 13 (3–4), 273–282. http://dx.doi.org/10.1007/s11740-019-00897-0, Publisher: Springer Berlin Heidelberg ISBN: 0123456789.

Eliasson, U., Martini, A., Kaufmann, R., Odeh, S., 2015. Identifying and visualizing architectural debt and its efficiency interest in the automotive domain: A case study. In: 2015 IEEE 7th International Workshop on Managing Technical Debt (MTD). pp. 33–40. http://dx.doi.org/10.1109/MTD.2015.7332622.

Endsley, M.R., 1988. Design and evaluation for situation awareness enhancement. Proc. Hum. Factors Soc. Annu. Meet. 32 (2), 97–101. http://dx.doi.org/10.1177/154193128803200221.

Endsley, M.R., 2000. Direct measurement of situation awareness: Validity and use of SAGAT. In: Endsley, M.R., Garland, D.J. (Eds.), Situation Awareness Analysis and Measurement. Routledge, Mahwah, NJ.

Ernst, N.A., Bellomo, S., Ozkaya, I., Nord, R.L., Gorton, I., 2015. Measure it? Manage it? Ignore it? Software practitioners and technical debt. In: 10th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, ESEC/FSE 2015 - Proceedings. Association for Computing Machinery, New York, NY, USA, pp. 50–60. http://dx.doi.org/10.1145/2786805.2786848.

Ernst, N., Kazman, R., Delange, J., 2021. Technical Debt in Practice. The MIT Press Cambridge, Massachusetts, London, England.

Falessi, D., Reichel, A., 2015. Towards an open-source tool for measuring and visualizing the interest of technical debt. In: 2015 IEEE 7th International Workshop on Managing Technical Debt (MTD). pp. 1–8. http://dx.doi.org/10.1109/MTD.2015.7332618.

Fernández-Sánchez, C., Humanes, H., Garbajosa, J., Díaz, J., 2017. An open tool for assisting in technical debt management. In: 2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA). pp. 400–403. http://dx.doi.org/10.1109/SEAA.2017.60.

Fowler, M., 2009. Technical debt quadrant. URL https://martinfowler.com/bliki/TechnicalDebtQuadrant.html.

Freire, S., Rios, N., Gutierrez, B., Torres, D., Mendonça, M., Izurieta, C., Seaman, C., Spinola, R.O., 2020. Surveying Software Practitioners on Technical Debt Payment Practices and Reasons for Not Paying off Debt Items. In: Proceedings of the Evaluation and Assessment in Software Engineering. EASE '20, Association for Computing Machinery, New York, NY, USA, pp. 210–219. http://dx.doi.org/10.1145/3383219.3383241.

Freire, S., Rios, N., Pérez, B., Castellanos, C., Correal, D., Ramač, R., Mandić, V., Taušan, N., López, G., Pacheco, A., Mendonça, M., Falessi, D., Izurieta, C., Seaman, C., Spínola, R., 2023. Software practitioners' point of view on technical debt payment. J. Syst. Softw. 196, 111554. http://dx.doi.org/10.1016/j.jss.2022.111554.

Führing, M., 2021. Change Management Essentials for Managers and Smart Leaders. Nomos Verlagsgesellschaft, Baden-Baden, Germany, URL https://doi.org/10.5771/9783957102935.

Gershon, N., Page, W., 2001. What storytelling can do for information visualization. Commun. ACM 44 (8), 31–37. http://dx.doi.org/10.1145/381641.381653.

Guo, Y., Seaman, C., da Silva, F.Q., Q.B. da Silva, F., da Silva, F.Q., Q.B. da Silva, F., 2016a. Costs and obstacles encountered in technical debt management – A case study. J. Syst. Softw. 120, 156–169. http://dx.doi.org/10.1016/j.jss.2016.07.008.

Guo, Y., Spínola, R.O., Seaman, C., 2016b. Exploring the costs of technical debt management – a case study. Empir. Softw. Eng. 21 (1), 159–182. http://dx.doi.org/10.1007/s10664-014-9351-7.

Holvitie, J., Leppänen, V., 2013. DebtFlag: Technical debt management with a development environment integrated tool. In: 2013 4th International Workshop on Managing Technical Debt (MTD). pp. 20–27. http://dx.doi.org/10.1109/MTD.2013.6608674.

Holvitie, J., Licorish, S.A., Spínola, R.O., Hyrynsalmi, S., MacDonell, S.G., Mendes, T.S., Buchan, J., Leppänen, V., 2018. Technical debt and agile software development practices and processes: An industry practitioner survey. Inf. Softw. Technol. 96, 141–160. http://dx.doi.org/10.1016/j.infsof.2017.11.015.

ISO/IEC, 2023. ISO 25010-2023 | systems and software engineering - systems and software quality requirements and evaluation (square) - system and software quality models. URL https://www.iso.org/obp/ui/en/#iso:std:iso-iec:25010:ed-2:v1:en.

Jaspan, C., Green, C., 2023. Defining, measuring, and managing technical debt. IEEE Softw. 40 (3), 15–19. http://dx.doi.org/10.1109/MS.2023.3242137, Conference Name: IEEE Software.

2025. Jira | issue & project tracking software | atlassian. URL https://www.atlassian.com/software/jira.

Junior, H.J., Travassos, G.H., 2022. Consolidating a common perspective on technical debt and its management through a tertiary study. Inf. Softw. Technol. 149 (October 2020), 106964. http://dx.doi.org/10.1016/j.infsof.2022.106964.

Kosara, R., Mackinlay, J., 2013. Storytelling: The next step for visualization. Computer 46 (5), 44–50. http://dx.doi.org/10.1109/MC.2013.36.

Kruchten, P., Nord, R.L., Ozkaya, I., 2012a. Technical debt: From metaphor to theory and practice. IEEE Softw. 29 (6), 18–21. http://dx.doi.org/10.1109/MS.2012.167.

Kruchten, P., Nord, R., Ozkaya, I., 2019. Managing Technical Debt: Reducing Friction in Software Development. Software Engineering Institute, Carnegie Mellon University.

Kruchten, P., Nord, R.L., Ozkaya, I., Visser, J., 2012b. Technical debt in software development. ACM SIGSOFT Softw. Eng. Notes 37 (5), 36–38. http://dx.doi.org/10.1145/2347696.2347698.

Lenarduzzi, V., Besker, T., Taibi, D., Martini, A., Fontana, F.A., 2020. A systematic literature review on technical debt prioritization: Strategies, processes, factors, and tools (journal version). J. OfSystems & Softw. J. 171 (171), http://dx.doi.org/10.1016/j.jss.2020.110827.

Lezcano Airaldi, A., Diaz-Pace, J.A., Irrazábal, E., 2021. Data-driven storytelling to support decision making in crisis settings: A case study. JUCS - J. Univers. Comput. Sci. 27 (10), 1046–1068. http://dx.doi.org/10.3897/jucs.66714, Number: 10 Publisher: Journal of Universal Computer Science.

Li, Z., Avgeriou, P., Liang, P., 2015. A systematic mapping study on technical debt and its management. J. Syst. Softw. 101, 193–220. http://dx.doi.org/10.1016/j.jss.2014.12.027.

Li, Y., Soliman, M., Avgeriou, P., 2022. Identifying self-admitted technical debt in issue tracking systems using machine learning. Empir. Softw. Eng. 27 (6), 131. http://dx.doi.org/10.1007/s10664-022-10128-3.

Limesurvey, 2025. LimeSurvey — Free online survey tool. URL https://www.limesurvey.org/.

Liu, J., Huang, Q., Xia, X., Shihab, E., Lo, D., Li, S., 2020. Is using deep learning frameworks free? Characterizing technical debt in deep learning frameworks. In: Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering in Society. Association for Computing Machinery, New York, NY, USA, pp. 1–10. http://dx.doi.org/10.1145/3377815.3381377.

Martini, A., Besker, T., Bosch, J., 2018. Technical debt tracking: Current state of practice: A survey and multiple case study in 15 large organizations. Sci. Comput. Program. 163, 42–61.

Martini, A., Bosch, J., 2015. The danger of architectural technical debt: Contagious debt and vicious circles. In: Proceedings - 12th Working IEEE/IFIP Conference on Software Architecture, WICSA 2015. pp. 1–10. http://dx.doi.org/10.1109/WICSA.2015.31.

Maximini, D., 2018. The Scrum Culture. Management for Professionals, Springer International Publishing, Cham, http://dx.doi.org/10.1007/978-3-319-73842-0.

McCambridge, J., Witton, J., Elbourne, D.R., 2014. Systematic review of the hawthorne effect: New concepts are needed to study research participation effects. J. Clin. Epidemiol. 67 (3), 267–277. http://dx.doi.org/10.1016/j.jclinepi.2013.08.015.

McConnell, S., 2008. Managing Technical Debt. Construx Inc..

Mendes, T.S., Gomes, F.G., Gonçalves, D.P., Mendonça, M.G., Novais, R.L., Spínola, R.O., 2019. VisminerTD: A tool for automatic identification and interactive monitoring of the evolution of technical debt items. J. Braz. Comput. Soc. 25 (1), http://dx.doi.org/10.1186/s13173-018-0083-1.

Meyer, B., 2014. Agile!: The Good, the Hype and the Ugly. Springer International Publishing, Cham, http://dx.doi.org/10.1007/978-3-319-05155-0.

Microsoft, 2023. Add portfolio backlogs - Azure DevOps. URL https://learn.microsoft.com/en-us/azure/devops/reference/add-portfolio-backlogs?view=azure-devops.

Microsoft, 2024. Use backlogs to manage projects - Azure Boards. URL https://learn.microsoft.com/en-us/azure/devops/boards/backlogs/backlogs-overview?view=azure-devops.

Microsoft, 2025a. Azure DevOps services | microsoft azure. URL https://azure.microsoft.com/en-us/products/devops.

Microsoft, 2025b. Free online spreadsheet software Excel | microsoft 365. URL https://www.microsoft.com/en-us/microsoft-365/excel.

Microsoft, 2025c. Power BI - data visualization | microsoft power platform. URL https://www.microsoft.com/en-us/power-platform/products/power-bi.

Microsoft, 2025d. Video conferencing, meetings, calling | microsoft teams. URL https://www.microsoft.com/en-us/microsoft-teams/group-chat-software.

Miro, 2025. What is the Eisenhower matrix? | Miro. URL https://miro.com/strategic-planning/what-is-an-eisenhower-matrix/.

Mural, 2025a. Retrospective template | mural. URL https://www.mural.co/templates/retrospective.

Mural, 2025b. Work better together with Mural's visual work platform. URL https://www.mural.co/.

Ocker, F., Seitz, M., Oligschlager, M., Zou, M., Vogel-Heuser, B., Oligschläger, M., Zou, M., Vogel-Heuser, B., 2019. Increasing awareness for potential technical debt in the engineering of production systems. In: 2019 IEEE 17th International Conference on Industrial Informatics (INDIN). vol. 1, Institute of Electrical and Electronics Engineers Inc., pp. 478–484. http://dx.doi.org/10.1109/INDIN41052.2019.8972268.

Oliveira, F., Goldman, A., Santos, V., 2015. Managing technical debt in software projects using scrum: An action research. In: Proceedings - 2015 Agile Conference, Agile 2015. pp. 50–59. http://dx.doi.org/10.1109/Agile.2015.7.

Perera, J., Tempero, E., Tu, Y.-C., Blincoe, K., 2024. A systematic mapping study exploring quantification approaches to code, design, and architecture technical debt. ACM Trans. Softw. Eng. Methodol. http://dx.doi.org/10.1145/3675393, Just Accepted.

Pérez, B., Castellanos, C., Correal, D., Rios, N., Freire, S., Spínola, R., Seaman, C., Izurieta, C., 2021. Technical debt payment and prevention through the lenses of software architects. Inf. Softw. Technol. 140, 106692. http://dx.doi.org/10.1016/j.infsof.2021.106692.

Pimentel, J.L., 2019. Some biases in likert scaling usage and its correction. Int. J. Sci.: Basic Appl. Research (IJSBAR) 45 (1), 183–191.

Rachow, P., Riebisch, M., 2022. An architecture smell knowledge base for managing architecture technical debt. In: Proceedings of the International Conference on Technical Debt. TechDebt '22, Association for Computing Machinery, New York, NY, USA, pp. 1–10. http://dx.doi.org/10.1145/3524843.3528092.

Ramač, R., Mandić, V., Taušan, N., Rios, N., Freire, S., Pérez, B., Castellanos, C., Correal, D., Pacheco, A., Lopez, G., Izurieta, C., Seaman, C., Spinola, R., 2021. Prevalence, common causes and effects of technical debt: Results from a family of surveys with the IT industry. J. Syst. Softw. http://dx.doi.org/10.1016/j.jss.2021.111114.

Ramač, R., Mandić, V., Taušan, N., Rios, N., Freire, S., Pérez, B., Castellanos, C., Correal, D., Pacheco, A., Lopez, G., et al., 2022. Prevalence, common causes and effects of technical debt: Results from a family of surveys with the IT industry. J. Syst. Softw. 184, 111114. http://dx.doi.org/10.1016/j.jss.2021.111114.

Ramasubbu, N., Kemerer, C.F., 2019. Integrating technical debt management and software quality management processes: A framework and field test. Test Eng. Manag. 45 (7–8), 883. http://dx.doi.org/10.1109/TSE.2017.2774832.

RandomTicker, 2025. Random ticker – apps bei google play. URL https://play.google.com/store/apps/details/?id=com.cvv.fanstaticapps.randomticker&hl=de.

Reboucas De Almeida, R., Kulesza, U., Treude, C., Cavalcanti Feitosa, D., Lima, A.H.G., 2018. Aligning technical debt prioritization with business objectives: A multiple-case study. Proc. - 2018 IEEE Int. Conf. Software Maint. Evol. ICSME 2018 655–664, arXiv:1807.05582.

Rios, N., de Mendonça Neto, M.G., Spínola, R.O., 2018. A tertiary study on technical debt: Types, management strategies, research trends, and base information for practitioners. Inf. Softw. Technol. 102, 117–145. http://dx.doi.org/10.1016/j.infsof.2018.05.010.

Rios, N., Oliveira Spínola, R., de Mendonça Neto, M.G., Seaman, C., 2019. Supporting analysis of technical debt causes and effects with cross-company probabilistic cause-effect diagrams. In: 2019 IEEE/ACM International Conference on Technical Debt (TechDebt). pp. 3–12. http://dx.doi.org/10.1109/TechDebt.2019.00009.

Rios, N., Spínola, R.O., Mendonça, M., Seaman, C., 2020. The practitioners' point of view on the concept of technical debt and its causes and consequences: A design for a global family of industrial surveys and its first results from Brazil. Empir. Softw. Eng. 25 (5), 3216–3287. http://dx.doi.org/10.1007/s10664-020-09832-9.

Rocha, J.C., Zapalowski, V., Nunes, I., 2017. Understanding technical debt at the code level from the perspective of software developers. In: Proceedings of the 31st Brazilian Symposium on Software Engineering. SBES '17, Association for Computing Machinery, New York, NY, USA, pp. 64–73. http://dx.doi.org/10.1145/3131151.3131164.

Royal, K.D., 2016. The impact of item sequence order on local item dependence: An item response theory perspective. Surv. Pr. 9 (5), http://dx.doi.org/10.29115/SP-2016-0027.

Rubin, K.S., 2012. Essential scrum: A practical guide to the most popular agile process.

Russo, B., Camilli, M., Mock, M., 2022. WeakSATD: Detecting weak self-admitted technical debt. In: 2022 IEEE/ACM 19th International Conference on Mining Software Repositories (MSR). pp. 448–453. http://dx.doi.org/10.1145/3524842.3528469.

Schmid, K., 2013a. A formal approach to technical debt decision making. In: QoSA 2013 - Proceedings of the 9th International ACM Sigsoft Conference on the Quality of Software Architectures. pp. 153–162. http://dx.doi.org/10.1145/2465478.2465492, ISBN: 9781450321266.

Schmid, K., 2013b. On the limits of the technical debt metaphor: Some guidance on going beyond. In: Fourth Workshop on Managing Technical Debt, Workshop At the International Conference on Software Engineering. IEEE, pp. 63–66. http://dx.doi.org/10.1109/MTD.2013.6608681.

Schwarz, N., Stone, A., Shiffman, S., Atienza, A., Nebeling, L., 2007. The science of real-time data capture: Self-reports in health research. Retrosp. Concurr. Self-Reports: Ration. for Real-Time Data Capture 11–27.

Seaman, C., Guo, Y., 2011. Measuring and monitoring technical debt. In: Zelkowitz, M.V. (Ed.), Advances in Computers. vol. 82, Elsevier, pp. 25–46. http://dx.doi.org/10.1016/B978-0-12-385512-1.00002-5.

SIGSOFT, A., 2025. ACM special interest group on software engineering's empirical standards for action research. URL https://www2.sigsoft.org/EmpiricalStandards/docs/standards?standard=ActionResearch#.

Silva, L., Unterkalmsteiner, M., Wnuk, K., 2022. MultiDimEr: A multi-dimensional bug analyzer. In: Proceedings of the International Conference on Technical Debt. TechDebt '22, Association for Computing Machinery, New York, NY, USA, pp. 66–70. http://dx.doi.org/10.1145/3524843.3528099.

Soliman, M., Avgeriou, P., Li, Y., 2021. Architectural design decisions that incur technical debt — An industrial case study. Inf. Softw. Technol. 139, 106669. http://dx.doi.org/10.1016/j.infsof.2021.106669.

Sonar, 2025. Better code & better software | ultimate security and quality | sonar. URL https://www.sonarsource.com/.

Staron, M., 2020. Action Research in Software Engineering: Theory and Applications. Springer International Publishing, Cham, http://dx.doi.org/10.1007/978-3-030-32610-4.

Stol, K.J., Fitzgerald, B., 2018. The ABC of software engineering research. ACM Trans. Softw. Eng. Methodol. 27 (3), http://dx.doi.org/10.1145/3241743.

Tom, E., Aurum, A., Vidgen, R., 2013. An exploration of technical debt. J. Syst. Softw. 86 (6), 1498–1516. http://dx.doi.org/10.1016/j.jss.2012.12.052.

Tonin, G.S., Goldman, A., Seaman, C., Pina, D., 2017. Effects of technical debt awareness: A classroom study. In: Baumeister, H., Lichter, H., Riebisch, M. (Eds.), 18th International Conference, XP 2017. pp. 84–100. http://dx.doi.org/10.1007/978-3-319-57633-6_6.

Vogel-Heuser, B., Bi, F., 2020. Interdisciplinary effects of technical debt in companies with mechatronic products — a qualitative study. J. Syst. Softw. 171, 110809. http://dx.doi.org/10.1016/j.jss.2020.110809.

Wiese, M., Borowa, K., 2023. IT managers' perspective on technical debt management. J. Syst. Softw. 202, 111700. http://dx.doi.org/10.1016/j.jss.2023.111700.

Wiese, M., Heinrichs, A., Rusieshvili, N., de Almeida, R.R., Borowa, K., 2025. The TechDebt game - enabling discussions about technical debt. URL https://arxiv.org/abs/2502.02174.

Wiese, M., Rachow, P., Riebisch, M., Schwarze, J., 2022. Preventing technical debt with the TAP framework for technical debt aware management. Inf. Softw. Technol. 106926. http://dx.doi.org/10.1016/j.infsof.2022.106926.

Wiese, M., Serwa, K., 2025. Additional material for establishing technical debt management - a five-step workshop approach and an action research study. URL https://doi.org/10.5281/zenodo.14916678.

Wikipedia, 2025. Time management, eisenhower method. Page Version ID: 1277294433. URL https://en.wikipedia.org/w/index.php?title=Time_management&oldid=1277294433#The_Eisenhower_Method.

Yli-Huumo, J., Maglyas, A., Smolander, K., 2016a. How do software development teams manage technical debt? – An empirical study. J. Syst. Softw. 120, 195–218. http://dx.doi.org/10.1109/TechDebt.2019.00009.

Yli-Huumo, J., Maglyas, A., Smolander, K., Haller, J., Törnroos, H., 2016b. Developing processes to increase technical debt visibility and manageability – an action research study in industry. In: Abrahamsson, P., Jedlitschka, A., Nguyen Duc, A., Felderer, M., Amasaki, S., Mikkonen, T. (Eds.), Product-Focused Software Process Improvement. Springer International Publishing, Cham, pp. 368–378. http://dx.doi.org/10.1007/978-3-319-49094-6_24.

Zalewski, A., Borowa, K., Ratkowski, A., 2017. On cognitive biases in architecture decision making. In: Lopes, A., de Lemos, R. (Eds.), Software Architecture. Springer International Publishing, Cham, pp. 123–137. http://dx.doi.org/10.1007/978-3-319-65831-5_9.

Zampetti, F., Fucci, G., Serebrenik, A., Di Penta, M., 2021. Self-admitted technical debt practices: A comparison between industry and open-source. Empir. Softw. Eng. 26 (6), 131. http://dx.doi.org/10.1007/s10664-021-10031-3.

Zhou, X., Han, J., Qin, H., Xue, C., 2023. Research on multilevel situation awareness changes under the cumulative effect of mental fatigue. Cogn. Technol. Work. 1–13. http://dx.doi.org/10.1007/s10111-023-00723-99.

Chapter 16

A Practical Guide for Establishing a Technical Debt Management Process

7 (Practical Guide)	
Title	A Practical Guide for Establishing a Technical Debt Management Process
Authors	Wiese M., Serwa K., Bittner E.
Year	2026
Conference	IEEE/ACM International Conference on Technical Debt (TechDebt)
Reference	[WSB26]

TABLE 16.1: Publication No. 7 *(Practical Guide)*

A Practical Guide for Establishing a Technical Debt Management Process

Marion Wiese
University of Hamburg
Hamburg, Germany
marion.wiese@uni-hamburg.de

Kamila Serwa
University of Hamburg
Hamburg, Germany
kamila.serwa@uni-hamburg.der

Eva Bittner
University of Hamburg
Hamburg, Germany
eva.bittner@uni-hamburg.de

Abstract

Context. Technical Debt (TD) refers to short-term beneficial software solutions that impede future changes, making TD management essential. However, establishing a TD management (TDM) process is one of the most pressing concerns in practice. *Goal.* We plan to identify which previously researched TDM approaches are feasible in practice and what typical challenges emerge to create a guideline for establishing a TDM process. *Method.* We replicated our previously published action research study by conducting five workshops introducing TDM with two teams from different companies. To determine the feasibility of TDM approaches, we presented the teams with approaches for various TD activities and let them decide which to adopt. Overall, we conducted 19 workshops and retrospectives, analyzing 108 meetings (96 hours) over a 30 -month period. *Results.* The adopted TD prevention strategies and documentation were similar in all teams. The teams utilized their respective backlogs and created a new backlog item type for TD, incorporating similar attributes such as *interest*, *contagiousness*, a *resubmission date*, and reminders to discuss drawbacks and risks. However, they used different prioritization approaches and deviating repayment methods. The teams had to overcome similar challenges during the establishment, which we list in this paper. *Conclusions.* We identified the TDM approaches used by all teams as a starting point for best practices. For challenges, we provided solutions or identified them as research gaps. *Issue tracking system* vendors should implement TD issue types employing the identified attributes. Finally, we created a white paper for practitioners to establish a TDM process based on our results.

CCS Concepts

• **Software and its engineering → Collaboration in software development**; **Software development methods**.

Keywords

Technical Debt, Technical Debt Management, Action Research, TDM Guide

ACM Reference Format:
Marion Wiese, Kamila Serwa, and Eva Bittner. 2026. A Practical Guide for Establishing a Technical Debt Management Process. In *International Conference on Technical Debt (TechDebt '26), April 12–13, 2026, Rio de Janeiro, Brazil.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3794915.3795789



This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.
TechDebt '26, Rio de Janeiro, Brazil
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2485-5/26/04
https://doi.org/10.1145/3794915.3795789

1 INTRODUCTION

Technical debt (TD) is the term for constructs in software systems that are beneficial in the short term, but hinder future changes [4]. These constructs can occur in various elements of the system or software engineering process, e.g., code, tests, or requirements [22, 39]. Managing TD is divided into multiple so-called TD activities, such as TD identification, TD documentation, TD measurement, TD prioritization, TD monitoring, TD repayment, and TD prevention [39].

Many studies on managing TD have been conducted, and various approaches to managing it have been developed over the last few years [25, 34, 52, 57, 67]. Unfortunately, the transfer of these results to practice remains inadequate and was defined as a goal on the research roadmap (Phase 3) of the Dagstuhl Manifesto "Reframing technical debt" [6]. Many case studies report the successful establishment of TD management processes and provide valuable approaches [10, 25, 28, 29, 40, 50, 52, 67, 70]. Yet, these studies lack a common strategy and guidance for transferring results to other companies. Consequently, a *"shared 'standard' best-practice processes and guidelines for TDM"* was also determined as part of the manifesto's roadmap [6]. In our previous study, we proposed a 5-step workshop concept for establishing a TDM process in practice, allowing practitioners to choose from a variety of research approaches. Moreover, we offered detailed guidance on how to apply this strategy to other companies [68].

The **goal** of this study is to create a TDM guide to identify best practices by replicating the workshop approach at TRUMPF SE & Co. KG and DATEV eG. We aim to uncover new insights by analyzing commonalities and differences in their approaches, and we will assess the challenges they encountered to guide future research. The study addresses the following research questions (RQ):

RQ 1: Which approaches from research do practitioners adopt to manage TD in practice? By this, we plan to identify commonalities and differences between the companies. This may provide initial guidance on establishing best practices for managing TD, which could be further improved through additional replications.

RQ 2: Which challenges commonly emerge when establishing a TD management process in practice? With this RQ, we aim to identify typical challenges that arise during this establishment. The respective solutions identified by the teams support the development of the TDM guide. In cases where the teams encountered challenges but were unable to find a solution, we reveal further research opportunities.

RQ 3: How is the awareness of TD impacted when a process for managing TD is established? Finally, we measure the TD awareness of the participants, as we did in our initial study [68]. With this, we provide more data on the impact of the 5-step workshop approach and the resulting TDM processes.

To answer these questions, we conducted 15 workshops and 4 retrospectives across all three teams (i.e., including our previous study). We analyzed a total of 108 meetings over 96 hours over a period of 30 months. We employed systematic observations to analyze the meetings, questionnaires during the workshops, and the so-called TD-SAGAT surveys, interrupting some of the meetings to assess the team's TD awareness, as introduced in our previous paper [68].

As the **contributions** of our study, we were able to generalize the results of our previous study by identifying TDM approaches used by all teams as a starting point for "best practices for TDM." Those approaches include (1) establishing a TD champion/manager, (2) a list of attributes with concrete values to use when documenting TD, (3) evaluating alternatives with drawbacks and risks to debias decision-making, (3) prioritizing TD against each other rather than against functional requirements, and (4) utilizing a *resubmission date*. Regarding typical challenges, we identified that teams need (1) more concrete support than currently given; (2) a simple approach to identify TD in a backlog; (3) support on how to start the process; (4) reminders to adhere to the process in the beginning, preferably by an outside person; (5) support in formulating a TD item's consequences. Based on our results, we developed a white paper for practitioners titled *"A Practical Guide to Establish TDM"*, which is part of our additional material [69]. Additionally, we identified suggestions for *issue tracking system* vendors on how to incorporate TDM into their tools, including a TD issue type for recording TD items in the backlog and visualizing these issues. Finally, we were able to support our previous findings that overall awareness of TD increased only when related attributes were included in the issue types as steady reminders.

In Section 2, we outline our research methodology, which is based on our previous study [68]. We present the results and answer the research questions in Section 3 and discuss the most relevant insights and suggestions for future research in Section 4. We investigate the related work in Section 5 and discuss the threats to validity in Section 6. With Section 7, we conclude our paper and summarize the contributions for practitioners and researchers.

2 METHOD

Our study replicates the 5-step workshop approach and employs the action research methodology as presented in our previous study [68]. Below, we provide a brief overview of the action research, introduce the data collection approaches, and the analysis of the data aligned with the RQs.

2.1 Action Research

Action research is classified as a field experiment, i.e., it is performed in a natural, pre-existing setting [61]. In contrast to other research methods, the researchers are allowed to interact with the study's participants [60]. This offers the advantages of collaborating with practitioners and embedding their feedback directly. These advantages are essential to our goal of understanding why research approaches are adopted or not by practitioners. Moreover, action research follows an iterative approach that aligns with the agile mindset prevalent in most IT teams and our iterative workshop approach.

Company	Domain	Employees	Organizational Form
DATEV eG	Finances & Taxes	> 8,000	cooperative
anonymus	Consumer goods	> 2,000	family business
TRUMPF SE & Co. KG	Machine tools	> 15,000	family business

Table 1: Participating companies, including the one from [68]
(The order of the entries has been randomized for anonymization purposes)

Team	Size	Team's main task	Issue-tracking System
A	8	Providing a central service	Azure DevOps [45]
B	5	Frontend (+Backend for Frontend)	Jira [2]
C	8	Central communication platform	GitLab Issues [1]

Table 2: Participating teams, including team A from [68]

Our action research is conducted in iteratively repeating action cycles following Staron's [60] guidelines. Each action cycle comprises five phases as shown in Figure 1 with different colors for each phase: (1) *Diagnosing* to identify the problems (green); (2) *Action planning* to identify actions to solve the problem (yellow); (3) *Action taking*, i.e., executing the identified actions (red); (4) *Evaluating* and analyzing the effects and the success of the chosen solutions (violet); (5) *Learning* to identify further improvements (blue).

2.1.1 Case Companies and Participants. We replicated our previous study in two teams from different companies with diverse backgrounds and team tasks to offer diverse perspectives. Both companies were selected for this study through the German Software Campus project [3]. For this paper, we combined our results with those from the initial study to compare the adopted approaches, challenges, and effects. We provide the details on the chosen companies and teams in Tables 1 to 2 and on the team members in Table 3.

2.1.2 Workshops. As described in our previous study, each of the five workshops consisted of the *Evaluating*, *Learning*, *Diagnosing*, and *Action Planning* phases of the respective action research cycle. The first workshop had a distinct structure from all other workshops. The first workshop's focus is on getting to know one another, analyzing the team's current TDM processes, and organizing the upcoming work. Workshops two to five followed a similar structure to introduce TD concepts. First, we presented and discussed the evaluation results, i.e., the results from the surveys and observations (*Evaluating* phase). We employed a retrospective workshop method for the *Learning* phase. After that, participants completed a survey to evaluate the study's progress. Then, we presented previously researched approaches regarding one or two TD activities (*Diagnosing* phase). The team discussed which presented approaches to employ or developed new approaches to try out (*Action planning* phase).

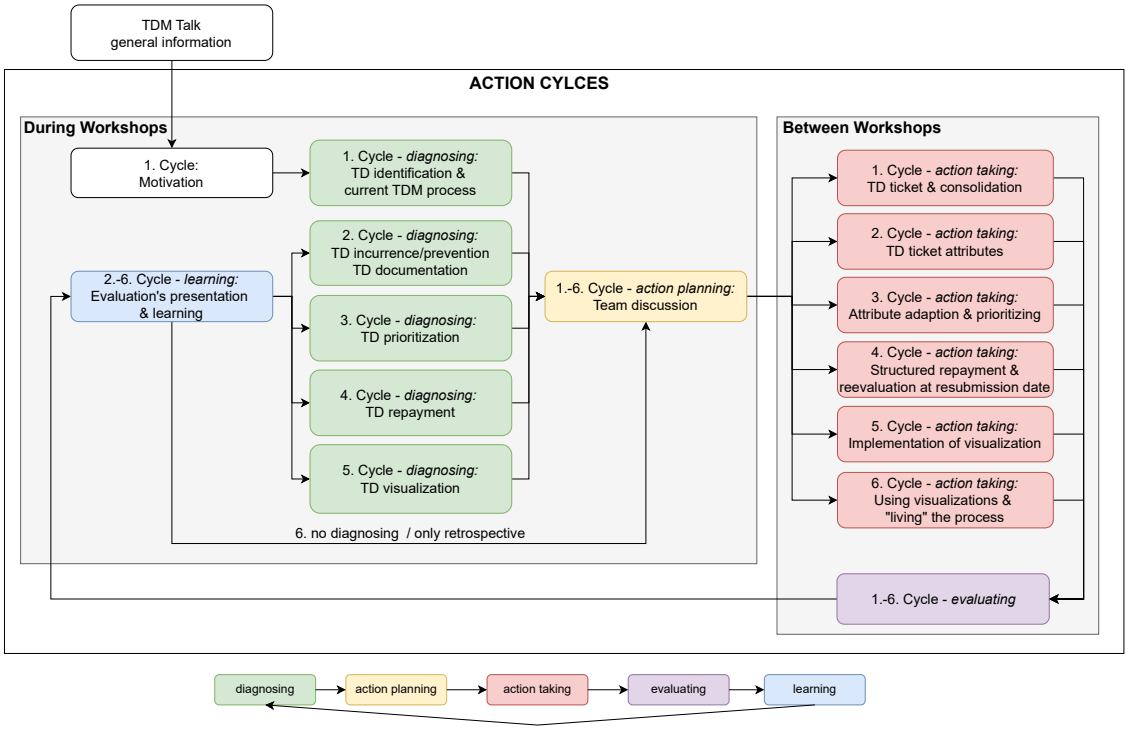


Figure 1: Study design presenting the action cycles [68]:
The workshops are presented in the left grey area and include learning, diagnosing, and action planning. The main actions taken and evaluation between the workshops are shown in the right grey area.

Participant	Team	Role	Years in IT	Gender	TD manager
P1	A	Manager	3-5	f	
P2	A	Software Architect	>10	m	x
P3	A	Engineer	6-10	m	
P4	A	Developer	0-2	m	
P5	A	Developer	6-10	f	
P6	A	Developer	15	m	
P7	A	Developer	>10	m	
P8	A	Quality Assurance	0-2	m	
P9	B	Product Owner	>10	m	
P10	B	Developer	>10	f	x
P11	B	Developer	>10	f	x
P12	B	Developer	>10	f	
P13	B	Developer	0-2	m	
P14	C	Product Owner	>10	m	
P15	C	Software Architect	>10	m	
P16	C	Developer	>10	m	x
P17	C	Developer	6-10	m	x
P18	C	Developer	>10	m	
P19	C	Developer	6-10?	m	
P20	C	Developer	3-5	m	

Table 3: Study Participants including the ones from the initial study of Wiese et al. [68]

The second workshop focused on TD documentation, i.e., creating a TD issue type with respective attributes, and TD prevention, i.e., avoiding TD by using quality attributes (QAs) for decision-making and debiasing decision-making by discussing alternatives, drawbacks, and risks. The third workshop focused on TD prioritization factors and a prioritization process. The fourth workshop focused on TD repayment strategies and when to use which strategy. During the fifth workshop on TD visualization, we discussed questions that should be answered by visualizing the TD issues. An overview of each presented approach is provided in Table 6, including the approaches developed within the initial study [68] marked with a "*".

2.1.3 Meetings. Including our first study, we attended 108 meetings, observing the teams for 96 hours. The teams decided which meetings the researchers were allowed to attend during the kickoff workshop, focusing on meetings involving TD-relevant decisions.

For Team A from our initial study, we participated in the *refinement*, and *review & planning* meetings, observing 40 meetings totaling 24.5 hours. *Refinement* meetings were used to analyze backlog issues, define an implementation approach, and estimate the effort and other relevant attributes required for each issue. During the *review & planning* meeting, the team reviewed the success of the former sprint (*review*) and organized its work, i.e., they prioritized the issues for the upcoming sprint (*planning*). Team A recorded the meetings for us to analyze them asynchronously.

For Team B, we participated in the *refinement*, *planning*, and *concept meetings*, observing 57 meetings totaling 53.5 hours. The *refinement* and *planning* meetings were similar to the respective

meetings from Team A, without reviewing the former sprint. The team further utilized a *concept meeting* to analyze more complex issues. This team did not want their meetings to be recorded, so we had to participate in all meetings synchronously.

For Team C, we participated in the *refinement* meetings, which were similar to those of Team A and B, observing 11 meetings totaling 18 hours. We did not attend the *planning* meetings as the team did not allow us to. This team recorded their *refinement* meetings for us.

2.2 Data collection

In this section, we present our data collection approaches, which comprise the action research itself, an awareness measurement approach adopted from the psychology domain, our meeting observations, and the self-assessment surveys.

2.2.1 Action Research. All teams conducted 5 workshops and at least one retrospective (see Table 4). We used a study protocol [69] to note down all actions and discussion results for each workshop. Additionally, the actions are captured on the last slide of each workshop slide deck [69], defining process changes and the associated tasks to be performed before the next workshop. In addition to workshops and meetings, we communicated with the team via email and chat to clarify specific situations and challenges.

2.2.2 TD-SAGAT – Situation Awareness Global Assessment Technique. Situation awareness is a concept from psychology that was proposed by Endsley [21] in 1988. As defined by our initial study, we adopted Endsley's definition for TD awareness as follows: *TD awareness is the (1) perception of the prerequisites (e.g., alternatives, benefits, drawbacks) of a decision (e.g., incurring TD) during decision-making situations (e.g., refinement meetings), (2) the comprehension of these prerequisites' meaning, and (3) the projection of their status in the future (e.g., TD's consequences).* Through the workshops, participants obtain a primary perception of the prerequisites (1). The action research aims to analyze if participants comprehend those prerequisites (2) and if they are able to apply them to project future effects (3) in decision-making situations.

Endsley defined the *Situation Awareness Global Assessment Technique (SAGAT)* to measure *situation awareness*, which we adopted and named TD-SAGAT. To identify the relevant prerequisites, the SAGAT approach instructs us to identify the main goals and subgoals for the situation, and the decisions that are necessary to meet the (sub)goal. The situation awareness prerequisites are then derived for these decisions, leading to the queries for a questionnaire by asking whether the participants considered the prerequisites [20]. In our previous study, we identified the following prerequisites for TD decisions: (1) solution alternatives, (2) solution's benefits, (3) solution's drawbacks, (4) solution's risks, (5) effort for implementing a solution, (6) TD item's principal, (7) TD item's interest, (8) component affected by TD, (9) functional requirement potentially competing with a TD item, and (10) quality attributes affected by TD. During this study, we recognized that the ninth prerequisite led to multiple follow-up questions and was often not understood. We removed this prerequisite from our analysis as the data validity was impacted by these possible misunderstandings.

The idea behind SAGAT is to interrupt participants in their work or a simulation of their work and ask them questions about their situation [20]. We employed the TD-SAGAT approach and interrupted the participants at two to three randomly selected time points, approximately every two to three meetings. Then, we asked the participants to complete the TD-SAGAT questionnaire, in which they were asked whether they considered the (randomly ordered) prerequisites during the previous issue's discussion with 'yes' or 'no' as answer options (see Table 5). Overall, we used the TD-SAGAT interruptions during 28 meetings, gaining 240 datasets (see Table 4).

2.2.3 Observations. We used the same structured observation protocol as proposed in our initial study [68] for all teams. As the meetings were structured around the backlog issues, we recorded our observations per issue, capturing: (1) basic information such as date, issue ID, and title, (2) the issue type (TD or not TD) as recorded by the team, as well as our assessment, (3) whether the researchers identified a risk for potential TD incurrence, (4) whether the TD-SAGAT prerequisites were mentioned, and (5) remarks. As remarks, we recorded why we assessed differences for (2) and the details for (3), as well as noticeable events we wanted to discuss with the participants during the upcoming workshop. Of all meetings, 24 were observed by two researchers who later discussed differences in their assessments to debias the results. Table 4 provides an overview, per Team, of the study duration, workshops/action cycles, meetings/observations, and TD-SAGAT interruptions.

2.2.4 Workshop Questionnaires. We created two separate questionnaires: one for the kickoff and another for the subsequent workshops. The kickoff questionnaire differed from the other questionnaires, as we collected basic information on the participants and the current TDM process of the teams. In contrast, the follow-up questionnaires included questions to identify the termination point of the action research, i.e., about the participants' satisfaction with the backlog and with the TDM process. Finally, we solicited feedback on the workshops and the study development to refine the workshop approach. To measure TD awareness, all questionnaires included nine questions about the TD-SAGAT requirements. For this, we asked participants if they usually considered those requirements during decision-making since the last workshop, in a 'yes' or 'no' format. This is less reliable than a direct TD-SAGAT intervention, but provides us with the participants' self-assessment for method triangulation. For space reasons, the complete list of all questions, their types, and purpose is only offered in the initial study [68] and our additional material [69].

2.3 Data Analysis

We present the data analysis aligned with our RQs.

2.3.1 Adopted Approaches (RQ1). The used TDM approaches are directly derived from the actions taken between the workshops or retrospectives. We focus our evaluation on the final TDM process of each team and do not discuss the course of adopting and dismissing approaches. We created an Excel sheet based on the notes in our study protocol, the workshop slides, and our observation protocols. We noted down all approaches and indicated for each team whether it was still in use by the end of the study (see [69]). We added remarks to the adoption, e.g., to clarify how the approach was

							Meetings				TD-SAGAT	
Team	Months	Study Begin	End	Work-shops	Retro-spectives	Action cycles	Cnt.	Hrs	Discussed issues	Observed by 2 researchers	in <x> meetings	No. of answers
A	16	2023-05	2024-08	5	1	5	40	24.5	488	5	12	90
B	16	2024-06	2025-09	5	2	6	57	53.5	507	14	10	57
C	11	2024-12	2025-10	5	1	5	11	18	137	5	6	93
Sum	30 (net 202305–202510; 43 gross)			15	4	16	108	96	1132	24	28	240

Table 4: Data collection per Team

No.	Question	Sub-Question or answer-options if applicable	Answer type
1	Regarding the topic currently	what alternatives are there to the solutions discussed?	y/n
2	under discussion, I considered	what are the advantages of the solutions discussed?	y/n
3	... (Please answer	what are the disadvantages of the solutions discussed?	y/n
4	spontaneously! If you have to	what are the risks of the solutions discussed?	y/n
5	think long and hard about	how much effort is required for the solutions discussed?	y/n
6	whether you have (just)	how much effort (if any) will it take to convert the discussed solutions into the ideal solution (repayment of TS)?	y/n
7	thought about something, the	how high, if any, is the (additional) effort required to permanently maintain the solutions discussed? (Interest from TS)	y/n
8	answer is probably – No)	which components are affected by the solutions.	y/n
9		which quality attributes are influenced.	y/n

Table 5: Questions from the TD-SAGAT questionnaire, and their answer types. Similar questions were element of the kickoff and progress questionnaire which are outlined in [69])

adopted or if the team wished to adopt the approach but was unable to do so due to technical problems.

2.3.2 Emerging Challenges (RQ2). We identified and discussed challenges in four different situations: First, we observed the participants during their meetings, utilizing an observation protocol as explained above. Second, for some challenges, we used a chat to directly discuss the challenges with the team and receive instant feedback. Third, we used the feedback from the workshop surveys to identify challenges. Fourth, we observed challenges discussed during the workshops. From the collected data, we created a final overview in the form of an Excel sheet (see [69]), which describes each challenge, the team affected by this challenge, and how and whether it was resolved. In this paper, we focus on the challenges that arose in several teams or that led to valuable solutions.

2.3.3 Effects on TD Awareness (RQ3). To analyze the teams' TD awareness, we (1) observed the team's discussions (see Section 2.2.3), (2) gathered more data with the TD-SAGAT survey [68] (Section 2.2.2), and (3) collected the participants' self-assessments through surveys completed during the workshops (see Section 2.2.4). With all three methods, we collected the same data points by asking or looking for the TD requirements as derived from the TD-SAGAT method.

Based on the observation protocols, we counted the number of backlog issues for which the respective prerequisite was discussed per meeting (*observation count*) and the number of all backlog issues (*backlog issues count*). We calculated the percentage by dividing the *observation count* by the *backlog issue count*. Regarding the TD-SAGAT results, we counted for each prerequisite how often it was considered by any of the participants. This means we summarized their 'yes' answers across all participants and interruptions (i.e., backlog issues) per meeting (*TD-SAGAT count*). We further counted the number of all answers we got in a meeting (*meeting's issue count*). We then calculated the percentage by dividing the *TD-SAGAT count* by the *meeting's issue count*. From the questionnaire results, we counted the 'yes' answers for each prerequisite in the workshop surveys (*survey count*) and the number of all answers per workshop (*workshop answers*). For each prerequisite, we calculated the percentage by dividing the *survey count* by the *workshop answers*.

The three action research studies were performed at different points in time for each team. To build a common timeline across all teams, we had to focus on the results between the respective workshops/retrospectives. For observations and TD-SAGAT, we summarized the values of all meetings between two respective workshops. For the workshop questionnaires, the participants assessed the period from the previous action cycle. As we also conducted a survey during the kick-off workshop, we gathered survey data for the period preceding the observations and the start of TD-SAGAT, leading to a period "0. - 1. Workshop" in the presentation of the results. By creating a common timeline, we were able to determine whether the overall awareness of each prerequisite increased during the action cycles.

3 RESULTS

In contrast to the previous study, this paper focuses on identifying differences and commonalities between the teams to distinguish transferable from context-specific approaches. We present the approaches in Section 3.1. Section 3.2 presents common challenges across companies to inform future research directions. Section 3.3 provides further insights into the development of TD awareness when establishing a TDM process and supports initial research findings. Crefsec:Results-awareness provides insights into developing TD awareness by establishing a TDM process and supports initial research findings.

3.1 Adopted approaches (RQ1)

In Table 6, we provide an overview of all research approaches presented to the teams, along with their adoption status. The approaches are structured according to the TD activities defined by Li et al. [39]. The remark column clarifies how the approaches were adopted and signals if an approach was newly developed by a team.

TD activity	Presented approaches	Adopted by A	B	C	Remark
(1) TD communication	**establishing a TD champion** [33]	y	y	y	A+B: Called *TD manager*
(2) TD identification & definition	Automatically by static analysis tools	(y)	(y)	(y)	Only as supporting method
	Manually by Dagstuhl definition [4]	(y)	n	n	A: But other TD causes exist
	Manually by Kruchten et al.'s backlog colors [37]	n	y	n	A+C: No external customers exist
	Manually by identifying whether TD interest occurs [5]	y	y	y	
	Manually by identifying who is willing to pay	y	y	y	**New approach**
(3) TD documentation	**Recording TD in the existing backlog** [35, 36]	y	y	y	
	Consequences of TD [4, 66]	y	y	y	
	*Consequences of repaying TD/"breaking change" check box [68]	y	y	n	
	Interest, interest probability [43, 55, 62]	y	y	y	
	Principal (effort) [43, 62]	y	y	y	A: In person days (PD), B: In PD and story points (SP), C: in SP
	Contagiousness [42]	y	y	y	
	Alternative, drawbacks, risks [11, 12]	y	y	y	
	Affected quality attributes (QAs) [7, 32]	n	y	n	
	***Resubmission date** [68]	y	y	y	
	"Pain factor" for the developers (annoyingness of the TD item)	n	n	y	**New approach**
(4) TD prevention	**Well-defined** requirements, code **standardization,** well-defined project processes, well-defined architecture/design, using **good practices**, testing, and reviews [26, 48, 54]	(y)	(y)	(y)	The teams generally optimized their work processes; not all strategies were followed
	Educational sessions on how to avoid TD [10]	n	n	n	No other sessions than workshops planned
	Incorporating TD repayment into project management [67]	n	n	n	
	Debiasing decision-making by listing at least one solution alternative, one drawback, and one risk per alternative [11, 12]	y	y	y	A+C: Additional attribute fields; B: Definition of Ready (DoR)
	Better decision-making by considering QAs [7, 32]	n	y	n	
	Counteract TD inheritance between mechatronic disciplines [18, 64]	n	n	n	
	*"Talked about TD" checkbox [68]	y	(y)	(y)	B: Part of DoR; C: Changed to: *Is it TD?* and *Does it incur TD?*
(5) TD measurement	**Using costs (interest, interest probability, principal) for prioritizing** [43, 55, 62]	y	y	(y)	To identify *low-hanging fruits*
	Using costs (interest, i.probability, principal) for decision-making [47]	y	n	n	
	*Calculating a return on investment (ROI) based on cost attributes [68]	(y)	y	n	A: To compare with an optimize the educated guess
	*Estimating an educated guess based on all attributes [68]	y	(y)	n	B: Attributes like QAs or resource availability might overrule ROI
	Mean attribute priority (all attributes on a 1-5 scale)	n	n	y	**New approach**
(6) TD prioritization	Prioritizing TD against functional requirements [38]	n	n	n	
	Prioritizing TD against each other [38]	y	y	y	
	Prioritizing once [38]	n	n	n	
	Prioritizing constantly [38]	y	y	y	By using a *re-submission date*
	Based on the systems evolution [55]	y	y	(y)	C: Not possible due to tool limitations
	Based on the business value [51]	n	n	n	
	Based on cost attributes (TD measurements) [62]	y	y	y	A+B: To identify *low-hanging fruits*; C: Included in mean calculation
	Based on the affected QAs [49]	n	y	n	B: As additional factor
(7) TD repayment	**Repay TD if it impedes a functional requirement** [27]	y	y	y	similar to refactor if it impedes development
	Pay the interest [14]	y	y	y	Use *re-submission date* to recall TD items
	Wait for discontinuing or migrating of the affected software system [55, 56]	y	y	y	Use *re-submission date* to recall TD items
	Repay TD if the positive effect is proven by analyzing the costs [43, 62]	y	y	(y)	repay *low-hanging fruits* if time allows; C: Not possible due to tool limitations
	Repay TD by assigning a quota [43, 67]	n	y	n	
	Repay if TD might hinder the system's evolution in the near future [55]	y	y	(y)	C: Not possible due to tool limitations
	Repay if TD was incurred due to a project deadline [67]	n	n	y	Team C planned to use it, but a fitting situation didn't emerge during the study time
(8) TD monitoring & visualization	DebtFlag [31], MultiDimEr [59], VisminerTD [44], MIND [23], TEDMA [24]	n	n	n	Not for backlog visualization, or only for self-developed tools
	Self-developed, externalized TD visualization (e.g., Microsoft PowerBI)	y	y	(y)	Interfaces to Azure DevOps and Jira exist; C: not possible due to tool limitations

Table 6: Presented approaches with references and their adoption/use by each of the teams in the final TDM process.
(bold – Approach adopted by all teams; * – Approach developed in the initial study; (y) – Approach partly adopted (see remark column))

We marked all approaches in bold that were chosen by all teams, and thereby might serve as a starting point for "best practices."

The ***best practices*** we derived from our study and which we present in the TDM Guide are: (1) Each team appointed a TD champion/manager to ensure adherence to the process and facilitate communication on TD topics. (2) The teams identified TD mostly by considering who "suffers" from the TD item and who would pay for its repayment. The teams used automatic TD identification solely as a supporting method. (3) The teams used their backlog, i.e., issue-tracking tool, to tag TD items. They created a specific TD issue type or template with various attributes to record their TD items. Common attributes for the TD item are the TD's *interest*,

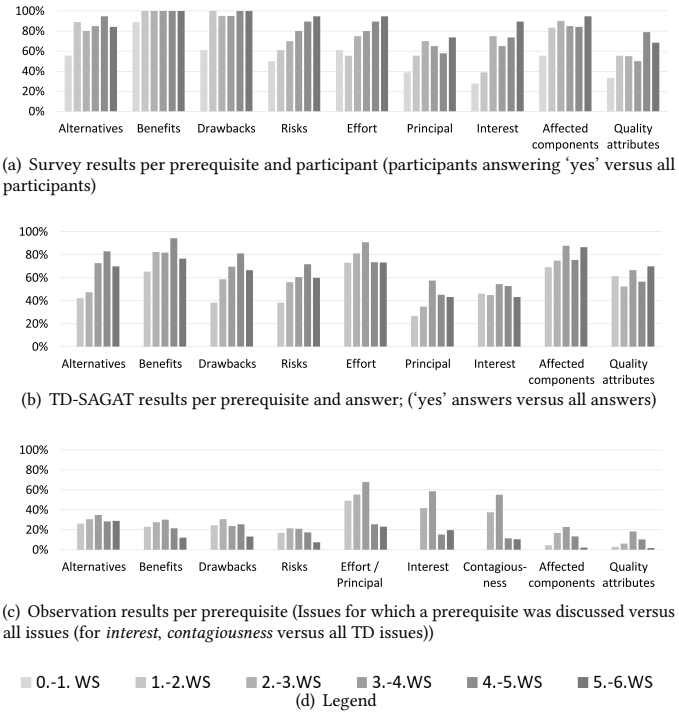
interest probability, *principal*, and *contagiousness*. For an ongoing and structured (re-)prioritization, they used a *resubmission date*. The teams measured TD-related attributes differently (e.g., a "very low" to "very high" scale or a 1 to 5 scale), but provided similar explanations to categorize the genuine values. The teams categorized the *interest* as: *< 15 min.*, *< 1hr.*, *< 4 hrs.*, *< 8 hrs.*, and *> 8 hrs.* They categorized *interest probability* as: *once a day*, *once a week*, *once a month*, *once a quarter*, *once a year*. The *contagiousness* was categorized with three values: the principal *decreases*, *remains the same*, or *increases*. (4) To avoid TD, the teams added attributes to all backlog item types, reminding them to discuss alternatives, drawbacks, and risks of an implementation approach. Various good practices to avoid TD were already part of the team's regular work processes. Some teams refined these processes, e.g., by modifying meeting structures, or utilizing "Scrum Poker." (5) For measuring the TD items and calculating a prioritization of the TD items, all teams used a different approach: An "educated guess" based on all attributes, a "return on investment" calculation based on *interest*, *interest probability*, and *effort*, and a "mean attribute priority" as a mean of all attributes measured on a one-to-five scale. (6) The teams agreed to prioritize TD against each other for ranking, avoiding competition with functional requirements. They also decided to continuously (re-)prioritize TD items based on the system's evolution using the *resubmission date*. (7) For the repayment of TD, all teams adopted multiple approaches depending on the situation. Commonly used approaches include paying the interest (high effort, low priority), repaying low-hanging-fruits (low effort, high priority), repaying TD items in components that will be changed soon, or combining a TD repayment with the implementation of functional requirements in the same component (system evolution). (8) For visualization purposes, no issue-tracking system provided adequate support. All visualizations of the backlog items' data were externalized to other systems. However, similar visualizations were chosen (see [69]).

3.2 Emerging Challenges (RQ2)

We describe the challenges that emerged in all or most of the teams in Table 7. We explain the proposed or adopted resolution we determined and executed during our study for each challenge. The overview also explicates suggestions, i.e., improvement measures from the authors' perspectives. The additional material [69] comprises an overview of all challenges, including team-specific ones.

3.3 Effects on TD awareness (RQ3)

We present the results of our awareness analysis in Figure 2. The **self-assessment** in Figure 2(a) reveals a steady increase in all prerequisites. This is to be expected, as self-assessments are usually less reliable and prone to the bias of "hypothesis guessing," i.e., providing the answer that researchers expect to receive. The exception to this rule is the decrease in alternatives between the last workshops, which aligns with the challenge of forgetting to use the template. The **TD-SAGAT results** in Figure 2(b) also show a steady increase. However, this is on a lower level, indicating that there are situations in which the developers choose not to consider certain attributes. The exceptions to the steady increase are the effort and principal. The effort was initially the highest, and the principal is essentially equivalent to the effort of a TD issue. Considering all



(a) Survey results per prerequisite and participant (participants answering 'yes' versus all participants)

(b) TD-SAGAT results per prerequisite and answer; ('yes' answers versus all answers)

(c) Observation results per prerequisite (Issues for which a prerequisite was discussed versus all issues (for *interest*, *contagiousness* versus all TD issues))

(d) Legend

Figure 2: Prerequisites' results measured by a survey, TD-SAGAT surveys, and observations between the workshops (WS)

other prerequisites may have reduced the focus on this initially most important topic. The **observation results** in Figure 2(c) show that not all prerequisites considered by the participants were actually discussed. Participants stated that they had considered most prerequisites but rarely discussed all of them. For the comparison prerequisites (alternatives, benefits, drawbacks, and risks), we observe an expected increase following the second workshop, which included information on cognitive biases. The peak for most of the prerequisites occurred between the third and fourth workshops. Prior to the fourth workshop, the TD issues were primarily refined for the first time in an initial endeavor. After this, only a few new TD issues arose, but many were rediscussed due to reaching their *resubmission date*. This means the TD issues were given a new resubmission date, but principal, interest, or contagiousness were not (re-)discussed, leading to a decline in those values. Additionally, the decrease after the fourth workshop aligns with the challenge of forgetting to use the template. As *affected components* and *quality attributes* were not recorded by all teams, their values tend to be lower than those of the other prerequisites.

4 DISCUSSION

In this section, we uncover research opportunities, offer support for practitioners, and propose improvements for issue-tracking tools.

	Challenge description	Emerged A	B	C	Challenge description and solution	Suggestion
Identification	Getting started: Refining all (old) TD issues	n	y	n	The TD managers and researchers tagged all TD items in the backlog, and teams held extra meetings to refine them. B: Due to the vast number of TD items, they only tagged the items created in the last year. Each team member, then, refined 2-3 issues weekly and set the *resubmission date* to present them to all developers in subsequent refinements.	Use the Team B method if there are too many open issues in the backlog.
	TD resulting from corporate policy or IT policy forcing changes	y	y	y	All teams were unsure how to deal with TD of the company or IT department, which they do not directly suffer from (e.g., switching to a better, company-wide unified issue-tracking tool). Team A + B marked those (mostly) as TD while Team C did not because they did not feel comfortable estimating the cost factors (*interest*, principal, etc.).	Further research is needed to determine TD items at different stakeholder levels and to explore effective ways to address them.
	Identifying TD issues	y	y	y	Even with an approach of asking who is 'suffering' from the TD item and who would be willing to pay for its repayment, teams were still struggling at times. Team C suggested creating a decision-flowchart and a list of 'TD smells,' indicators in an issue and its discussion that might signify it as a TD item (e.g., specific terms in the title or a developer introducing the issue instead of a product owner). We created both as part of the TDM Guide [65, 69].	Further research should improve the smell list.
	Remembering to identify TD	n	y	y	Team B+C forgot to mark new issues as TD and use the appropriate template. They added a reminder to consider whether an issue is actually a TD item to the template for all issues/DoR.	Adding this reminder to the template from the start.
Prevention	Uncovering alternatives	n	y	n	Team A+C: The architect led the discussion on alternatives. Team B+C: The team supported the uncovering by changing the discussion order: They started by identifying affected QAs, then derived drawbacks and risks for those QAs to identify the room for improvement, and by this, uncovered alternatives.	Further research on supporting the uncovering of solution alternatives on the developer (instead of the architect) level.
	Debiasing decisions in non-TD items or existing items	y	y	y	We had to (repeatedly) remind the team to assess alternatives (+drawbacks/risks) for all issues (not just TD issues and new issues). Furthermore, we had to instruct the teams to discuss drawbacks and risks even if no alternative existed.	Better incorporation into the workshop concept. External support is necessary.
Documentation & Measurement	Formulating substan- tial TD consequences	y	y	y	We had to (repeatedly) explain the method of using "the five whys" [58] to avoid lapidary statements like *'The TD item impacts maintainability.'*, which do not describe the core problem to business stakeholders.	Better incorporation into the workshop concept. External support is necessary.
	Defining architectural components to use in evolution-based repayment	y	y	y	We had to (repeatedly) explain the difference between component and cross-cutting concerns to generic components like "documentation", "test", or "feature flag removal." The evolutionary repayment method can only work if teams are specific when assessing the component.	Better incorporation into the workshop concept.
	Understanding and using TD-specific terms, esp. *interest probability* or *contagiousness*	y	y	y	All teams renamed "probability" to "frequency" and used *once a week*, *once a month*, etc., as values to better understand the meaning of *interest probability*. Additionally, all teams explained the fields in the template by adding underlying questions, e.g., "Will the repayment effort rise over time?" for *contagiousness*.	Further research on the difference between *interest probability* and frequency, and how to address both. Researching names that are understood by developers.
	Addressing multiple *interests* (e.g., additional effort for the team + risk of bugs + security risks).	y	y	y	We specifically discussed this with Team B+C. The teams decided to explain all possible *interests* in the "Consequences of the TD item" attribute, but to evaluate only the most critical one in terms of *interest* and *interest probability*.	Further research on dealing with multiple *interests*.
	Avoiding the central tendency bias for scaled values (e.g., 1-5, very low - very high)	y	n	n	Team A fell for the central tendency bias. Team B+C used categorized real values (e.g., 0-15 Min./15-60 Min./1-4 hrs./4-8 hrs./>8hrs.) for *interest* instead of a "very low to very high" scale. As a side effect of this decision, they avoided this bias.	Use categorized values at least as an additional explanation.
Repayment	Integrating a recurring task like feature flag removal or updating 3rd party libraries	n	y	n	Team B had the issue of regularly updating 3rd party libraries and removing feature flags. Due to the large number of them, they did not want to create TD issues for each library/feature flag. They adopted the procedure: Determine the most pressing removal or update, create an issue, and assess the attributes for this specific item. After this item is repaid, create a new issue for the next most pressing item.	Other teams might adopt this approach.
Monitoring & Visualizing	Adapting the issue-tracking tool	y	y	y	All three issue-tracking tools lacked helpful visualization support, necessitating the use of an additional tool for visualization (e.g., Power BI[1]). The TDM needs more tool maintenance and isn't well-aligned with the team's processes. Using two tools for managing TD raises the chance that the process/visualization won't be utilized effectively.	Issue-tracking tool vendors should support the visualization of issues more effectively.
	Implementing the required visualization of issues over time (e.g., *interest burden* per time period)	y	y	y	To identify the number of open issues for the previous time periods (over time), an additional code fragment with a loop over months that evaluates open and closed dates per item is necessary. Since existing tools do not support this, a third component would be required.	Issue-tracking tool vendors should support the visualization of issues more effectively.

Table 7: Observed challenges, in which team(s) they emerged, explanations or solutions, and suggestions on addressing them

4.1 Research

Concrete support. During our study, we identified multiple instances in which theoretical concepts of TD were too abstract for practitioners to understand or adopt. For example, the concepts of *interest probability* or *contagiousness*, while comprehensible during a talk, were fully understood only when we added a guiding question and renamed *interest probability* to *interest frequency*. Another example is the definition of an attribute's content. While the concepts of *interest* and *interest probability* were generally understood, applying them in practice required participants to discuss and define the content of these attributes, i.e., the combo boxes. Overall,

we suggest adding more specific support instructions for practitioners, e.g., as enumerated application instructions in the additional material of research papers that claim to benefit practitioners. To adhere to this rule, we added a detailed TDM guide that outlines the TDM process and how to implement it for practitioners to our additional material [69] and published it on Arxiv [65].

Research opportunities arising from the challenges. We identified research opportunities arising from the challenges we observed (see Section 3.2). We pinpointed the topic of unclear TD identification across multiple stakeholders. In a corporate context, an issue might be a TD item for one stakeholder, but might not be classified as TD from another stakeholder's perspective. Following practitioners' suggestions, we created a matrix and a flowchart for TD identification, along with a list of TD indicators, as part of the TDM Guide [65, 69]. We further suggested conducting more research on the debiasing of decisions at a lower level of abstraction than architecture. Specifically, we should develop concepts for supporting the identification of alternative solutions. Finally, we encourage the research community to discuss how to prioritize multiple *interests*.

Key Performance Indicators for Priorization. In this paper, we presented two calculated key performance indicators (KPIs), the ROI and the mean priority value, in addition to the *educated guess* approach used by Team A. Because all teams employed different approaches, we were unable to identify a best practice. The ROI might enable the generation of helpful visualizations, e.g., the overall *interest burden* per month. This was a specific request from participants from the management perspective. However, tool limitations restricted us from creating such a visualization (see Section 4.3). In contrast, the mean prioritization was easy to use and applicable even when the tool prevented us from creating attribute fields. Further research might encompass identifying which KPIs are indeed best practices in terms of context and company-wide aggregation.

4.2 Practitioners

Establishing a TDM process. Our overall research provides two approaches to establishing a TDM process. First, the workshop concept itself proved to be a valuable approach for establishing a TDM process. The primary objective of the workshops is to establish a TDM process iteratively, focusing on each area of the TDM process separately, and allowing for a process tailored to the context. Second, we created a white paper outlining a TDM process based on the approaches adopted in this study and solutions to the identified challenges. Establishing the process as described in the white paper is less time-intensive and allows for a more standardized approach.

TD awareness. As part of our research, we aimed to evaluate whether a TDM process increases overall TD awareness, as measured using the TD-SAGAT prerequisites. We demonstrated that adding the attributes to the respective backlog items sustainably increased awareness of single prerequisites. Educational workshops raise awareness only immediately after the workshop and can serve only as a first step. This became particularly evident when all teams initially overlooked evaluating alternatives, drawbacks, and risks for non-TD items. Only after adding the attribute fields to the non-TD issue types did the team's awareness of the potential for TD incurrence increase. The challenges and decreased values in Figure 2 indicate that a verbal reminder from the researchers seemed necessary at the beginning of the process to internalize its use.

Workshop critique and improvement potential. We recognized that the five workshops varied in the complexity of their content. While the second workshop was filled with two different TD activities (documentation & prevention), the third and fourth workshops were single-focused (prioritization and repayment). As TD prioritization and repayment are closely tied, it may be reasonable to combine both topics in a single workshop to reduce the number of invested person-days to establish the process. Overall, the workshop concept is not yet scalable for big companies. Conducting workshops with all the teams within a company would be expensive. Additionally, the workshop approach would allow each team to develop an individual process, making a cross-company evaluation impossible. Consequently, it is essential to identify strategies to enhance the approach's scalability. One method would be to conduct the workshops with one or two representative teams, and then allow other teams to adopt the approach, utilizing the developed issue type and visualizations. Finally, an alternative would be not to conduct workshops with future teams at all, but to adopt the process as presented in this work (Section 3.1 and our guide [65, 69]). However, in the questionnaires' open-ended questions, the participants emphasized the value of having an expert present at their refinement meetings: *"Looking at the actual tickets [issues] together helped me the most. If no support is provided at this point, I believe that the greatest added value of the workshops is lost."* Similarly, the evaluation of the TD awareness prerequisites over time demonstrated the need for an external expert. Thus, another option is to use a predefined process and refine and establish it with the assistance of an expert.

4.3 Tool Vendors

In Section 3.2, we uncovered problems that are specifically related to the tool vendors. First, this highlights the importance of selecting an appropriate issue-tracking tool, not only for TDM but also for organizing projects, products, and features. Team C was unable to implement the desired processes and had to endure multiple workarounds, prompting a discussion about employing a more professional issue-tracking tool within the company.

Second, there is significant potential to enhance professional issue-tracking tools, particularly in their ability to support effective calculations and visualizations. While we were able to implement an ROI calculation in Jira, it proved extremely complicated due to the rule-based calculations. Additionally, the standard visualizations in the issue-tracking tools were inadequate for our purpose. In all teams, we had to use an additional visualization tool, which has several disadvantages. Besides the effort of implementing and maintaining this additional tool, we observed that teams shied away from or simply forgot to use it during their regular work.

Finally, some requested visualizations were not feasible even in the additional tools. For example, the participants wanted to visualize the overall *interest burden* (= *interest* * *interest probability*) for all open issues per month to analyze whether the TDM process is indeed successful. We presented the *interest burden* of issues opened or closed at a specific time. To assess the *interest burden* of open issues from a previous month, we needed to evaluate all

issues' opened and closed dates. Creating a timeline required looping through all past months and issues, which would have involved developing an additional program. We were unwilling to implement this, as our teams could not guarantee the additional long-term maintenance effort.

5 RELATED WORK

Two recent studies on TDM by Rios et al. [53] and Junior et al. [35] provided an overview of TDM research, proving a lack of research on creating a systematic and holistic TDM process. Similarly, the "Reframing TD manifesto" developed by Avgeriou et al. identified that TD research should *"manage TD in alignment with its context"*, and that *"a minimum viable TDM process shall be formulated"* [6]. Through our research, we fill this research gap by providing an approach for establishing a context-sensitive TDM process.

Case Studies and Action Research on TDM. Most research on establishing TDM processes in a practical environment has been conducted in the form of case studies [28, 40, 40, 50, 70], which have been tailored to the specific company. In contrast, the workshop approach employed in our research does not provide a finalized TDM process but rather a framework for developing one. In only a few instances, researchers performed action research [13, 16, 17, 46, 68, 71], which each focused on a few TD activities [13, 46, 71] or TD types [16, 17] and did not provide a holistic approach. We conducted three action research studies, all following the same structure, which, to the best of the authors' knowledge, has not been previously performed in action research.

TD Awareness. One separate focus of our study is the analysis of a TDM establishment on TD awareness. While some papers claimed that their TDM approach increased TD awareness, few provided evidence to support this claim [8, 19, 30, 41, 63, 67]. Researchers typically used self-assessments, i.e., questionnaires [9, 67] or concluded TD awareness from indicators derived by static analysis tools [15]. To the best of the authors' knowledge, no other study has employed method triangulation to assess TD awareness.

6 THREATS TO VALIDITY

We present threats to validity based on the typical threats to validity for action research provided by Staron [60].

Construct Validity. As we replicated our studies, the "mono-operation bias" is mitigated. However, we still only report on three action research projects from one country. The feasibility of the resulting TDM Guide [65, 69] in the long run will be revealed by their adoption across various teams. The 'interaction of testing and treatment' is part of the action research's goal, as researchers influence the participants' processes. However, the awareness metrics must be evaluated carefully, as our presence in some meetings may have influenced the participants' awareness. Our study may be influenced by "hypothesis guessing". To mitigate this, we divided the teams into action and reference teams, emphasizing the value of open criticism at the beginning of each workshop.

Internal Validity. The voluntary participation leads to a threat of "biased selection of subject." Thus, our results are limited to teams willing to invest in a TDM process.

External Validity. Staron describes the "reactive effects of experimental arrangements", which occur when researchers influence the context too much. We have been present in the team's meetings and were sometimes asked about the intricacies of the process, to give more direct feedback, or, on rare occasions, we actively intervened to remind the team of essential process-related commitments. As we see this as the most significant threat, we plan to introduce the process to additional teams using only the TDM guide for future work. In our context, the threat of "constructs, methods, and confounding factors" means that the workshop approach is limited to agile working teams, as we relied on team decision-making, limiting the results to teams that are empowered to make such decisions.

Conclusion Validity. We mitigated the single-observer bias by regularly conducting observations with two researchers, who then discussed their findings together afterward. A "low reliability of measures" may have influenced the TD-SAGAT results. We applied the TD-SAGAT questionnaire only once per month & team and received about three responses per participant & meeting. However, summarizing the results over all teams remedies this issue to some extent.

7 CONCLUSION

In this research, we replicated our five-step workshop approach [68] in two teams. We identified commonalities and challenges in establishing a TDM process and evaluated changes in the teams' awareness of TD using the TD-SAGAT prerequisites.

For practitioners, we contributed a white paper outlining the establishment of a TDM process. It provides an introduction to the TDM topic, guidelines for the TDM process, a step-by-step guide to establishing the process, a collection of common mistakes and questions with respective suggestions and answers, and an outlook on how to scale such a process at the company level.

For researchers, we identified various research opportunities relevant to practitioners, including TD identification issues, debiasing decision-making, identifying alternative solutions, and the need for TD-KPI research. We urge researchers to offer concrete guidance for practitioners and encourage collaboration with *issue-tracking system* vendors.

For Future Work, we plan to evaluate the TDM guide to identify if a TDM establishment can be performed without external support from researchers or consultants.

Acknowledgment and Funding:

We thank the teams for their participation, trust, and insights. The project on which this report is based was sponsored by the Federal Ministry of Research, Technology, and Space of Germany under the funding code 01IS24031. Responsibility for the content of this publication lies with the authors.

References

[1] 2025. GitLab |The most-comprehensive AI-powered DevSecOps platform. https://about.gitlab.com/
[2] 2025. Jira | Issue & Project Tracking Software | Atlassian. https://www.atlassian.com/software/jira
[3] 2026. Software Campus. https://softwarecampus.de/
[4] Paris Avgeriou, Philippe Kruchten, Ipek Ozkaya, and Carolyn Seaman. 2016. Managing Technical Debt in Software Engineering. *Dagstuhl Reports* 6, 4 (2016), 110–138. doi:10.4230/DagRep.6.4.110

[5] Paris Avgeriou, Ipek Ozkaya, Alexander Chatzigeorgiou, Marcus Ciolkowski, Neil A. Ernst, Ronald J. Koontz, Eltjo Poort, and Forrest Shull. 2023. Technical Debt Management: The Road Ahead for Successful Software Delivery. In *2023 IEEE/ACM International Conference on Software Engineering: Future of Software Engineering (ICSE-FoSE)*. 15–30. doi:10.1109/ICSE-FoSE59343.2023.00007

[6] Paris Avgeriou, Ipek Ozkaya, Heiko Koziolek, Zadia Codabux, and Neil Ernst. 2025. Manifesto from Dagstuhl Perspectives Workshop 24452 – Reframing Technical Debt. doi:10.48550/arXiv.2505.13009 arXiv:2505.13009 [cs].

[7] Len Bass, Paul Clements, and Rick Kazman. 2012. *Software Architecture in Practice: Software Architect Practice_c3*. Addison-Wesley.

[8] Olga Baysal, Reid Holmes, and Michael W. Godfrey. 2013. Developer dashboards: The need for qualitative analytics. *IEEE Software* 30, 4 (2013), 46–52. doi:10.1109/MS.2013.66

[9] Clara Berenguer, Adriano Borges, Sávio Freire, Nicolli Rios, Nebojsa Tausan, Robert Ramac, Boris Pérez, Camilo Castellanos, Darío Correal, Alexia Pacheco, Gustavo López, Davide Falessi, Carolyn Seaman, Vladimir Mandic, Clemente Izurieta, and Rodrigo Spínola. 2021. Technical Debt is not Only about Code and We Need to be Aware about It. 1–12. doi:10.1145/3493244.3493285

[10] Terese Besker, Antonio Martini, and Jan Bosch. 2020. Carrot and Stick Approaches When Managing Technical Debt. In *Proceedings of the 3rd International Conference on Technical Debt (TechDebt '20)*. Association for Computing Machinery, New York, NY, USA, 21–30. doi:10.1145/3387906.3388619

[11] Klara Borowa, Maria Jarek, Gabriela Mystkowska, Weronika Paszko, and Andrzej Zalewski. 2022. Debiasing Architectural Decision-Making: A Workshop-Based Training Approach. In *Software Architecture*, Ilias Gerostathopoulos, Grace Lewis, Thais Batista, and Tomáš Bureš (Eds.). Springer International Publishing, Cham, 159–166. doi:10.1007/978-3-031-16697-6_11

[12] Klara Borowa, Andrzej Zalewski, and Szymon Kijas. 2021. The Influence of Cognitive Biases on Architectural Technical Debt. In *International Conference on Software Architecture (ICSA)*. 115–125. doi:10.1109/ICSA51549.2021.00019

[13] Nichlas Bødker Borup, Ann Louise Jul Christiansen, Sabine Hørdum Tovgaard, and John Stouby Persson. 2021. Deliberative Technical Debt Management: An Action Research Study. In *Software Business (Lecture Notes in Business Information Processing)*, Xiaofeng Wang, Antonio Martini, Anh Nguyen-Duc, and Viktoria Stray (Eds.). Springer International Publishing, Cham, 50–65. doi:10.1007/978-3-030-91983-2_5

[14] Frank Buschmann. 2011. To pay or not to pay technical debt. *IEEE Software* 28, 6 (Nov. 2011), 29–31. doi:10.1109/MS.2011.150

[15] Yania Crespo, Carlos López-Nozal, Raúl Marticorena-Sánchez, Margarita Gonzalo-Tasis, and Mario Piattini. 2022. The role of awareness and gamification on technical debt management. *Information and Software Technology* 150, April (2022), 106946. doi:10.1016/j.infsof.2022.106946

[16] Thober Detofeno, Andreia Malucelli, and Sheila Reinehr. 2022. PriorTD: A Method for Prioritization Technical Debt. In *Proceedings of the XXXVI Brazilian Symposium on Software Engineering (SBES '22)*. Association for Computing Machinery, New York, NY, USA, 230–240. doi:10.1145/3555228.3555238

[17] Thober Detofeno, Sheila Reinehr, and Malucelli Andreia. 2021. Technical Debt Guild: When experience and engagement improve Technical Debt Management. In *Proceedings of the XX Brazilian Symposium on Software Quality (SBQS '21)*. Association for Computing Machinery, New York, NY, USA, 1–10. doi:10.1145/3493244.3493271

[18] Quang Huan Dong, Felix Ocker, and Birgit Vogel-Heuser. 2019. Technical Debt as indicator for weaknesses in engineering of automated production systems. *Production Engineering* 13, 3-4 (2019), 273–282. doi:10.1007/s11740-019-00897-0 Publisher: Springer Berlin Heidelberg ISBN: 0123456789.

[19] Ulf Eliasson, Antonio Martini, Robert Kaufmann, and Sam Odeh. 2015. Identifying and visualizing Architectural Debt and its efficiency interest in the automotive domain: A case study. In *2015 IEEE 7th International Workshop on Managing Technical Debt (MTD)*. 33–40. doi:10.1109/MTD.2015.7332622

[20] M. Endsley. 2000. Direct Measurement of Situation Awareness: Validity and Use of SAGAT. (2000).

[21] Mica R. Endsley. 1988. Design and Evaluation for Situation Awareness Enhancement. *Proceedings of the Human Factors Society Annual Meeting* 32, 2 (1988), 97–101. doi:10.1177/154193128803200221

[22] Neil Ernst, Rick Kazman, and Julien Delange. 2021. *Technical Debt in Practice*. The MIT Press Cambridge, Massachusetts, London, England.

[23] Davide Falessi and Andreas Reichel. 2015. Towards an open-source tool for measuring and visualizing the interest of technical debt. In *2015 IEEE 7th International Workshop on Managing Technical Debt (MTD)*. 1–8. doi:10.1109/MTD.2015.7332618

[24] Carlos Fernández-Sánchez, Héctor Humanes, Juan Garbajosa, and Jessica Díaz. 2017. An Open Tool for Assisting in Technical Debt Management. In *2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. 400–403. doi:10.1109/SEAA.2017.60

[25] Markus Finke, Thomas Neff, and Tobias Reichl. 2023. How to introduce TD Management into a Software Development Process – A Practical Approach. In *2023 ACM/IEEE International Conference on Technical Debt (TechDebt)*. 52–61. doi:10.1109/TechDebt59074.2023.00013

[26] Sávio Freire, Nicolli Rios, Boris Gutierrez, Dario Torres, Manoel Mendonça, Clemente Izurieta, Carolyn Seaman, and Rodrigo O. Spinola. 2020. Surveying Software Practitioners on Technical Debt Payment Practices and Reasons for Not Paying off Debt Items. In *Proceedings of the Evaluation and Assessment in Software Engineering (EASE '20)*. Association for Computing Machinery, New York, NY, USA, 210–219. doi:10.1145/3383219.3383241

[27] Sávio Freire, Nicolli Rios, Boris Pérez, Camilo Castellanos, Darío Correal, Robert Ramač, Vladimir Mandić, Nebojša Taušan, Gustavo López, Alexia Pacheco, Manoel Mendonça, Davide Falessi, Clemente Izurieta, Carolyn Seaman, and Rodrigo Spínola. 2023. Software practitioners' point of view on technical debt payment. *Journal of Systems and Software* 196 (2023), 111554. doi:10.1016/j.jss.2022.111554

[28] Yuepu Guo, Carolyn Seaman, Fabio Q.B. da Silva, Fabio Q.B. da Silva, Fabio Q.B. da Silva, and Fabio Q.B. da Silva. 2016. Costs and obstacles encountered in technical debt management – A case study. *Journal of Systems and Software* 120 (2016), 156–169. doi:10.1016/j.jss.2016.07.008

[29] Yuepu Guo, Rodrigo Oliveira Spínola, and Carolyn Seaman. 2016. Exploring the costs of technical debt management – a case study. *Empirical Software Engineering* 21, 1 (2016), 159–182. doi:10.1007/s10664-014-9351-7

[30] Rajeev Kumar Gupta, Prabhulinga Manikreddy, Sandesh Naik, and K. Arya. 2016. Pragmatic approach for managing technical debt in legacy software project. In *ACM International Conference Proceeding Series*, Vol. 18-20-Febr. 170–176. doi:10.1145/2856636.2856655

[31] Johannes Holvitie and Ville Leppänen. 2013. DebtFlag: Technical debt management with a development environment integrated tool. In *2013 4th International Workshop on Managing Technical Debt (MTD)*. 20–27. doi:10.1109/MTD.2013.6608674

[32] ISO/IEC. 2023. ISO 25010-2023 | Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models. https://www.iso.org/obp/ui/en/#iso:std:iso-iec:25010:ed-2:v1:en

[33] Ciera Jaspan and Collin Green. 2023. Defining, Measuring, and Managing Technical Debt. *IEEE Software* 40, 3 (May 2023), 15–19. doi:10.1109/MS.2023.3242137 Conference Name: IEEE Software.

[34] Helvio Jeronimo Junior. 2023. *isTDM: AN EVIDENCE-BASED FRAMEWORK TO SUPPORT THE MANAGEMENT OF TECHNICAL DEBT IN SOFTWARE PROJECTS*. Ph. D. Dissertation. Universidade Federal do Rio de Janeiro.

[35] Helvio Jeronimo Junior and Guilherme Horta Travassos. 2022. Consolidating a common perspective on Technical Debt and its Management through a Tertiary Study. *Information and Software Technology* 149, October 2020 (2022), 106964. doi:10.1016/j.infsof.2022.106964 Publisher: Elsevier B.V..

[36] Philippe Kruchten, Robert Nord, and Ipek Ozkaya. 2019. *Managing Technical Debt: Reducing Friction in Software Development*. Software Engineering Institute, Carnegie Mellon University.

[37] Philippe Kruchten, Robert L. Nord, and Ipek Ozkaya. 2012. Technical debt: From metaphor to theory and practice. *IEEE Software* 29, 6 (Nov. 2012), 18–21. doi:10.1109/MS.2012.167

[38] Valentina Lenarduzzi, Terese Besker, Davide Taibi, Antonio Martini, and Francesca Arcelli Fontana. 2020. A systematic literature review on Technical Debt prioritization: Strategies, processes, factors, and tools (journal version). *The Journal ofSystems & Software journal* 171, 171 (2020). doi:10.1016/j.jss.2020.110827

[39] 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. doi:10.1016/j.jss.2014.12.027

[40] Somayeh Malakuti and Sergey Ostroumov. 2020. The quest for introducing technical debt management in a large-scale industrial company. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 12292 LNCS. 296–311. doi:10.1007/978-3-030-58923-3_20

[41] Antonio Martini, Terese Besker, and Jan Bosch. 2016. The Introduction of Technical Debt Tracking in Large Companies. In *2016 23rd Asia-Pacific Software Engineering Conference (APSEC)*. 161–168.

[42] Antonio Martini and Jan Bosch. 2015. The Danger of Architectural Technical Debt: Contagious Debt and Vicious Circles. In *Proceedings - 12th Working IEEE/IFIP Conference on Software Architecture, WICSA 2015*. 1–10. doi:10.1109/WICSA.2015.31

[43] Steve McConnell. 2008. Managing technical debt. *Construx Inc.* (2008).

[44] Thiago S. Mendes, Felipe G.S. Gomes, David P. Gonçalves, Manoel G. Mendonça, Renato L. Novais, and Rodrigo O. Spínola. 2019. VisminerTD: a tool for automatic identification and interactive monitoring of the evolution of technical debt items. *Journal of the Brazilian Computer Society* 25, 1 (2019). doi:10.1186/s13173-018-0083-1

[45] Microsoft. 2025. Azure DevOps Services | Microsoft Azure. https://azure.microsoft.com/en-us/products/devops

[46] Frederico Oliveira, Alfredo Goldman, and Viviane Santos. 2015. Managing Technical Debt in Software Projects Using Scrum: An Action Research. In *Proceedings - 2015 Agile Conference, Agile 2015*. 50–59. doi:10.1109/Agile.2015.7

[47] Judith Perera, Ewan Tempero, Yu-Cheng Tu, and Kelly Blincoe. 2024. A Systematic Mapping Study Exploring Quantification Approaches to Code, Design, and Architecture Technical Debt. *ACM Trans. Softw. Eng. Methodol.* (July 2024).

doi:10.1145/3675393 Just Accepted.

[48] Boris Pérez, Camilo Castellanos, Darío Correal, Nicolli Rios, Sávio Freire, Rodrigo Spínola, Carolyn Seaman, and Clemente Izurieta. 2021. Technical debt payment and prevention through the lenses of software architects. *Information and Software Technology* 140 (2021), 106692. doi:10.1016/j.infsof.2021.106692

[49] Paula Rachow and Matthias Riebisch. 2022. An architecture smell knowledge base for managing architecture technical debt. In *Proceedings of the International Conference on Technical Debt (TechDebt '22)*. Association for Computing Machinery, New York, NY, USA, 1–10. doi:10.1145/3524843.3528092

[50] Narayan Ramasubbu and Chris F. Kemerer. 2019. Integrating technical debt management and software quality management processes: A framework and field test. *Test Engineering and Management* 45, 7-8 (2019), 883. doi:10.1109/TSE.2017.2774832

[51] Rodrigo Reboucas De Almeida, Uirá Kulesza, Christoph Treude, D'Angellys Cavalcanti Feitosa, and Aliandro Higino Guedes Lima. 2018. Aligning technical debt prioritization with business objectives: A multiple-case study. *Proceedings - 2018 IEEE International Conference on Software Maintenance and Evolution, ICSME 2018* (2018), 655–664. arXiv:1807.05582

[52] Rodrigo Rebouças de Almeida, Rafael do Nascimento Ribeiro, Christoph Treude, and Uirá Kulesza. 2021. Business-Driven Technical Debt Prioritization - An Industrial Case Study. In *2021 IEEE/ACM International Conference on Technical Debt (TechDebt)*. 74–83. arXiv:2010.09711

[53] Nicolli Rios, Manoel Gomes de Mendonça Neto, and Rodrigo Oliveira Spínola. 2018. A tertiary study on technical debt: Types, management strategies, research trends, and base information for practitioners. *Information and Software Technology* 102 (2018), 117–145. doi:10.1016/j.infsof.2018.05.010

[54] Nicolli Rios, Rodrigo Oliveira Spínola, Manoel Mendonça, and Carolyn Seaman. 2020. The practitioners' point of view on the concept of technical debt and its causes and consequences: a design for a global family of industrial surveys and its first results from Brazil. *Empirical Software Engineering* 25, 5 (2020), 3216–3287. doi:10.1007/s10664-020-09832-9

[55] Klaus Schmid. 2013. A formal approach to technical debt decision making. *QoSA 2013 - Proceedings of the 9th International ACM Sigsoft Conference on the Quality of Software Architectures* (2013), 153–162. doi:10.1145/2465478.2465492 ISBN: 9781450321266.

[56] Klaus Schmid. 2013. On the Limits of the Technical Debt Metaphor: Some Guidance on Going Beyond. In *Fourth Workshop on Managing Technical Debt, Workshop at the International Conference on Software Engineering*. IEEE, 63–66. doi:10.1109/MTD.2013.6608681

[57] Carolyn Seaman and Yuepu Guo. 2011. Measuring and Monitoring Technical Debt. In *Advances in Computers*, Marvin V. Zelkowitz (Ed.). Vol. 82. Elsevier, 25–46. doi:10.1016/B978-0-12-385512-1.00002-5

[58] Olivier Serrat. 2017. The Five Whys Technique. In *Knowledge Solutions: Tools, Methods, and Approaches to Drive Organizational Performance*, Olivier Serrat (Ed.). Springer, Singapore, 307–310. doi:10.1007/978-981-10-0983-9_32

[59] Lakmal Silva, Michael Unterkalmsteiner, and Krzysztof Wnuk. 2022. MultiDimEr: a multi-dimensional bug analyzEr. In *Proceedings of the International Conference on Technical Debt (TechDebt '22)*. Association for Computing Machinery, New York, NY, USA, 66–70. doi:10.1145/3524843.3528099

[60] Miroslaw Staron. 2020. *Action Research in Software Engineering: Theory and Applications*. Springer International Publishing, Cham. doi:10.1007/978-3-030-32610-4

[61] Klaas Jan Stol and Brian Fitzgerald. 2018. The ABC of software engineering research. *ACM Transactions on Software Engineering and Methodology* 27, 3 (2018). doi:10.1145/3241743

[62] Edith Tom, Aybüke Aurum, and Richard Vidgen. 2013. An exploration of technical debt. *Journal of Systems and Software* 86, 6 (2013), 1498–1516. doi:10.1016/j.jss.2012.12.052

[63] Christoph Treude and Margaret Anne Storey. 2010. Awareness 2.0: Staying aware of projects, developers and tasks using dashboards and feeds. *Proceedings - International Conference on Software Engineering* 1 (2010), 365–374. doi:10.1145/1806799.1806854 ISBN: 9781605587196.

[64] Birgit Vogel-Heuser and Fandi Bi. 2020. Interdisciplinary effects of technical debt in companies with mechatronic products — a qualitative study. *Journal of Systems and Software* 171 (2020), 110809. doi:10.1016/j.jss.2020.110809

[65] Marion Wiese. 2026. A Practical Guide to Establishing Technical Debt Management. doi:abs/2601.11430

[66] Marion Wiese and Klara Borowa. 2023. IT managers' perspective on Technical Debt Management. *Journal of Systems and Software* 202 (April 2023), 111700. doi:10.1016/j.jss.2023.111700

[67] Marion Wiese, Paula Rachow, Matthias Riebisch, and Julian Schwarze. 2022. Preventing technical debt with the TAP framework for Technical Debt Aware Management. *Information and Software Technology* (2022), 106926. doi:10.1016/j.infsof.2022.106926

[68] Marion Wiese, Kamila Serwa, Anastasia Besier, Ariane S. Marion-Jetten, and Eva Bittner. 2026. Establishing technical debt management — A five-step workshop approach and an action research study. *Journal of Systems and Software* 231 (Jan. 2026), 112606. doi:10.1016/j.jss.2025.112606

[69] Marion Wiese, Kamila Serwa, and Eva Bittner. [n. d.]. Additional Material for A Practical Guide for Establishing a Technical Debt Management Process. https://doi.org/10.5281/zenodo.18325238

[70] Jesse Yli-Huumo, Andrey Maglyas, and Kari Smolander. 2016. How do software development teams manage technical debt? – An empirical study. *Journal of Systems and Software* 120 (2016), 195–218. doi:10.1109/TechDebt.2019.00009

[71] Jesse Yli-Huumo, Andrey Maglyas, Kari Smolander, Johan Haller, and Hannu Törnroos. 2016. Developing Processes to Increase Technical Debt Visibility and Manageability – An Action Research Study in Industry. In *Product-Focused Software Process Improvement*, Pekka Abrahamsson, Andreas Jedlitschka, Anh Nguyen Duc, Michael Felderer, Sousuke Amasaki, and Tommi Mikkonen (Eds.). Springer International Publishing, Cham, 368–378. doi:10.1007/978-3-319-49094-6_24

Chapter 17

Exploring Web Search Engines to Find Architectural Knowledge

8 *(AK in Web Search Engines)*	
Title	Exploring Web Search Engines to Find Architectural Knowledge
Authors	Soliman M., Wiese M., Li Y., Riebisch M., Avgeriou P.
Year	2021
Conference	IEEE 18th International Conference on Software Architecture (ICSA)
Reference	[Sol+21b]

TABLE 17.1: Publication No. 8 *(AK in Web Search Engines)*

2021 IEEE International Conference on Software Architecture (ICSA)

Exploring Web Search Engines to Find Architectural Knowledge

Mohamed Soliman*, Marion Wiese†, Yikun Li*, Matthias Riebisch†, and Paris Avgeriou*
* Bernoulli Institute for Mathematics, Computer Science and Artificial Intelligence
University of Groningen, Groningen, The Netherlands
{m.a.m.soliman, yikun.li, p.avgeriou}@rug.nl
†Department of Informatics, Universität Hamburg, Germany
Email: {wiese, riebisch}@informatik.uni-hamburg.de

***Abstract*—Software engineers need relevant and up-to-date architectural knowledge (AK), in order to make well-founded design decisions. However, finding such AK is quite challenging. One pragmatic approach is to search for AK on the web using traditional search engines (e.g. Google); this is common practice among software engineers. Still, we know very little about what AK is retrieved, from where, and how useful it is. In this paper, we conduct an empirical study with 53 software engineers, who used Google to make design decisions using the Attribute-Driven-Design method. Based on how the subjects assessed the nature and relevance of the retrieved results, we determined how effective web search engines are to find relevant architectural information. Moreover, we identified the different sources of AK on the web and their associated AK concepts.**

***Index Terms*—Architecture knowledge Architecture design decisions Search engines**

I. Introduction

Architectural knowledge (AK) is crucial for software engineers to make architectural design decisions [1]. For instance, knowledge about technologies or architectural patterns, including their benefits and drawbacks, is important to select an architectural solution for a design problem. However, finding architectural knowledge (AK) is a challenging task [2] for a number of reasons. First, AK resides in multiple heterogeneous *AK sources*, such as technology documentation [2], issue tracking systems [3], and developer communities (e.g. Stack Overflow) [4]. Thus there is no single source of AK that contains all required AK.

Second, each source of AK contains different *AK concepts* (e.g. design decisions, solution alternatives [5], or the benefits and drawbacks of architectural solutions [4]). For instance, developer communities contain predominantly general AK concepts [6], such as the benefits and drawbacks of architectural solutions [7]. In contrast, issue tracking systems contain mainly design and reasoning AK concepts [6], such as design decisions of existing systems [3]. Thus, depending on the AK concept, one may need to look into a different AK source.

Third, all AK sources are characterized by a fast pace of change, and accelerating technology churn [8]. This makes it even harder for software engineers to find and analyze information within the different sources of AK.

This work was supported by ITEA3 and RVO under grant agreement No. 17038 VISDOM (https://visdom-project.github.io/website).

One approach to facilitate finding AK is to manually capture AK (i.e. search for AK, and codify it) from multiple sources, and subsequently structure and store it in a repository (e.g. [9]). This supports software engineers to directly find relevant AK concepts, without navigating through many sources of AK depending on the concept. However, manually capturing AK requires significant efforts to gather and keep knowledge up to date; this makes it an expensive means in industrial practice [10].

A more pragmatic way to search for AK from different sources is to use web search engines (e.g. Google). Web search engines are commonly used by software engineers in their daily business to find technical solutions for problems [11]. Moreover, web search engines can provide access to multiple AK concepts, such as design decisions from an existing system (e.g. within open source systems [3]), as well as descriptions of architectural solutions (e.g. within technology documentation [2]). It is even possible to use web search engines to populate AK repositories [2].

While web search engines can provide support to find AK, recent experiences show that web search engines return many irrelevant results when searching for AK [2]. In fact, there is little to no empirical evidence about: a) which AK sources and AK concepts can actually be found by web search engines; and b) when and how web search engines can be helpful for practitioners.

Our main **goal** in this paper is to *explore which AK sources and AK concepts are retrieved by web search engines, and to gauge the effectiveness of web search engines to find relevant AK during the architectural design process*. To this end, we conducted an empirical study with 53 software engineers with different levels of experience. The subjects used the most popular web search engine (i.e. Google) to find relevant AK concepts when conducting the steps of the Attribute-Driven Design (ADD) method [12]; ADD was chosen as it is one of the most popular architectural design processes in literature. The study results in the following contributions:

- We empirically identified AK sources, that web search engines are able to find. Moreover, we associated each

978-1-7281-6260-7/21/$31.00 ©2021 IEEE
DOI 10.1109/ICSA51549.2021.00023

AK source with the AK concepts they contain, based on the perspective of software engineers. Furthermore, we determined possible correlations between AK concepts.
- We created a corpus of empirically classified and evaluated web pages and their respective AK sources and AK concepts.
- We measured and compared the effectiveness of web search engines to support software engineers during the execution of the ADD steps. Moreover, we determined the most relevant AK sources for each ADD step according to the evaluation of software engineers.
- We identified the AK concepts that make web pages highly relevant for software engineers when making design decisions.

The rest of the paper is structured as follows: Section II provides a background on the ADD steps, while Section III presents the research questions and study design. Sections IV, V, VI and VII present the results per research question, Section VIII discusses our results and their implications to practitioners and researchers, and Section IX discusses threats to validity. Finally, Section X discusses some related work, and Section XI concludes the paper.

II. Attribute Driven Design Steps

Kazman et al. [12] proposed a number of iterative steps within ADD to make architectural decisions. For the purposes of our study, we select three of these steps, which are the most *information-intensive* [13]: they require searching for architectural information to be conducted. We explain these three steps and the type of information that software engineers need to perform them.

Identify design concepts: In this step, alternative architectural solutions are identified for a design issue. For example, a software engineer might look for alternative broker technologies, which might fulfill system requirements and align with the system constraints. As an example, these alternative solutions for broker technologies could be RabbitMQ, Kafka and ActiveMQ. To perform this step, software engineers need to search for information regarding alternative solution options.

Select design concepts: In this step, one architectural solution is selected from a list of alternative solutions (from the previous step). This is done by comparing alternative solutions with each other regarding their ability to fulfill functional requirements and quality attributes. For example, RabbitMQ, Kafka and ActiveMQ are compared regarding performance and reliability to decide on the most suitable broker technology. To perform this step, software engineers need to search for benefits and drawbacks of the alternative options, e.g. information about performance benchmarks and reliability features for each of the alternative solutions.

Instantiate architecture elements: In this step, the selected architectural solution (from the previous step) is customized to match the system requirements. For instance, to achieve high availability, replication tactics need to be implemented. This is done by configuring the selected broker technology (e.g. Kafka) to add a specific number of replicated instances. To perform this step, software engineers need to search for information and experiences regarding technology features, and their abilities to implement architectural tactics [12].

III. Study design

A. Research questions

To achieve our goal (see Section I), we ask the following research questions (RQs):

(RQ1) *Which AK can web search engines support to find?*

- ***(RQ1.1)*** *Which AK sources can web search engines find?*
- ***(RQ1.2)*** *Which AK concepts are prominent within the found AK sources?*

As discussed in Section I, researchers have explored multiple different AK sources and their AK concepts. However, there is no comprehensive list of AK sources and AK concepts on the web. Thus, we ask RQ1.1 to determine possible sources of AK that exist on the web. Answering RQ1.1 can confirm the AK sources that are already known but it can also reveal new AK sources, which have not been previously explored. Moreover, we ask RQ1.2 to determine the AK concepts, which commonly appear in each of the found AK sources.

(RQ2) *Which AK concepts co-occur on the web?*

We ask RQ2 to determine possible relationships between AK concepts on the web. Answering RQ2 can help us determine if AK concepts appear together on the web similarly to their conceptual relationships in AK ontologies (e.g. architectural solutions have benefits and drawbacks [4]). This can support assessing whether the relations between concepts in existing AK ontologies are reflected on the AK found in the web.

(RQ3) *How well do web search engines support software engineers in following the ADD steps?*

- ***(RQ3.1)*** *How effective are web search engines to find AK needed for performing the ADD steps?*
- ***(RQ3.2)*** *Which AK sources have the biggest contribution on the effectiveness of web search engines for each of the ADD steps?*

Within each step of the ADD (see Section II), software engineers need to search for different types of architectural information. We ask RQ3.1 to quantitatively measure the effectiveness of web search engines to find relevant architectural information, and to determine how much web search engines can support software engineers during the different architectural design steps. Moreover, we ask RQ3.2 to determine the most useful sources of AK for each ADD step. Answering RQ3.2 can support prioritizing and directing our future research efforts to explore and capture AK, by focusing on certain AK sources, that yield the highest benefit to software engineers.

(RQ4) *Which AK concepts make web pages more relevant for design decisions?*

When using web search engines to perform architectural design tasks, some AK concepts may increase the relevance of the corresponding web pages. We ask RQ4 to determine which AK concepts indeed can make a web page more relevant for

TABLE I: Industrial experience of participants

Software development		Software architecture	
# Years	# Participants	# Years	# Participants
>10 Years	4	>5 Years	5
3-10 Years	13	2-5 Years	9
<3 Years	36	<1 Year	39

specific architectural tasks than others. Answering RQ4 could provide guidance on which AK is worth sharing on the web.

B. Overview on the research process

To answer the RQs, we conducted an exploratory case study [14] with 53 software engineers (see Section III-C) who used the most popular web search engine (i.e. Google) to perform the ADD steps (see Section II). The conducted case study is an embedded case study, where the ADD steps constitute our case and the executed search queries from the 53 software engineers are the units of analysis.

To collect data, we asked the 53 software engineers to solve six architectural design searching tasks (see Section III-D) using Google, where each searching task performs one of the three ADD steps [12], as explained in Section II. For each searching task, the participants executed multiple queries in Google, and assessed the resulted web pages regarding two aspects: 1) The relevance of each web page to the searching task, and 2) The AK concepts which exist in each web page. Further details are presented in Section III-E.

To analyze the collected data, and answer the RQs, we used the following analysis methods:

- *Web pages classification*: To answer RQ1.1, RQ1.2 and RQ3.2, we classified collected web pages (retrieved by Google for each query) into their respective AK source using a semi-automated approach (see Section III-F).
- *Descriptive statistics and correlations*: To answer RQ1.2, RQ2 and RQ 3, we used descriptive statistics and evaluated correlations between AK sources, AK concepts and relevance of web pages using Pearson $\tilde{\chi}^2$ test [15] (see Section III-H).
- *Effectiveness measurement*: To answer RQ3.1 and RQ3.2, we measured the effectiveness of Google using standard information retrieval metrics (see Section III-G).

C. Participants of the case study

The participants of the case study are 53 software engineers. 50 of the participants attended a software architecture master course at the University of Hamburg, and 3 additional software engineers volunteered to participate in the study. An overview on the industrial experience of the participants is presented in Table I. Additional information regarding the technical background of the participants is available online[1].

D. Searching tasks

To answer the RQs, each participant (see Section III-C) solved three searching tasks, where each task corresponds to one of the three ADD steps (see Section II).

[1]github.com/m-a-m-s/ICSA2021

TABLE II: Architectural searching tasks

ADD step	ID	Task description
Identify design concepts	T1	For a realtime dashboard, identify middleware technologies which scale to >100k users
	T2	A system needs to communicate with mobile apps. Identify JSON parsers for Java with high performance, considering license constraints.
Select design concepts	T3	A system communicates with a knowledge base via publish/subscribe patterns. Compare interoperability and latency of RabbitMQ, Kafka, and ActiveMQ.
	T4	Compare three technology families for big data systems: data collector, message brokers, and ETL engines. Requirements are throughput of 15,000 events/sec and availability of 99.99%.
Instantiate architecture elements	T5	CRM apps communicate with other systems using Apache Camel and RabbitMQ. Search for technology features and components designs to determine mechanisms channeling, translation and routing, and deployment topology.
	T6	An application exposes services to other apps. Search for best practices regarding service decomposition to achieve high cohesion and low coupling.

To support the validity of our results, we have designed two searching tasks for each ADD step. Thus, we have in total six tasks, from which we have randomly assigned three tasks (one task per ADD step) to each participant. The six searching tasks are real design problems, which have been gathered based on interviews with practitioners in a previous study [16] within the field of architectural knowledge. Table II presents a brief description of the six searching tasks, and their relationship to the ADD steps. A complete description of each task is available online[1].

E. Case Study procedures

1) Preparations before the study: To support the validity of our results, it is important to ensure that the participants have a clear understanding about the procedures of the study, as well as the searching tasks, and AK concepts. Therefore, the authors met with the participants in two sessions and explained the study procedures, as well as the assessment of web pages regarding their relevance and AK concepts. After the first session, each participant received a user-guide (available online[1]) with details about the study and a video tutorial on how to perform the tasks during the study. At the beginning of the second session and before conducting the study, the authors made a demo on assessing the relevance and specifying AK concepts for an example web page. In this demo, the participants were asked to specify the relevance and AK concepts for this example web page and provide their input via a polling feature in a web-conferencing tool. The polling results were discussed with the participants to show them how to correctly assess the relevance and how to specify AK concepts for each web page e.g. by paying particular attention to the requirements and constraints of the tasks.

We provide below the definitions for the degrees of relevance, that participants could choose from (in a five-level Likert scale):

- *Very High Relevance (VH)*: The web page discusses a similar problem to that of the task and contains useful

information. The web page provides an answer to the searching goal, and helps with fulfilling more than one requirement of the task.
- *High Relevance (H)*: The web page addresses a similar problem to that of the task and contains useful information. The web page provides an answer to the searching goal, and helps with fulfilling one requirement of the task.
- *Medium Relevance (M)*: The web page addresses a different problem to that of to the task at hand, but it provides some relevant information to the task, which could be an answer to the searching goal. Nevertheless, the provided information does not match specifically the task's requirements.
- *Low Relevance (L)*: The web page contains information, which is only remotely relevant to solving the given task, but might help for refining the search.
- *No Relevance (N)*: The web page has nothing to do with the task. It has no relevant information.

Moreover, each participant specified certain AK concepts for each web page. The list of AK concepts has been derived from existing literature [4], [17], [18] and is as follows:

- *Solution description*: general information on an architectural solution.
- *Solution alternatives*: multiple (alternative) architectural options for a certain design issue. The architectural options could be listed in the text or as a comparison of different options.
- *Solutions benefits*: information about the advantages of certain architectural solutions.
- *Solutions drawbacks*: information about the disadvantages of certain architectural solutions, even discouraging their application.
- *Made design decisions*: explanation about the architecture of a specific system. This includes the description of an existing architectural design of a specific system, or the explanation about certain design decisions of a specific system.
- *Others*: other relevant architectural information.

2) Study execution: We asked the participants to perform the tasks in the order given to them. To ensure that the sequence of tasks does not influence the study, we provided each participant with a different sequence of tasks (available online[1]). Moreover, we asked each participant to perform at least three queries per task and to evaluate the top 10 Google results for each query, as most users do not assess more than the top 10 results on the first page.

To facilitate specifying the relevance and AK concepts for each web page, we developed a Google Chrome plugin (provided online[1]), which offers a user interface and stores submitted relevance and AK concepts in a database.

The participants started the study in a synchronous web-conference meeting, where they were able to directly ask for clarification if they had any uncertainties. By the end of the web-conference, the participants continued solving the tasks independently, while using the provided plugin to specify the relevance and AK concepts for each web page.

3) After the study: After solving all the tasks, the participants were asked to fill out an exit survey (available online[1]). We asked the participants about their experience searching for architectural information using Google, as well as the complexity to analyze web pages regarding AK concepts.

F. Web pages classification

As a result of the study, we received 5175 web pages (with 2623 unique pages) from executing 477 queries in Google, where each web page is evaluated from a participant regarding its relevance and AK concepts. To answer RQ1.1 and support answering the other RQs, we analyzed the resulted unique web pages to determine the AK source (e.g. forum, blog) for each page. To achieve this, we followed a semi-automated approach using two main steps, which are explained in the following sub-sections.

1) Automatically clustering URLs of web pages: We executed a clustering algorithm [19] on the list of URLs to determine initial clusters of web pages. We have decided on this algorithm due to its excellence in clustering short text, compared to other classical clustering and topic modeling algorithms (e.g. LDA). Before executing the clustering, we filtered the URLs regarding symbols and stop words, and differentiated between host name and path within a URL.

We started the clustering algorithm using a big number of clusters (i.e. 100 clusters), and then reduced the number of clusters gradually after checking the results to reach the best possible clusters of URLs. Starting with a big number of clusters facilitated determining the commonalities between smaller groups of URLs, which could be later aggregated into a single cluster. After several iterations of clustering, we achieved the best results by having 31 clusters.

By executing the clustering algorithm on the 2623 unique URLs, it succeeded to split the URLs into 27 consistent clusters with 1280 URLs (e.g. one consistent cluster is all blog pages in Apache websites), and 4 inconsistent clusters (i.e. mixed of different AK sources) with 1343 URLs. This separation has been determined by inspecting samples of web pages of the URLs within each cluster.

2) Manually classifying web pages: To determine the AK source for each web page, we started by analyzing the 27 consistent clusters to determine dominant AK sources in each cluster. Manually classifying web pages in the consistent clusters was done by inspecting sample web pages from the cluster to determine the dominant AK source category of this cluster. As a result of this step, we identified 15 initial categories of AK sources, and classified the 1280 URLs within the consistent clusters.

Based on the 15 initial categories of AK sources, the first three authors manually categorized the rest of the URLs (1343 web pages from within the 4 inconsistent clusters) into their respective categories of AK sources. This has been done by inspecting each of the web pages. While manually inspecting the web pages, we ignored offline web pages, spam and web pages in languages other than English or German. Moreover,

a cross-check validation has been conducted between the first and second authors, as well as between the first and third authors to ensure agreement on the classification. To ensure agreement between the authors, we merged the 15 initial categories into 9 categories.

As a result of this step, we identified categories of AK sources on the web (see Section IV-A). Moreover, we created a corpus of 2522 unique web pages, which are categorized based on their AK sources.

G. Measurement of effectiveness

To answer RQ3.1 and RQ3.2, we measure the effectiveness of Google using two metrics: $Precision@k$ and *Normalized Discount Cumulative Gain*($nDCG@k$), where k is the maximum number of search results that are considered for evaluation. We considered k from 1 to 10 in our evaluations.

$Precision@k$ [20] is the ratio between the number of relevant web pages (low or medium or high or very high), by the number of retrieved web pages in the results.

The ranking and relevance of the retrieved web pages are important factors to assess the effectiveness of search engines. However, Precision does not consider the ranking and relevance of web pages. Therefore, we use $nDCG@k$ [20], which consider both the ranking and relevance of the retrieved web pages. $nDCG@k$ is a well known metric in information retrieval and has been used successfully in software engineering research (e.g. [21]).

The main idea of $nDCG@k$ [20] is to compare the ideal ranking of web pages ($IDCG$) to the ranking retrieved from a search engine for a certain query (DCG). For example, consider a task for which two users execute a query: the first user rates the top three web pages with relevance 3, 3, 1 while the second user rates the top three web pages with relevance 2, 2, 1. Meanwhile, the ideal ranking for this task is 3, 3, 2 when evaluating the top three search results. The $nDCG@3$ will compare the rankings 3, 3, 1 and 2, 2, 1 against 3, 3, 2.

To compare the rankings of queries with the ideal ranking, we divide the $DCG@k$ of each query with the $IDCG@k$ (the $DCG@k$ of the ideal ranking). The ideal ranking is based on combining the individual rankings of web pages (based on their relevance to a task) from all participants in the study.

In order to translate the importance of relevance and ranking into a metric, we calculate the $DCG@k$ for each ranking (from each query). The $DCG@k$ provides different weights for web pages based on their relevance and ranking in the list of results (i.e. the higher the relevance and rank, the more weight). One common way to implement the $DCG@k$ is to use the logarithmic scale to provide the right weight based on the relevance and ranking. Specifically, we calculate $DCG@k = \sum_{i=1}^{k} \frac{2^{rel_i} - 1}{log_2(i+1)}$, where rel_i is the degree of relevance of a web page found by a query (based on the Likert scale defined in Section III-E1).

To calculate the $nDCG@3$ for the previous examples q1 and q2, we calculate the $DCG@k$ for the rankings 3, 3, 1 (ranking of q1), 2, 2, 1 (ranking of q2) and 3, 3, 2 (the ideal ranking).

H. Statistical analysis and significance tests

To answer RQ1.2, RQ2 and RQ4 we used descriptive statistics. Moreover, we executed Pearson $\tilde{\chi}^2$ correlation tests [15] to determine the most relevant relationships.

To compute Pearson $\tilde{\chi}^2$ correlation tests and measure the phi coefficient we used the statistics tool SPSS[2]. Due to the large sample size, all correlations are significant (even to the 1% level), but only the phi coefficients measured for RQ2 revealed correlations with effect sizes greater than 0.3 which means medium strong or strong correlations [22].

To answer RQ1.2, we used descriptive statistics to determine relationships between AK sources (based on our classification see Sections III-F and IV-A), and the AK concepts (as specified by practitioners, see Section III-E1). For this, we counted the occurrences of each AK concept on the web pages and aggregated this for each AK source separately.

To answer RQ2, we performed a two-tailed T-test [23] with $\alpha = 0.05$ on all values of $Precision@k$ and $nDCG@k$ for each of the three ADD steps separately. Executing the T-test requires determining if the calculated values of $Precision@k$ and $nDCG@k$ have equal or unequal variances. Thus, we executed an F-test [23] with $\alpha = 0.05$ to determine if the values of $Precision@k$ and $nDCG@k$ have equal or unequal variances. Based on the results of F-test, we have executed the right method of T-test, either with equal or unequal variances.

We answered RQ4 using descriptive statistics to present the distribution of relevance within each AK concept, i.e. we evaluated only the web pages that contain the respective AK concept. Both the AK concepts and relevance of web pages have been specified by the participants during the study (see Section III-E1).

IV. RQ1: Architecture knowledge sources and concepts on the web

A. RQ1.1: AK sources on the web

Table III shows the identified AK sources, which have been retrieved by Google when searching for architectural information. Moreover, Table III shows the percentages of each category based on our corpus (2522 web pages). **The results suggest that the predominant categories are blogs and tutorials as well as technology vendor documentations** (and to a lesser extent scientific contents). During our manual classification (see Section III-F), we found a wide variety of blogs, such as personal blogs (e.g. Martin Fowler's blog[3]) and technology blogs (e.g. SAP technology blog[4]). Moreover, technology vendor documentations come at different levels of detail, from a high level description of technology to detailed code specification (e.g Apache technologies[5][6]).

[2]https://www.ibm.com/de-de/analytics/spss-statistics-software
[3]martinfowler.com
[4]blog.sap-press.com
[5]kafka.apache.org/documentation
[6]flink.apache.org

TABLE III: AK sources on the web

AK source	Example	%
Blogs and tutorials	dzone.com	39%
Technology vendor documentations	metamug.com/docs	23%
Scientific contents	ieeexplore.ieee.org	13%
Forums	stackoverflow.com	7%
Technical books and white papers	livebook.manning.com	4.5%
Source code repositories	github.com	4.5%
Knowledge repositories	stackshare.io	4%
Presentations and videos	slideshare.net	2.7%
Others (e.g. tools, patents)	google.com/patents json.parser.online.fr	2.3%

TABLE IV: Correlation between each of the AK concepts

	Descr.	Altern.	Benef.	Drawb.	Decisions
Description	1.000	0.181	**0.347**	0.203	0.222
Alternatives		1.000	**0.355**	**0.340**	0.079
Benefits			1,000	**0.651**	0.157
Drawbacks				1.000	0.122
Decisions					1.000

TABLE V: Co-occurrences between Benefits and Drawbacks

	Drawbacks		
Benefits	does not contain	contains	
does not contain	3203	46	3249
contains	820	1106	1929
Sum	4023	1152	5175

On the other hand, **source code repositories (e.g. Github) and knowledge repositories are not commonly found** in the search results, and they come in less variety. Furthermore, the results show that **issue tracking systems (e.g. Apache issue trackers[7]) are not retrieved at all by Google** when searching for architectural information.

B. RQ1.2: Prominent AK concepts in the AK sources

Figure 1 shows the percentages of occurrence for each of the AK concepts in the found AK sources (see Section IV-A). Because each web page can contain multiple AK concepts, the sum of percentages for all AK concepts in an AK source can exceed 100% (for more details see the online resources[1]). Based on Fig. 1, we can observe the following:

- Solution alternatives, benefits and drawbacks are less present within the technology vendor documentations and source code repositories; presumably this is because these AK sources usually discuss a single architectural solution, such as the documentation or source code of a specific technology.
- **Made design decisions are underrepresented in all AK sources**. Thus, finding concrete examples of design decisions (e.g. a decision on specific components of a system), and their rationale for existing systems are not as easy to find among the AK sources on the web.
- Solution descriptions are more prominent than the other AK concepts within technology vendor pages. This is probably because this AK source focuses on describing a certain architectural solution in more detail.

V. RQ2: Co-occurrences of AK concepts

In order to evaluate the co-occurrences of AK concepts in web pages, we measured the correlation between the AK concepts (as explained in Section III-H). Table IV shows the correlation coefficients between each pair of AK concepts.

From Table IV, we can observe that **the correlation between benefits and drawback stands out with a correlation coefficient of 0.651** which means a strong correlation exists [22]. This means that either a) web pages containing benefits often also contain drawbacks or b) web pages containing drawbacks often also contain benefits. To determine the right interpretation for the correlation between benefits and drawbacks, we inspected the exact frequencies of co-occurrences between benefits and drawbacks (see Table V). We can observe that drawbacks are rarely presented without benefits (46 out of 1152), while there are more web pages containing benefits without drawbacks (820 out of 1929). This means that **it is common to find web pages with benefits and no drawbacks, while it is rare to find web pages with drawbacks and no benefits**. This can be also seen in Fig. 1. For example, blogs and tutorials contain nearly twice the amount of benefits compared to drawbacks. This indicates that software engineers in communities tend to praise the benefits of technologies rather than realistically evaluate the pros and cons of technologies equally.

From Table IV, we can also observe that the correlations between alternatives and benefits, as well as between alternatives and drawbacks (0.355 and 0.340 respectively) are also medium strong [22]. This indicates that **lists of alternative solutions, are usually accompanied with a comparison between them regarding their benefits and drawbacks**. One common example for this are knowledge repositories and forum entries containing alternatives, with their drawbacks and benefits.

A final observation from Table IV, is a medium strong correlation (0.347) [22] between solution descriptions and benefits. This is quite common in technology vendor pages, where they commonly describe their technologies and their benefits, while omitting their drawbacks (see Fig. 1).

VI. RQ3: Search engines support for ADD steps

A. RQ3.1: Effectiveness of search engines to support ADD

Figure 2 shows the average $nDCG@k$ and $Precision@k$ when searching for architectural information during the three ADD steps. From Fig. 2, we can observe that the maximum average $Precision@1$ is 0.75 (for the "Select design concepts" step), which means that on average 75% of the queries retrieved a relevant (low, medium, high, very high) web page at the top result (k=1) from Google. On the other hand, the lowest $Precision@10$ is 0.48 (for the "Identify design concepts" step), which means that on average 48% of the retrieved top 10 web pages were relevant (i.e. have relevance of low, medium, high, very high); in this case, web search engines return both relevant and irrelevant results equally.

[7] issues.apache.org

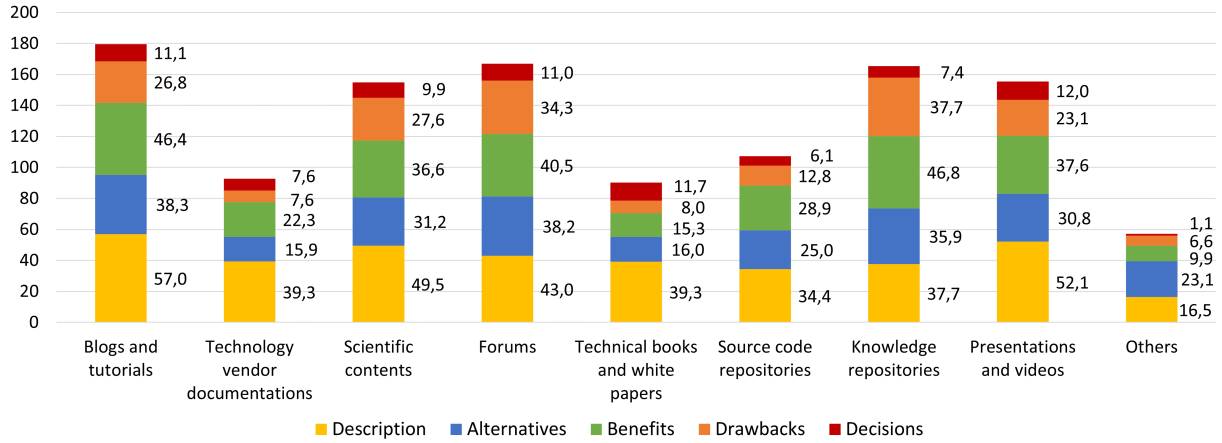


Fig. 1: Distribution of AK Concepts per AK source

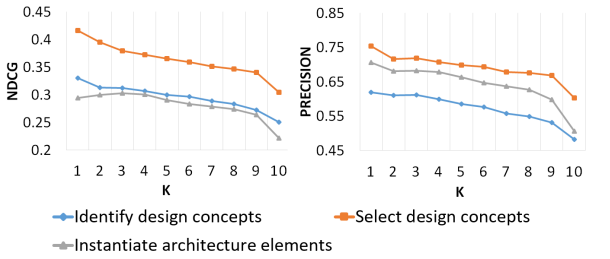


Fig. 2: Average nDCG and precision in finding architectural information for each ADD step.

To compare the differences between ADD steps regarding their $nDCG@k$ and $precision@k$, we executed a significant T-test (as explained in Section III-H). The detailed results of the tests are available online[1].

From the test, **we found that the "Select design concepts" step has significantly higher $nDCG@k$ for k between 1 and 9 ($nDCG@k_{1\rightarrow9}$) compared to the other two design steps**, while there is no significance difference in $nDCG@k_{1\rightarrow10}$ between the "Identify design concepts" and "Instantiate architecture elements" steps. For example, from Fig. 2 the average $nDCG@1$ for the "Select design concept" ADD step is 0.42, while the average $nDCG@1$ is 0.29 for the "Instantiate architecture elements" ADD step. This means that on average 42% of the "Select design concept" queries retrieved highly relevant (i.e. very high) web pages at the top result from Google, compared to just 29% for the "Instantiate architecture elements". Thus, **finding AK to select an architectural solution from alternatives is easier than finding AK to instantiate a certain architectural solution**.

The significance test also showed that the "Select design concepts" step has significantly better $precision1 \rightarrow 9$ compared to the "Identify design concepts". In contrast, there is only a slight difference of $precision1 \rightarrow 10$ between the "Select design concepts" and "Instantiate architecture elements".

Looking at both the $nDCG@k$ and $precision@k$, on the one hand we can notice that the "Identify design concepts" has significantly lower $nDCG@k$ and $precision@k$ compared to the "Select design concepts". This means **it is harder to find AK on alternative architectural solutions than to find information on how to compare them**. On the other hand, the "Instantiate architecture elements" step has significantly lower $nDCG$ compared to the "Select design concept" step but a comparable $Precision$. Because $nDCG$ considers the ranking and relevance of results compared to $Precision$, the $nDCG$ and $Precision$ of the "Instantiate architecture elements" indicate that **Google finds many distantly relevant web pages (e.g. general concepts on components design) to support the "Instantiate architecture elements" step**. In contrast, it is challenging for Google to find highly relevant solutions, which fulfill specific requirements.

B. RQ3.2: The most influential AK sources

Figure 3 shows the top six AK sources (see Table III) with the highest effectiveness in each of the ADD steps. We can observe the following:

Blogs and tutorials have the biggest contribution on the effectiveness of Google in all three ADD steps. Thus, **blogs have the highest relevance and highest ranking, and show to contain the most useful AK compared to other sources**.

***Technology vendor documentation* shows to have the second highest contribution on the effectiveness of Google for both the "Identify design concepts" and the "Instantiate architecture elements" steps**. This indicates their usefulness to identify options for architectural solutions. Moreover, technology vendor documentation contains detailed information regarding the technology features, which are useful for the "Instantiate architectural elements" step. As shown in Section IV-B solution alternatives and solution drawbacks are underrepresented in technology vendor documentation, which may be the reason why they are not as effective for the "Select design concepts" step.

***Knowledge repositories* (e.g. [9]) show to be effective for $@k_{1\rightarrow2}$ (i.e. for the top two Google results) only within the "Select design concepts" step**, while it has a negligible contribution on the effectiveness within the other two ADD

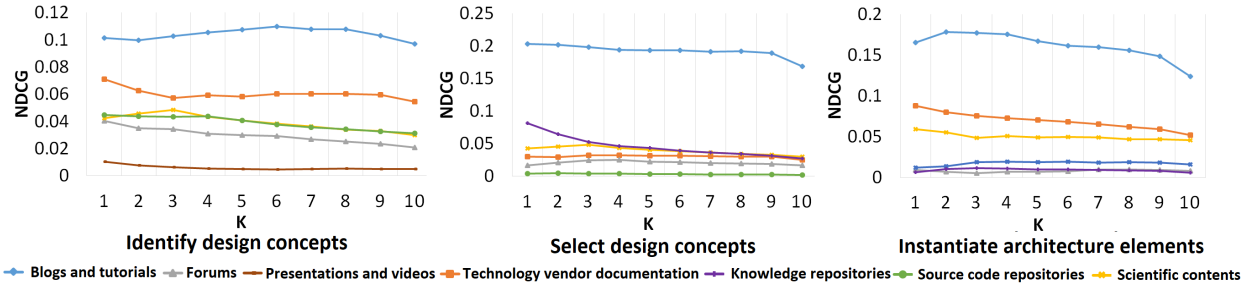


Fig. 3: The contribution of each AK source in the effectiveness of Google per ADD step

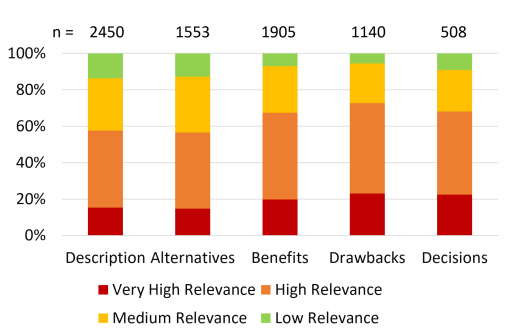


Fig. 4: Distribution relevance per concept

steps. This is because most knowledge repositories on the web (e.g. stackshare) contain mainly high level information to compare multiple technologies with each other; they can thus somewhat help with "Selecting design concepts".

***Scientific contents* (e.g. papers or thesis) show to have a higher contribution on the effectiveness of Google than forums and source code repositories, for the "Instantiate architecture elements" step**. This could be due to the scarcity of AK regarding component designs in forums and source code repositories, while scientific contents are rich with component designs (e.g. reference architectures).

***Forums and source code repositories* show to have a limited influence on the effectiveness of Google of searching within the three ADD steps**. This might be because both forums and source code repositories are not well found by Google (see Table III) compared to the other AK sources.

VII. RQ4: AK concepts in highly relevant web pages

Figure 4 shows the total number of relevant web pages (presented as "n" in the figure), and the percentages of the different degrees of relevance (low, medium, high and very high as rated by the participants), in which AK concepts appear. From Figure 4, we can observe that **web pages that contain solution benefits, solution drawbacks and made design decisions are most relevant for finding AK**, while web pages with solution description' and solution alternatives have a lower probability to be high or very high in relevance. For example, a web page containing solution drawbacks has a higher probability to be rated high or very high in relevance (72.7%) than a web page containing solution alternatives (56.5%). (Details are available online[1]).

Even though, solutions drawbacks and made design decisions are not found by Google in its results as often as solution descriptions and solution alternatives, they showed to be the most common AK concepts in highly relevant pages, and thus they could be the most important for making design decisions.

VIII. Discussion

A. RQ1: AK sources and concepts on the web

1) Implications for practitioners: The list of AK sources on the web (see Table III) and their respective AK concepts (see Fig. 1) could guide practitioners to determine the scope of searching (i.e. to search in the whole web or in specific web sites). For example, forums like Stackoverflow have shown to contain useful AK (e.g. [7], [24]) about the benefits and drawbacks between architectural solutions (see Fig. 1). However, forums are not well considered by Google (see Table III). Thus, practitioners should focus the scope of searching on specific forums like Stackoverflow to find more AK on the benefits and drawbacks between solutions.

2) Implications for researchers: The list of AK sources in Table III provides an overview of possible web sources of AK, from which researchers could extend current approaches to document AK (e.g. [2]). On the one hand, blogs are well indexed by Google but not previously explored by researchers for AK. Thus current studies on AK (e.g. [7], [24]) could be extended to explore the AK in blogs. On the other hand, some AK sources are previously explored by researchers for AK (e.g. forums [24] and issue tracking systems [3]), but they are not well indexed by Google. These AK sources deserve extra attention to improve their ranking on web search results.

The distribution of AK concepts in each AK source in Fig. 1 show that there is no single AK source which contains specific AK concepts. However, certain AK concepts come more often in certain AK sources. Thus, researchers could make use of this information to develop AK documentation approaches, which consider multiple AK sources. For example, an AK approach can find and document benefits and drawbacks from forums, and solution description from technology vendor documentations.

B. RQ2: Co-occurrences of AK concepts

1) Implications for practitioners: The results in Section V show that benefits are preconditions for the occurrence of

drawbacks in web pages, and that drawbacks rarely come alone in a separate web page. The results increase the awareness of practitioners that the predominant presence of benefits on the web does not mean lack of drawbacks for architectural solutions; it rather means the rarity of drawbacks on the web. To resolve this problem, practitioners should use web search engines to search explicitly for the drawbacks of architectural solutions.

2) Implications for researchers: The co-occurrences between AK concepts in Section V shows that AK concepts do not come all together in a single web page. However, subsets of AK concepts co-occur together more often than others. For instance the correlations between AK concepts in Table IV could be grouped into three subsets of AK concepts: (benefits and drawbacks), (alternatives, benefits and drawbacks), and (Solution description and benefits). Researchers need to consider this division of AK concepts on the web, when developing approaches to automatically capture AK (i.e. search for AK, and codify it), e.g. developing dedicated AK capturing approaches for each subset of AK concepts.

C. RQ3: How search engines support the ADD steps

1) Implications for practitioners: The results in Section VI-A verified previous experiences with practitioners [2] that web-search engines return many irrelevant results. However, our study provides the first precise evidence on the effectiveness of Google to find AK for each ADD step. The differences in the effectiveness between the ADD steps (as shown in Fig. 2) can help increase the awareness regarding the expected relevance of the retrieved results. This can guide practitioners to determine when to rely on web search engines, and when to seek other ways (e.g. asking experts) to search for AK. For instance, practitioners could rely on search engines to find AK for the "Select design concepts" step. But, they are better off seeking other ways during the "Instantiate architecture elements" step, as many of the retrieved results from web search engines are distantly relevant.

2) Implications for researchers: The results in Section VI-A support prioritizing requirements for AK management approaches, and especially about which ADD steps to support better. For instance, researchers should give higher priority to extending AK capturing approaches for the "Identify design concepts" and the "Instantiate architecture elements" steps, as Google has lower effectiveness in supporting these steps. This is probably because Google cannot determine the context of software engineers when identifying or instantiating architectural solutions. To support the "Identify design concepts" step, researchers could propose approaches that relate business requirements, design issues and alternative architectural solutions. Moreover, to support the "Instantiate architecture elements" step, design decisions and their rationale should be captured from existing systems and shared with practitioners.

The results in Section VI-B verify that blogs and tutorials are valid options for exploring and capturing AK, because they are the most relevant and highly ranked. However, during our web pages classification (see Section III-F), we found that blogs and tutorials are quite diverse (e.g. private versus company blogs). This makes it very challenging to apply information retrieval or extraction techniques on blogs. Thus, we propose first exploring the different types of blogs and tutorials and the AK concepts inside them. Moreover, the results in Section VI-B can support developing specialized software architecture searching approaches. One idea is to re-rank results of Google differently for each ADD step; this can be achieved by developing heuristics based on the effectiveness of AK sources to support ADD steps (see Fig. 3).

D. RQ4: AK concepts in highly relevant web pages

1) Implications for practitioners: Section VII shows that solution drawbacks and made design decisions are the most frequently appearing AK concepts within highly relevant web pages. This indicates the importance of solution drawbacks and made design decisions for practitioners. On the one hand, solution drawbacks provide AK on when an architectural solution could be discouraged. This is important for practitioners to prevent selecting architectural solutions, which must be replaced later due to their drawbacks. Replacing architectural solutions after their implementation often requires substantial effort. On the other hand, made design decisions can be potentially reused by practitioners. Thus, practitioners should share more AK about solution drawbacks and their made design decisions on the web (e.g. in blogs or white papers); this can be done in academic or industry conferences, or by creating dedicated community websites for this type of AK.

2) Implications for researchers: The results in Section VII support prioritizing AK management approaches to focus on important AK concepts with the highest relevance to practitioners. For example, researchers should consider automatically capturing AK related to made design decisions and solution drawbacks more than other AK concepts, because they contain the most relevant AK for making design decisions. Developing such approaches can be challenging because both made design decisions and solutions drawbacks present minorities compared to other AK concepts on the web (see Fig. 1). Thus, using classification algorithms could be useful to filter web pages on the web with drawbacks or decisions.

IX. Threats to validity

1) Construct validity: In our study in Section III-E2, the experience and background of participants (see Table I) might have influenced their assessment of web pages (i.e. specifying the relevance and the AK concepts). To mitigate this, the participants received training, and materials (as a user guide and video). In addition, the participants were accompanied by the researchers at the beginning of the study. Also, since participants solved some tasks on their own time, we did not have full control of their behavior (e.g. fatigue). However, we tried to mitigate this by changing the sequence of tasks for each participant.

Another threat of validity is the possibility of mistakes in gathering data from participants during the study. To mitigate this, the participants used a plugin, which captured their input

and stored it in a database for analysis. In this way, we were able to verify the data for any possible mistakes, and validate it with the participants.

2) Reliability: The classification of web pages into their respective AK sources presents a threat to reliability. However, we tried to ensure consistency in the classification, by establishing categories that have the highest agreement between the first three authors of the paper. Moreover, to facilitate replicating the analysis, we provide our corpus online[1].

3) External validity: Our study used Google as our case for a search engine, without exploring other search engines (e.g. Bing or Baidu). However, Google is the most popular search engine, and thus our results could be generalized on most users. Our limited number of tasks (six tasks in Table II) compared to the high variety of architectural tasks in practice is another threat to the external validity. To partially mitigate this, we have designed two tasks for each ADD step to reduce the dependency on a single task. Finally, the limited number of participants (see Table I) in the study is another threat to the external validity of results. However, the participants have different backgrounds and experiences, which supports, to some extent the generalizability of results.

X. Related Work

We are not aware of any previous studies on web searching approaches to find AK. Thus, our study is the first to investigate using web searching to perform architectural tasks. In this section, we discuss some related work in the fields of AK, as well as studies on web searching in software engineering.

Architectural knowledge. Researchers in the field of AK have explored the main AK concepts, such as design decisions [18], their types [25], rationale of decisions [26] and solutions alternatives [5]. These studies established the fundamental AK concepts, which we investigate in our presented study. However, they do not propose approaches for capturing or finding AK.

Some approaches propose catalogs and repositories of AK to facilitate structuring and sharing AK. For example, Elmalki and Zdun [27] modeled common types of ADDs for microservice architectures. Malakuti et al. [28] created a catalog for the types of ADDs when designing IOT system. While they can guide software engineers during design space exploration, they require extra manual effort to find and codify AK.

Recent efforts on AK propose approaches to automatically capture AK from different AK sources using machine learning and information retrieval techniques. For example Gorton et al. [2] proposed an approach to identify AK in technology documentation, and specially identify documents with certain architectural tactics (as one architectural solution). Bhat et al. [3] captured AK from issue tracking systems. They especially captured and classified the different types of design decisions (as one AK concept) in issue tracking systems. Soliman et al. [16] improved the effectiveness of searching for AK in Stackoverflow. The approach uses machine learning and heuristics to re-rank the results of search engines. However, all three approaches do not study the effectiveness of web searching to support architectural activities.

Web searching in software engineering. Some approaches empirically investigated the usage of web searching by software engineers. Xia et al. [11] empirically determined the most common and most complex software engineering tasks (e.g. searching for solutions to bugs or for third party libraries) using web search engines. Rahman et al. [29] studied the effectiveness of web searching when searching for source code on the web. Their results show that searching for source code is harder than searching for other types. Hassan et al. [30] empirically analyzed web search queries and results related to code exceptions. Moreover, they proposed an approach to capture knowledge related to code exceptions from the web. However, these three studies do not consider finding AK to perform software architectural tasks.

Other approaches tried to improve the effectiveness of retrieving code examples from the web. For example, Wang et al. [31] proposed an approach to capture code examples regarding API usage from the web. Sirres et al. [32] proposed an approach to improve the effectiveness of source code retrieval by augmenting queries with knowledge from Stackoverflow and Github. Their results improve the effectiveness of source code searching compared to Google. However, their approaches focus on finding source code rather than AK.

XI. Conclusion and Future Work

Our main goal in this paper is to explore the retrieved architectural knowledge (AK) from the web, and the effectiveness of web search engines, when performing three Attribute-Driven Design (ADD) architectural steps. To achieve our goal, we conducted an exploratory study with software engineers, who used Google to find AK for making design decisions. Our results provide several interesting results. First, we provide an overall view on the different sources of AK and their associated AK concepts on the web. This can be useful to extend current AK capturing approaches. Second, we determined the differences in the effectiveness of Google when performing the ADD steps, which help to better understand the main capabilities of web search engines to search for AK. Finally, we identified AK concepts on the web, which provide the highest benefit for practitioners when making design decisions. This can provide a guidance for practitioners to share their AK on the web.

The results of this study motivate us to extend current AK capturing approaches to focus on the most useful AK sources (e.g. blogs), and the most scarce and useful AK concepts (e.g. design decisions and drawbacks of solutions). Moreover, we aim to propose specialized web searching approaches to enhance the effectiveness of searching for AK.

References

[1] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*, 3rd ed. Addison-Wesley Professional, 2012.

[2] I. Gorton, R. Xu, Y. Yang, H. Liu, and G. Zheng, "Experiments in Curation: Towards Machine-Assisted Construction of Software Architecture Knowledge Bases," in *IEEE/IFIP ICSA 2017*, 4 2017, pp. 79–88.

[3] M. Bhat, K. Shumaiev, A. Biesdorf, U. Hohenstein, and F. Matthes, "Automatic extraction of design decisions from issue management systems: A machine learning based approach," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10475 LNCS. Springer Verlag, 2017, pp. 138–154.

[4] M. Soliman, M. Riebisch, and U. Zdun, "Enriching Architecture Knowledge with Technology Design Decisions," in *WICSA*, 5 2015, pp. 135–144.

[5] O. Zimmermann, J. Koehler, F. Leymann, R. Polley, and N. Schuster, "Managing architectural decision models with dependency relations, integrity constraints, and production rules," *Journal of Systems and Software*, vol. 82, no. 8, pp. 1249–1267, 2009.

[6] A. Tang, P. Avgeriou, A. Jansen, R. Capilla, and M. Ali Babar, "A comparative study of architecture knowledge management tools," *Journal of Systems and Software*, vol. 83, no. 3, pp. 352–370, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/B6V0N-4X4GHP5-1/2/84a45c0d6dda12f7f563273ff85be120

[7] M. Soliman, M. Galster, and M. Riebisch, "Developing an Ontology for Architecture Knowledge from Developer Communities," in *IEEE/IFIP ICSA 2017*, 4 2017, pp. 89–92. [Online]. Available: https://www.inf.uni-hamburg.de/en/inst/ab/swk/research/publications/pdf/2017-soliman-icsa.pdf

[8] A. Barua, S. W. Thomas, and A. E. Hassan, "What Are Developers Talking About? An Analysis of Topics and Trends in Stack Overflow," *Empirical Softw. Engg.*, vol. 19, no. 3, pp. 619–654, 6 2014.

[9] I. Gorton, J. Klein, and A. Nurgaliev, "Architecture Knowledge for Evaluating Scalable Databases," in *WICSA, IEEE/IFIP*, 5 2015, pp. 95–104.

[10] R. Capilla, A. Jansen, A. Tang, P. Avgeriou, and M. A. Babar, "10 Years of Software Architecture Knowledge Management," *J. Syst. Softw.*, vol. 116, no. C, pp. 191–205, 6 2016. [Online]. Available: http://dx.doi.org/10.1016/j.jss.2015.08.054

[11] X. Xia, L. Bao, D. Lo, P. S. Kochhar, A. E. Hassan, and Z. Xing, "What do developers search for on the web?" *Empirical Software Engineering*, vol. 22, no. 6, pp. 3149–3185, 12 2017. [Online]. Available: https://link.springer.com/article/10.1007/s10664-017-9514-4

[12] R. Kazman and H. Cervantes, *Designing Software Architectures: A Practical Approach*. Addison-Wesley Professional, 2016.

[13] K. Byström and P. Hansen, "Conceptual framework for tasks in information studies," *Journal of the American Society for Information Science and Technology*, vol. 56, no. 10, pp. 1050–1061, 2005. [Online]. Available: http://dx.doi.org/10.1002/asi.20197

[14] P. Runeson, *Case study research in software engineering : guidelines and examples*. Wiley, 2012.

[15] K. Pearson, "I. Mathematical contributions to the theory of evolution. {\textemdash}{VII}. On the correlation of characters not quantitatively measurable," *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, vol. 195, no. 262-273, pp. 1–47, 1 1900.

[16] M. Soliman, A. Rekaby Salama, M. Galster, O. Zimmermann, and M. Riebisch, "Improving the Search for Architecture Knowledge in Online Developer Communities," in *Proceedings - 2018 IEEE 15th International Conference on Software Architecture, ICSA 2018*. Institute of Electrical and Electronics Engineers Inc., 7 2018, pp. 186–195.

[17] O. Zimmermann, "Architectural decision identification in architectural patterns." in *WICSA/ECSA Companion Volume*, ser. ACM International Conference Proceeding Series, T. Mnnist, M. A. Babar, C. E. Cuesta, and J. E. Savolainen, Eds., vol. 704. ACM, 2012, pp. 96–103. [Online]. Available: http://dblp.uni-trier.de/db/conf/wicsa/wicsac2012.htmlZimmermann12

[18] A. Jansen and J. Bosch, "Software Architecture as a Set of Architectural Design Decisions," in *WICSA*, 2005, pp. 109–120.

[19] J. Yin and J. Wang, "A Dirichlet multinomial mixture model-based approach for short text clustering," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, New York, USA: Association for Computing Machinery, 2014, pp. 233–242. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2623330.2623715

[20] C. D. Manning, P. Raghavan, and H. Schütze, "Introduction to Information Retrieval," 2008. [Online]. Available: https://nlp.stanford.edu/IR-book/information-retrieval-book.html

[21] S. Gottipati, D. Lo, and J. Jiang, "Finding Relevant Answers in Software Forums," in *IEEE/ACM ASE 2011*, ser. ASE '11. IEEE Computer Society, 2011, pp. 323–332.

[22] J. Cohen, "Statistical Power Analysis for the Behavioral Sciences — ScienceDirect." [Online]. Available: https://www.sciencedirect.com/book/9780121790608/statistical-power-analysis-for-the-behavioral-sciences

[23] A. Dean, D. Voss, and D. Draguljić, *Design and Analysis of Experiments*, ser. Springer Texts in Statistics. Cham: Springer International Publishing, 2017. [Online]. Available: http://link.springer.com/10.1007/978-3-319-52250-0

[24] M. Soliman, M. Galster, A. R. Salama, and M. Riebisch, "Architectural Knowledge for Technology Decisions in Developer Communities: An Exploratory Study with StackOverflow," in *IEEE/IFIP WICSA 2016*, 4 2016, pp. 128–133.

[25] P. Kruchten, P. Lago, and H. Vliet, "Building Up and Reasoning About Architectural Knowledge," in *Quality of Software Architectures*, ser. Lecture Notes in Computer Science, C. Hofmeister, I. Crnkovic, and R. Reussner, Eds. Springer Berlin Heidelberg, 2006, vol. 4214, pp. 43–58.

[26] A. Tang, Y. Jin, and J. Han, "A rationale-based architecture model for design traceability and reasoning," *Journal of Systems and Software*, vol. 80, no. 6, pp. 918–934, 6 2007. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0164121206002287

[27] A. El Malki and U. Zdun, "Guiding architectural decision making on service mesh based microservice architectures," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11681 LNCS. Springer Verlag, 2019, pp. 3–19.

[28] S. Malakuti, T. Goldschmidt, and H. Koziolek, "A catalogue of architectural decisions for designing IIoT systems," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11048 LNCS. Springer Verlag, 2018, pp. 103–111.

[29] M. M. Rahman, J. Barson, S. Paul, J. Kayani, F. A. Lois, S. F. Quezada, C. Parnin, K. T. Stolee, and B. Ray, "Evaluating how developers use general-purpose web-search for code retrieval," in *Proceedings - International Conference on Software Engineering*. New York, NY, USA: IEEE Computer Society, 5 2018, pp. 465–475. [Online]. Available: https://dl.acm.org/doi/10.1145/3196398.3196425

[30] F. Hassan, C. Bansal, N. Nagappan, T. Zimmermann, and A. H. Awadallah, "An empirical study of software exceptions in the field using search logs," in *International Symposium on Empirical Software Engineering and Measurement*. New York, NY, USA: IEEE Computer Society, 10 2020, pp. 1–12. [Online]. Available: https://dl.acm.org/doi/10.1145/3382494.3410692

[31] L. Wang, L. Fang, L. Wang, G. Li, B. Xie, and F. Yang, "APIExample: An effective web search based usage example recommendation system for java APIs," in *2011 26th IEEE/ACM International Conference on Automated Software Engineering, ASE 2011, Proceedings*, 2011, pp. 592–595.

[32] R. Sirres, T. F. Bissyandé, D. Kim, D. Lo, J. Klein, K. Kim, and Y. L. Traon, "Augmenting and structuring user queries to support efficient free-form code search," *Empirical Software Engineering*, vol. 23, no. 5, pp. 2622–2654, 10 2018. [Online]. Available: https://link.springer.com/article/10.1007/s10664-017-9544-y

Chapter 18

Debiasing Architectural Decision-Making: An Experiment With Students and Practitioners

9 *(Debiasing)*	
Title	Debiasing Architectural Decision-Making: An Experiment With Students and Practitioners
Authors	Borowa K., de Almeida R. R., M. Wiese M.
Year	2025
Conference	IEEE 22nd International Conference on Software Architecture (ICSA)
Reference	[BAW25]

TABLE 18.1: Publication No. 9 *(Debiasing)*



Debiasing Architectural Decision-Making: An Experiment With Students and Practitioners

1st Klara Borowa
Warsaw University of Technology,
Institute of Control and Computation Engineering
Warsaw, Poland
klara.borowa@pw.edu.pl

2nd Rodrigo Rebouças de Almeida
Federal University of Paraíba
Department of Exact Sciences
Rio Tinto/PB, Brazil
rodrigor@dcx.ufpb.br

3rd Marion Wiese
Universität Hamburg
Department of Informatics
Hamburg, Germany
marion.wiese@uni-hamburg.de

2025 IEEE 22nd International Conference on Software Architecture (ICSA) | 979-8-3315-2090-8/25/$31.00 ©2025 IEEE | DOI: 10.1109/ICSA65012.2025.00029

***Abstract*—Cognitive biases are predictable, systematic errors in human reasoning. They influence decision-making in various areas, including architectural decision-making, where architects face many choices. For example, anchoring can cause architects to unconsciously prefer the first architectural solution that they came up with, without considering any solution alternatives. Prior research suggests that training individuals in debiasing techniques during a practical workshop can help reduce the impact of biases. The goal of this study was to design and evaluate a debiasing workshop with individuals at various stages of their professional careers. To test the workshop's effectiveness, we performed an experiment with 16 students and 20 practitioners, split into control and workshop group pairs. We recorded and analyzed their think-aloud discussions about improving the architectures of systems they collaborated on. The workshop improved the participants' argumentation when discussing architectural decisions and increased the use of debiasing techniques taught during the workshop. This led to the successful reduction of the researched biases' occurrences. In particular, anchoring and optimism bias occurrences decreased significantly. We also found that practitioners were more susceptible to cognitive biases than students, so the workshop had a more substantial impact on practitioners. We assume that the practitioners' attachment to their systems may be the cause of their susceptibility to biases. Finally, we identified factors that may reduce the effectiveness of the debiasing workshop. On that basis, we prepared a set of teaching suggestions for educators. Overall, we recommend using this workshop to educate both students and experienced practitioners about the typical harmful influences of cognitive bias on architectural decisions and how to avoid them.**

***Index Terms*—Software Architecture, Architectural Decisions, Architectural Decision-Making, Cognitive bias, Debiasing**

I. Introduction

Cognitive biases are systematic errors caused by the heuristics the human mind uses to reduce the complexity of various tasks [1], including decision-making. Cognitive biases distort human decision-making in various domains: from clinicians making erroneous medical diagnoses [2] to developers copying code without reading it [3]. Stacy and Macmillan [4] first observed the possible impact of cognitive biases in the realm of software engineering. In particular, how *representativeness, availability bias* and *confirmation bias* could influence software developers' daily activities. Since then, research on cognitive biases has significantly expanded to include at least 36 biases and their impact on various software engineering knowledge areas [5]. The influence of cognitive biases is particularly impactful in architectural decision-making (ADM) [6] [7], since software architecture can be considered as a set of design decisions [8]. However, few studies have focused on behavioral factors in ADM, with even fewer containing any empirical validation of decision-making techniques [9]. Notable examples of studies that have focused on improving the process of ADM include those that assess the usefulness of reflection – Razavian et al. [10] (experiment on students), and Tang et al. [11] (experiment on students and professionals).

Debiasing is a process that improves one's judgment by using techniques decreasing the impact of a particular cognitive bias [12]. A workshop can effectively mitigate some influences of cognitive biases on software practitioners' [13]. As far as we know, the only empirically tested debiasing training for ADM was conducted on a group of students with limited experience in software development [14]. This study had students perform theoretical tasks, and employed a pretest-posttest design. Given the doubts expressed by the software engineering community about the validity of the findings in experiments with student participants [15] [16], as well as numerous threats to validity associated with pretest-posttest experiments [17], further evaluation of debiasing through workshops is warranted. Therefore, the **goal** of this study is to conduct a debiasing intervention with a control group to ascertain its effectiveness and investigate its impact on individuals at various stages of their careers, i.e., students and practitioners. As such, we strive to answer the following research question:

> **RQ: How does the proposed debiasing architectural decision-making workshop influence students and practitioners?**

In particular, we explored whether the following effects would appear: **(1) Would the workshop decrease the number of cognitive bias occurrences? (2) Would the workshop increase the participants' use of debiasing techniques?**

To answer this research question, we performed 18 controlled experiments with 16 students and 20 software practitioners, divided into control and workshop groups.
The study's main **contribution** is the design and empirical evaluation of the workshop, which resulted in the following:

Part of this work was supported by Funetec-PB – Ct.: Phoebus 01/24 and Snet 04/22.

2835-7043/25/$31.00 ©2025 IEEE
DOI 10.1109/ICSA65012.2025.00029

Authorized licensed use limited to: Bibliothekssystem Universitaet Hamburg. Downloaded on December 11,2025 at 16:51:00 UTC from IEEE Xplore. Restrictions apply.

1) The workshop reduced the occurrence of biases for both students and practitioners. It decreased the occurrence of all three researched biases (*anchoring*, *confirmation bias*, *and optimism bias*). The decrease in anchoring and optimism bias was statistically significant.
2) The workshop increased the use of all three proposed debiasing techniques. This change was statistically significant in the case of the "listing multiple solutions" and "discussing drawbacks" techniques.
3) The workshop significantly improved the amount of non-biased arguments for and against architectural solutions while reducing biased arguments.
4) The students discussing their university projects were less impacted by cognitive biases than practitioners in the case of real-life architectures.
5) The positive effects of the workshop on practitioners were more substantial, i.e., the decrease in bias occurrences was greater for practitioners. This change was possible because of the higher initial bias level of practitioners, which left more room for improvement.
6) A set of teaching suggestions based on the factors that impacted the few unsuccessful participants.

Section II provides information on previous research relevant to this study. Section III describes the research method. The results of the experiment are presented in Section IV and discussed in Section V. Section VI explores the threats to validity. Finally, we summarize this study in Section VII.

II. Related work

A. Cognitive biases

The seminal work of Tversky and Kahneman from 1974 [1] introduced the concept of cognitive biases, describing three of them: *representatives*, *availability*, and *anchoring*. In their work, the researchers found that human beings rely heavily on heuristics during the decision-making process while often being blind to logical, statistical facts.

These findings later evolved into the dual process theory, describing the human mind as divided into Systems 1 and 2. System 1 performs fast and intuitive decisions that heavily rely on heuristics. Inversely, System 2 performs slow decisions that are logical and rule-based. By using energy mainly for important decisions (System 2), this natural phenomenon allows the human body to save precious energy when making simplified decisions (System 1). However, humans are prone to using the energy-saving System 1 for decision-making, even in cases that require rule-based thinking. In these cases, cognitive biases might occur [18].

As a counterpoint to biased reasoning, **rational reasoning** can be described based on the research by William James [19] as having two components: (1) perception of a specific piece of **factual information**, and (2) a **logical consequence** of this information [19].

Viewing software architecture as a set of design decisions is a well-established concept [8]. It has resulted in a substantial amount of research regarding issues related to ADM, such as documenting architectural decisions [20], models of architectural decisions [21], decision-making best practices [22], and human aspects of ADM [23], particularly cognitive biases [7].

The impact of cognitive biases on ADM can have severe consequences, such as designing sub-par solutions [24] or incurring dangerous architectural technical debt [25].

Based on previous research on cognitive biases in ADM [7], [25], [26], as well as the most often researched biases in software engineering [5], this study focuses on the following three cognitive biases:

Anchoring bias is the decision-maker's preference for initial information/ideas/solutions (which then become an 'anchor') [1]. In ADM, for example, architects may anchor on the first solution idea that comes to their minds, and they may refuse to change or adjust this solution, even when it becomes necessary [6].

Confirmation bias is the tendency to purposefully search and interpret information to verify one's beliefs [27]. For example, due to confirmation bias, an architect may have the belief that microservices are the best architectural style and insist on designing a microservice-based system, even when it is not appropriate [26].

Optimism bias is the overestimation of the probability of positive future outcomes [28]. This may affect architects by leading them to underestimate the maximum amount of requests that would be sent to a particular component [26].

B. Debiasing

Debiasing refers to the process of identifying and mitigating cognitive biases that may affect decision-making. Possible types of debiasing treatments can be described using the levels proposed by Fischhoff [12]. A- and B-level treatments consist of informing practitioners about cognitive biases (A) and how they may impact practitioners (B). Levels C and D require more organizational resources: a C-level treatment requires giving personalized feedback to each debiased person, and the D-level requires extensive long-term training.

In the field of software engineering, two notable C-level debiasing treatments were reportedly successful. Firstly, rationalizing development time estimates by Shepperd et al. [13], where researchers purposefully anchored software developers on pessimistic estimates to counter over-optimistic predictions. Secondly, in ADM, in their short paper, Borowa et al. [14] reported achieving a debiasing effect on a group of student participants. However, their experiment had several weaknesses that our study avoids:

1) The experiment employed the pretest-posttest design, which is known for numerous threats to validity [17]. For example, the debiasing effect may have not resulted from the workshop but from the student participants' learning while performing the pretest task.
2) The participants were all students, making it unclear whether the results would be the same for experienced practitioners.
3) The participants performed different tasks before and after the workshop, so comparing these may not have been

Authorized licensed use limited to: Bibliothekssystem Universitaet Hamburg. Downloaded on December 11,2025 at 16:51:00 UTC from IEEE Xplore. Restrictions apply.

TABLE I
PARTICIPANTS
W-WORKSHOP GROUP / C-CONTROL GROUP,
AGE - AGE IN YEARS,
IT EXP. - COMMERCIAL EXPERIENCE IN YEARS,
DOM - COMPANY DOMAIN,
SIZE - COMPANY SIZE: S = BELOW 100, M=101-500, L=501-5000, XL= OVER 5000

	No	Pair	C/W	IT exp.	Role	DOM	SIZE
Pilot	1	P1	W	6-10	Developer	Digital payment	S
	2	P1	C	1-2	Developer	Digital payment	S
	3	P2	W	11-20	Analyst	Digital payment	S
	4	P2	C	3-5	Product Manager	Digital payment	S
Main - practitioners	5	P3	W	11-20	Developer	Media	XL
	6	P3	C	>20	Developer	Media	XL
	7	P4	W	11-20	Developer	Marketing	M
	8	P4	C	11-20	Developer	Marketing	M
	9	P5	W	11-20	CTO	Finance	M
	10	P5	C	>20	Architect	Finance	M
	11	P6	W	6-10	Architect	Audio	L
	12	P6	C	>20	Sys. Eng.	Audio	L
	13	P7	W	11-20	Architect	Retail	XL
	14	P7	C	11-20	Architect	Retail	XL
	15	P8	W	3-5	Developer	Education	L
	16	P8	C	3-5	Developer	Education	L
	17	P9	W	11-20	Product Manager	Government	S
	18	P9	C	11-20	Coordinator	Government	S
	19	P10	W	1-2	Developer	Education	L
	20	P10	C	1-2	Developer	Education	L
Main - students	21	S1	W	-	-	-	-
	22	S1	C	1-2	Developer	Mobile	XL
	23	S2	W	<1	Developer	Electronics	XL
	24	S2	C	3-5	Developer	Utilities	L
	25	S3	W	1-2	Developer	Insurance	S
	26	S3	C	1-2	Developer	ERP	M
	27	S4	W	<1	Developer	Ecommerce	S
	28	S4	C	1-2	Developer	Security	XL
	29	S5	W	-	-	-	-
	30	S5	C	-	-	-	-
	31	S6	W	1-2	Tester	Web dev.	S
	32	S6	C	1-2	Developer	ERP	M
	33	S7	W	1-2	Pre-sales	Sales	XL
	34	S7	C	<1	Developer	Energetics	L
	35	S9	W	3-5	Developer	Security	L
	36	S8	C	1-2	DevOps	Earth Observation	M

reliable. For example, if designing one architecture was more difficult than the other, participants' bias occurrence count could be increased by this added difficulty.

4) The participants performed purely theoretical tasks by designing architectures for domains that they may not have had any prior knowledge about. This may have caused mistakes due to their lack of domain knowledge.
5) The debiasing effect was achieved by improving the amount of non-biased arguments, **but, despite the authors' efforts, the bias occurrence was not reduced**. We measured the bias occurrences using a similar method to Borowa et al. [14] and did achieve a bias reduction.

III. METHOD

We ran an experiment built on top of an existing debiasing workshop structure [14]. However, we improved the workshop and experiment, as explained in the following subsections.

The study was divided into two phases: (1) a pilot phase with two experiments, i.e., two pairs of practitioners from one company, and (2) the main experiment, i.e., with eight pairs of students and eight pairs of practitioners from seven different companies. While we present the pilot's results in this paper, the study plan underwent major changes between these phases, which led us to exclude the pilot's data from quantitative calculations (i.e., code count averages and p-values). The differences between the pilot and the main study are also explained in the respective subsections.

A. Sample

We recruited two types of participants: students and practitioners. We define students as individuals studying to obtain a computer science-related degree who had performed major ADM tasks in an industrial setting. Practitioners, on the other hand, are individuals who have performed major ADM tasks in an industrial setting. Student pairs discussed projects from their coursework during this study, while practitioners discussed real-life industrial projects. However, it is worth noting that most student participants were also young practitioners with some industrial experience.

The first author recruited student participants from Warsaw University of Technology through convenience sampling, by advertising the search for participants through their institution's channels. To qualify for participation, a pair of students had to have developed a software system from scratch as part of a team during their coursework, i.e., they had to have taken part in requirements gathering, architectural design, implementation, and testing of a system with at least three major components. Sixteen students applied and met these criteria. Students could obtain extra credit in a software architecture course by participating, though only two students used this opportunity.

We recruited practitioner participants through convenience sampling from our network in three countries (Brazil, Germany, and Poland) and offered no rewards to practitioner participants. This was the most appropriate method of acquiring participants for two reasons:

- The experiment required participants to discuss existing systems used by their companies. As such, participants needed to trust the researchers to handle sensitive data appropriately, which allowed us to face real-world designs instead of theoretical examples.
- We needed to find pairs of practitioners who had worked together on developing the same system and who had similar roles. Explaining the nuances to the participants required personal discussion with one of the researchers.

The participants were informed beforehand that the experiment was related to improving ADM and about the overall steps of the experiment. However, they were not informed about cognitive biases, had no access to the workshop materials, and did not know the Step 4 task of the experiment. No pair of participants had the chance to contact another pair to disclose such information. Table I presents basic information about the participants.

B. The experiment

We conducted an experiment based on an existing debiasing workshop structure [14] with several modifications. First, we replaced a debiasing practice that was having one person monitor the discussion for statements influenced by

Authorized licensed use limited to: Bibliothekssystem Universitaet Hamburg. Downloaded on December 11,2025 at 16:51:00 UTC from IEEE Xplore. Restrictions apply.

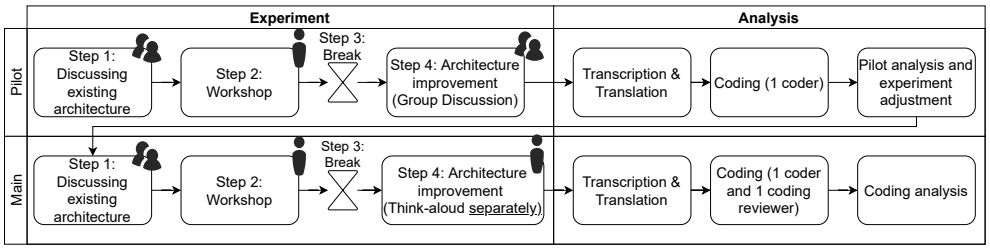


Fig. 1. Study steps

confirmation-bias, since the previous workshop's authors admit that their participants were rarely able to use this technique at all [14]. Instead, our participants were taught to list multiple solution options for their architectural decisions.

Second, we performed an experiment with a matched pair design [29] instead of the pretest-posttest design. As such, we compared the results for the same architectural task done by matched pairs: (1) where one had attended the workshop (workshop group), and (2) the other had not (control group). By contrast, in the experiment of [14], researchers compared different tasks done before and after the workshop by the same students. The matched pair design, combined with recruiting pairs of individuals who had worked on a project together, allowed us to explore **whether the discussed software systems could have benefited from having an individual participate in the debiasing workshop,** while enhancing internal validity by randomizing each pair's split into workshop/control groups.

Third, the architectural designs used in the experiment came from students' coursework projects or real-life cases from practitioners' companies, not hypothetical scenarios.

The experiment itself consisted of (see Figure 1):

1) *Step 1: Discussing existing architecture (up to 30 min.):* Both participants took part in a meeting with a researcher. First, they were asked to fill out a questionnaire to gather demographic data. Then, they were asked to explain the architecture of a system they had both worked on previously. In the case of student participants, they had to discuss a university project where they performed the whole development process, from gathering requirements through architectural design, implementation, and testing. Practitioner participants discussed real-life industrial projects. Participants drew a simple "boxes and arrows" diagram of the architecture and gave the researcher the following information: (1) the overall idea and context of the system, (2) the system's main components, (3) relationships between components, and (4) the technologies used. This way, we ensured that both participants knew the architecture at the same level.
2) *Step 2: Debiasing workshop (around 60 min.):* During this step, the workshop group participant took part in the debiasing workshop, as described in Section III-C. The control group participant had a break.
3) *Step 3: Break:* Research on cognitive biases [18] notes that rational logic-based thinking is physically exhausting. To avoid tiring the workshop participants more than the control participants, there was always a break before the last step of the experiment. The length varied but, was always long enough for at least one meal.
4) *Step 4: Architecture improvement (up to 30 min.):* Participants were asked to identify issues with the Step 1 system's architecture. They were asked to propose possible improvements to resolve these issues. The participants were requested to note the changes on the boxes-and-arrows diagram from Step 1.
   In the pilot run with two pairs, we allowed participants to discuss together. However, in one pilot experiment, we noticed that this approach caused a participant to omit their opinions. This was caused by the fact that the other participant, their superior, dominated the discussion.
   In the main experiment, we changed our data gathering approach to a "think-aloud protocol" session [30]. This method was suggested by Razavian et al. [9] for studying behavioral factors in ADM. Accordingly, the participants proposed their improvements separately (i.e. one after another, in a separate room, with no option to hear each other) in the subsequent sessions, to avoid interferences between participants. The participants were asked to voice their thoughts, i.e., "think aloud", during the whole session. We could not use evaluation methods such as questionnaires or having the participant write down their argumentation since cognitive biases occur during an individual's thought process, and their occurrence may not be visible when the participants have additional time to curate their answers.

The experiment's Step 4 was recorded, with the audio recordings transcribed for further analysis. All the experiment sessions were performed in the native language of the participants by a native-speaking researcher, who subsequently translated the transcript to English to make it accessible to all the authors. We allowed participants to choose between on-site and online participation. Four experiments were performed during a personal meeting (two pilot and two practitioner main), and the rest were held online (all main). All the experiments for each pair were performed separately, so no step was performed with many pairs present in the same room at the same time. The experiment plan is available

Authorized licensed use limited to: Bibliothekssystem Universitaet Hamburg. Downloaded on December 11,2025 at 16:51:00 UTC from IEEE Xplore. Restrictions apply.

as part of the additional material [31].

C. Debiasing Workshop

Our workshop was an intervention that included not only informing about cognitive biases and the specific effects that they may have on the participants but also trained the participants through a teaching experience with personalized feedback. Therefore, it is a C-level debiasing intervention on Fischoff's debiasing scale [12].

Overall, the workshop comprised (1) a short lecture about cognitive biases and the work of Daniel Kahneman and Amos Tversky in general [1] [18], (2) an explanation of how cognitive biases affect ADM in particular, (3) a design session, during which the participant was supposed to design a solution for a fictional architecture task and make use of the debiasing techniques. We taught the following debiasing techniques to the participants:

1) *Generating multiple solution options:* This is a new debiasing technique that aims to counter *anchoring* and *confirmation biases*. The main impact of both these biases is limiting the scope of considered solution alternatives [26], i.e., anchoring makes individuals fixated on one solution idea [6] and confirmation bias makes them unlikely to change their initial decision based on evidence [32]. As such, it seems prudent to explicitly list solution alternatives before architects believe that one single solution is the best option before considering any others.
   Listing multiple solutions is also a basic reasoning technique suggested by many ADM researchers [22] [33] [34] for improved reasoning.
2) *Listing at least one drawback of the solution alternative:* This technique is targeted towards countering the effects of *anchoring*, since *anchoring* often makes practitioners over-focused on one solution's advantages [14]. Unlike risks, drawbacks always impact the solution, e.g., a component's license is expensive.
3) *Listing at least one risk associated with the solution alternative:* Since individuals are prone to over-optimistic predictions, explicitly discussing risks is targeted towards countering *optimism bias* [14]. Risks, unlike drawbacks, have a probability associated with their occurrence. This means there is a possibility that the risk may not occur at all.

The researcher leading the workshop demonstrated the debiasing techniques using examples and actively assisted the participants in the design session when they were struggling with the techniques.

D. Data Analysis

For the data analysis procedure, we used the hypothesis coding technique [35], which involved creating a set of codes before the coding process. The codes that we used are listed in Table II and were prepared based on the same coding scheme used by Borowa et al. [14]. A coding guide with examples from this study's participants is available online [31].

Recordings from the sessions were transcribed and translated to English by the researcher organizing the workshop, then coded by another author and, for the main experiment, the coding was reviewed by an additional author.

In the pilot phase, the coder was unaware of which participant was the control/workshop one. The main experiment, as suggested by Ericsson and Simon [30], employed **context-free coding** as follows:

1) The workshop organizer divided the transcripts into segments (one for each architectural decision). This was a natural split since the participants usually discussed one possible change to the architecture after another.
2) For each pair of participants, the control and workshop groups' segments were randomly ordered.
3) Another researcher coded the segments, without knowing whether a particular segment belonged to the workshop or control group participant.
4) An additional researcher reviewed the coding and discussed change proposals with the original coder until a consensus was reached on all codes.
5) The workshop organizer revealed the participants' information to create a summary of each experiment's coding.

We anticipated that a coder might unconsciously try to find codes proving the research hypotheses. The context-free coding method aims to decrease the possibility of this occurring.

We evaluated our coding as follows: (1) We calculated the average of all codes for a particular measurement for each group of participants. (2) We checked whether the measurements for cognitive biases and bias-impacted statements decreased. (3) We checked whether the measurements of debiasing techniques and non-biased statements increased.

In the work of Borowa et al. [14], the authors assumed that if most arguments used while discussing one decision were non-biased, then the decision was non-biased. We have not followed this assumption since one argument (biased or non-biased) can be decisive in decision-making.

In order to determine whether the debiasing intervention was successful, we defined a set of research hypotheses (H_R):

- In the case of **values that the workshop strived to decrease** (biased arguments/counterarguments, cognitive biases), the research hypothesis (H_R) was that the measurements with the workshop (workshop group) were smaller than without it (control group). Our null hypothesis (H_0) was that the control group measurements were smaller than or equal to the workshop group measurements.
- The opposite was the case with **values that the workshop tried to increase** (not biased arguments/counterarguments, use of debiasing techniques). In this case, our research hypothesis (H_R) was that the measurements with the workshop (workshop group) were bigger than without it (control group). In these cases, the null hypothesis (H_0) was that the control group measurements were bigger (or equal to) the workshop group measurements.

To examine which results are statistically significant and can be considered as accepted with a high confidence level,

Authorized licensed use limited to: Bibliothekssystem Universitaet Hamburg. Downloaded on December 11,2025 at 16:51:00 UTC from IEEE Xplore. Restrictions apply.

TABLE II
CODING SCHEME: ADAPTED FROM [14]

Code	Code meaning	Description
Arg	**Argument**	A statement in **support** of a possible solution alternative. E.g. "MS SQL would be a better choice than Oracle since it is the cheaper solution."
Carg	**Counterargument**	A statement in **opposition** of a possible solution alternative. E.g. "This solution requires turning off the access to users during maintenance".
Anch	**Anchoring**	A statement suggesting that the participant is impacted by **anchoring**. E.g. "Django would be the best choice" when no additional contextual information suggests why Django was chosen or whether alternatives were considered.
Conf	**Confirmation bias**	A statement suggesting that the participant is impacted by **confirmation bias**. E.g. "We already decided on a centralized system, so let's leave it like this," when issues with this choice were being discussed.
Opt	**Optimism bias**	A statement suggesting that the participant is impacted by **optimism bias**. E.g. "I am sure we can easily connect to some external REST API to handle the payment," without researching or having prior knowledge about payment systems.
Ddraw	**Decision's drawback**	Use of the **anti-anchoring technique**, i.e., a statement where the participant discusses a drawback of the solution alternative. E.g. "Splitting this service into microservices would require additional work on orchestration."
Dmulti	**Decision with multiple alternatives**	Use of the **anti-confirmation bias and anti-anchoring technique**, i.e., a statement where the participant mentions more than one solution alternative. E.g. "We could consider various free RDBMS: like MySQL, PostgreSQL, Maria DB or even SQLite."
Drisk	**Decision's risk**	Use of the **anti-optimism bias technique**, i.e. a statement where the participant discusses a risk associated with a solution alternative. E.g. "This SSH Server is a possible bottleneck in the system and may stop responding due to too many requests."

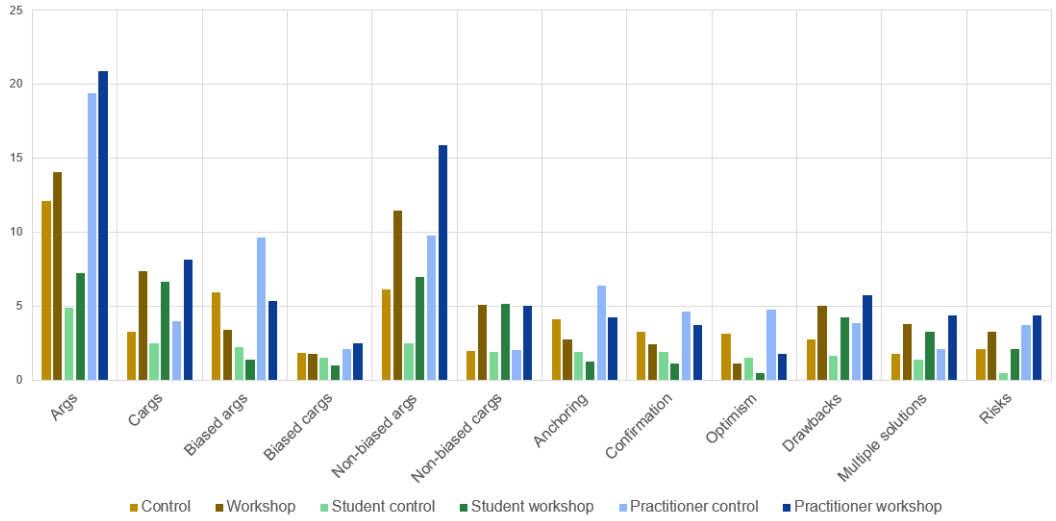


Fig. 2. Average amount of codes for control and workshop group for (1) All participants, (2) students, (3) practitioners.

TABLE III
(NON-)BIASED ARGUMENTS AND COUNTERARGUMENTS (B – BIASED, NB – NON-BIASED)

		Control				Workshop						
		Arguments		Counterarg.		Arguments		Counterarg.				
	Pair	B	NB	B	NB	B	NB	B	NB	**Control Biased statements [%]**	**Workshop Biased statements [%]**	**Difference Biased statements [pp]**
Pilot	P1	5	2	1	2	4	0	1	6	60%	38%	-22
	P2	8	1	0	1	1	3	0	1	80%	17%	-63
Main: practitioner	P3	12	10	4	1	8	8	2	9	59%	36%	-24
	P4	3	6	1	1	11	5	9	4	36%	67%	30
	P5	14	22	1	4	3	20	0	1	37%	11%	-26
	P6	9	10	2	3	5	13	3	7	46%	26%	-20
	P7	24	17	6	4	10	54	3	13	59%	15%	-43
	P8	3	1	0	1	1	10	0	1	60%	8%	-52
	P9	7	9	2	1	3	11	2	4	47%	24%	-24
	P10	5	3	1	1	2	6	1	1	60 %	30 %	-30
Main: student	S1	1	3	1	1	1	4	0	8	33 %	8 %	-25
	S2	1	3	1	1	1	8	0	1	33 %	10 %	-23
	S3	1	2	2	1	1	15	2	11	50 %	10 %	-40
	S4	2	1	2	0	1	5	1	3	80 %	20 %	-60
	S5	3	1	3	0	3	3	3	5	86 %	43 %	-43
	S6	2	1	0	5	4	4	1	2	25 %	45 %	20
	S7	5	7	2	5	0	11	0	4	37 %	0 %	-37
	S8	3	2	1	2	0	6	1	7	50 %	7 %	-43

we performed the non-parametric Wilcoxon Signed Rank test [36] for each of the measurements and the percentage of biased statements (i.e. arguments and counterarguments). This test is used to measure whether a statistical difference can be observed between small paired (dependent) data samples. We consider the control group and workshop group

Authorized licensed use limited to: Bibliothekssystem Universitaet Hamburg. Downloaded on December 11,2025 at 16:51:00 UTC from IEEE Xplore. Restrictions apply.

TABLE IV
OCCURRENCES OF COGNITIVE BIASES (ANCH – ANCHORING, OPT – OPTIMISM BIAS, CONF – CONFIRMATION BIAS)

		Control			Workshop			**Control**	**Workshop**	**Difference**
	Pair	ANCH	CONF	OPT	ANCH	CONF	OPT	**Bias sum**	**Bias sum**	**Bias sum**
Pilot	P1	6	3	1	1	1	1	10	3	-7
	P2	7	3	1	1	0	0	11	1	-10
Main: practitioner	P3	9	7	2	7	4	0	18	11	-7
	P4	3	1	2	12	11	0	6	23	17
	P5	10	3	8	1	2	3	21	6	-15
	P6	9	3	0	5	3	1	12	9	-3
	P7	6	21	23	6	6	4	50	16	-34
	P8	1	0	3	0	0	2	4	2	-2
	P9	7	2	0	2	4	2	9	8	-1
	P10	6	0	0	1	0	2	6	3	-3
Main: student	S1	0	1	1	0	1	1	2	2	0
	S2	0	1	1	1	0	0	2	1	-1
	S3	3	2	3	1	1	1	8	3	-5
	S4	2	1	1	1	1	1	4	3	-1
	S5	4	3	1	4	3	1	8	8	0
	S6	2	1	1	2	3	0	4	5	1
	S7	4	4	0	0	0	0	8	0	-8
	S8	0	2	4	1	0	0	6	1	-5

participant pairs to be dependent since they both shared an understanding of the particular architecture and performed the task of discussing improvements for the same systems. We accept H_R as statistically significant if the p-value is smaller than 0.05, i.e., there is a less than 5% chance that H_R is false.

IV. RESULTS

In this section, we present the results of our analysis. Due to the change in data-gathering methods, the results from the pilot study were not included in any calculations (p-values, overall code sum/average for the control/workshop group), but raw values are presented in this section as well.

Figure 2 showcases the average code counts for each measurement totaled across all participants, i.e. for each code, we summed the number of occurrences in the transcript for each participant and calculated the averages for participant groups. When considering only the code averages, all measured values changed in accordance with the research hypothesis H_R. That is: (1) the amount of arguments and counterarguments increased, (2) the amount of biased arguments and counterarguments decreased, (3) the amount of non-biased arguments and counterarguments increased, (4) the amount of bias occurrences decreased, (5) the amount of debiasing techniques use increased.

A. Statistical Significance

The p-values calculated using the Wilcoxon Signed Rank test [36] are presented in Table V. We calculated the p-values for all measurements that we coded and the percentage of biased statements. Most observed measurement changes were statistically significant, with the exception of (1) increased argument amount, (2) decreased amount of biased counterarguments, (3) decreased amount of confirmation bias occurrences, and (4) increased amount of the "listing risks" technique uses.

B. Arguments and Counterarguments

We coded as arguments the statements that supported choosing a particular architectural solution, e.g. Participant 6 said "(...) what is special about this architecture is that

TABLE V
P-VALUES FOR EACH MEASUREMENT
W-WORKSHOP GROUP / C-CONTROL GROUP,
AVG - AVERAGE OF CODE SUMS,
H_R - RESEARCH HYPOTHESIS

Measurement	C-AVG	W-AVG	p-value	(H_R)
Arg	12.13	14.06	0.1619	W >C
Carg	3.25	7.28	**0.0022**	W >C
Anch	4.13	2.75	**0.0414**	W <C
Conf	3.25	2.24	0.2062	W <C
Opt	3.13	1.13	**0.0404**	W <C
Biases sum	10.50	6.31	**0.0108**	W <C
Biased Args	5.94	3.38	**0.0205**	W <C
Biased Cargs	1.81	1.71	0.1482	W <C
Non-biased Args	6.13	11.44	**0.0016**	W >C
Non-biased Cargs	1.94	5.06	**0.0057**	W >C
% Biased statements	49,90%	23,13%	**0.0008**	W <C
Ddraw	2.75	5.00	**0.0123**	W >C
Dmulti	1.75	3.81	**0.0018**	W >C
Drisk	2.13	3.25	0.0692	W >C
Techniques use sum	10.57	14.57	**0.0038**	W >C

everything is dynamically configurable." Counterarguments are statements that give reasons against choosing a solution, e.g. Participant 2 notes price "(...) AWS itself is another third party service that is paid as well." Table III presents the numbers of arguments and counterarguments for each pair of participants. When combined with Figure 2, showing the average of each type of statement in the control and workshop groups, it allows us to make the following observations:

- The number of arguments and counterarguments used was higher for workshop participants.
- Workshop participants used fewer biased arguments and counterarguments overall.
- Practitioners used more arguments than students in both workshop and control groups.
- The number of practitioners' biased counterarguments slightly increased after the workshop.

Additionally, based on the data shown in Table V, we can distinguish a set of statistically significant results related to arguments and counterarguments. **The following differences between participant pairs were statistically significant: (1) increased counterargument amount, (2) increased non-biased statements amount for both arguments and**

Authorized licensed use limited to: Bibliothekssystem Universitaet Hamburg. Downloaded on December 11,2025 at 16:51:00 UTC from IEEE Xplore. Restrictions apply.

counterarguments, (3) decreased amount of biased arguments. However, the increase in arguments amount in favor of a solution was not significant, nor was the decrease in biased counterargument amount. Additionally, when looking at the individual values per pair, two participants, one student (pair S6), and one practitioner (pair P4), used more biased statements, despite the workshop's intervention.

C. Cognitive Biases

We coded biased statements using three codes. We identified anchoring in arguments favoring solutions without factual data or solution alternatives, e.g., Participant 14 said the following, without providing additional rationale: "I would move all this work on data, manipulation of this data, certainly, to the server." Second, we used the confirmation bias code when participants avoided discussing improvements despite identifying problems; Participant 7 remarked: "Even if we were to talk about splitting up the middleware or converting it to microservices or other structures, I wouldn't do any of that because that's what [colleague] and I have defined as the actual system." Finally, optimism bias was coded for overly optimistic predictions, e.g., Participant 36 proposed parallel processing, stating "Surely this would also speed up the whole process," without considering communication overheads.

Table IV presents the number of biased statements from each participant pair. Figure 2 summarizes of average bias occurrences in each of the control and workshop groups. Overall, the workshop group participants were less impacted by each of the researched biases. **The results were statistically significant (see Table V) in the cases of decreasing anchoring and optimism bias occurrences.**

However, two differences between practitioners and students are noticeable: **(1) the average bias occurrence amount (both control and workshop) was higher in the case of practitioners**, and **(2) decreased bias occurrence was higher in practitioners than students**. Overall, the occurrence of biases of three student pairs (S1, S2, S5) did not decrease, which was the case for only one practitioner pair (P4).

D. Debiasing Techniques

Figure 2 shows the usage of each technique in the workshop and control groups. Overall, **the workshop participants used each of the three techniques more frequently than the control group participants**. This was statistically significant (see Table V) in the cases of "discussing drawbacks" and "listing multiple solutions", but not for the "discussing risks" technique. Two participants' use of debiasing techniques did not improve after the workshop, one practitioner (pair P5) and one student (pair S6). The code count for each participant's use of debiasing techniques is part of the additional material [31].

E. Decisions

We did not attempt to influence the decision amount through this experiment, so no p-values were calculated. However, during the experiment, we noticed that the amount of discussed decisions varied, so we decided to explore whether decision count could possibly be linked to cognitive bias susceptibility. In most cases (11 out of 18), the workshop participants discussed more architectural decisions than their control group collaborators. Pair P4 was an outlier in decision count, with the workshop participant discussing 19 decisions against 9 discussed by the control participant, which was the biggest difference of all the pairs. This pair is the only practitioner pair where the workshop participant's bias occurrences and percentage of biased statements were higher than their control group counterparts. Detailed decision count data is presented in the additional material [31].

V. Discussion

In this section, we discuss our results in the context of (1) comparison with previous work on debiasing ADM, (2) differences between student and practitioner participants, (3) a set of teaching suggestions for educators, (4) implications for researchers.

A. Comparison with Previous Work

Compared to the only study reporting an effective debiasing ADM intervention by Borowa et al. [14], performed with solely student participants, **our results show more positive outcomes**. In their study, the amount of non-biased arguments and counterarguments and the use of debiasing techniques were improved. Our study identified similar outcomes and additionally showed: **(1) a decrease in bias occurrences (optimism bias and anchoring decreased significantly) and (2) a significant decrease of biased arguments in support of a solution**. This may be the result of two factors: (1) improved experiment and workshop design and (2) the practitioner participants' stronger response to the debiasing workshop (i.e. bigger decrease in bias occurrences) in comparison to student participants.

B. Comparision between Practitioners' and Students'

We discovered that there are notable differences in how students and practitioners reason about architectural decisions and respond to the debiasing workshop.

First, **practitioners were much more skilled overall in using arguments in support of a solution**, in both control and workshop groups, i.e. on average, they used almost twice as many arguments in favor of solutions than students. We suspect this may be due to the practitioners' having the preexisting skill of arguing in favor of their chosen solutions since they most likely had to do this repeatedly in their workplace. In contrast, students may find the formulation of any arguments, both for and against architectural solutions, new due to their limited experience.

Second, **the number of biased counterarguments slightly increased in the case of practitioners**, which was not the case with students. This may be because practitioners had more previous negative experiences (e.g. failed projects), so they had a higher tendency to prematurely dismiss solutions that they had a personal negative experience with.

Authorized licensed use limited to: Bibliothekssystem Universitaet Hamburg. Downloaded on December 11,2025 at 16:51:00 UTC from IEEE Xplore. Restrictions apply.

Thirdly, **practitioners from both control and workshop groups were overall more impacted by biases than students**, i.e., the average bias occurrence measurement for all biases was higher in the case of practitioners. As a result, practitioners' decrease in bias occurrence was more extreme than in the case of students. Having industrial experience may actually make practitioners more prone to the impact of cognitive biases. This higher bias susceptibility may be due to practitioners' higher attachment to the solutions that they worked on. In the case of a finished student project, the student can freely critique the solution since there is no danger of losing customers or damaging their reputation. Conversely, practitioners spend years polishing their systems, which they have to sell to customers. This strong attachment to their solutions is further evidenced by practitioners' frequent use of arguments to support their choices and the non-significant decrease in the occurrences of confirmation bias—i.e., the tendency to dismiss information that contradicts their beliefs.

Similarly, in the software testing domain, Calikli et al. [37] found that active software testers were more prone to confirmation bias than ex-testers or trained students. They attributed this to habits formed by routine tasks and suggested breaking routines by periodically performing varying tasks. A similar approach could also benefit software architects, such as participating in non-architectural activities.

C. Recommendations for Educators

Overall, the workshop was a successful debiasing intervention and we recommend its use by educators. However, we additionally investigated the participants who did not perform as expected, to explore what educators might take into account to improve this debiasing intervention.

Teaching suggestion 1: Debiasing might only be effective when team members of all seniority levels participate in a debiasing intervention. As shown by Participant 3 (pair P2 from the pilot), who discussed only two design decisions, group discussion can be heavily impacted by pressure from senior colleagues. As such, socio-cultural factors associated with group decision-making [38] make it crucial to perform the intervention on all individuals, not only a few team members, who may be unable to pass their knowledge about biases to their colleagues.

Teaching suggestion 2: Avoid discussing too many decisions. Pair P4 was the only practitioner pair where the workshop participant's bias occurrences were higher than their control group colleague's. The only value that seems to distinguish this pair is that the workshop participant discussed many more decisions than the control group participant. This difference was the highest of all the pairs. As such, discussing too many decisions in a short time span may increase the likelihood of cognitive bias and should be avoided. This might result from a heightened cognitive load [39] experienced by the participant, who attempted to discuss a large number of design decisions swiftly. Psychological research shows that cognitive load impacts *anchored* time estimates [40]. Thus, we suspect that it may also have impacted our participant. Especially since anchoring was the main bias impacting this participant.

Teaching suggestion 3: Address overconfidence to counter susceptibility to cognitive biases. Student pair S6 was the only student pair where the workshop group participant did not perform better in almost any regard (bias occurrences, biased statements, and debiasing techniques). This student's case was quite distinct because, during the workshop (before which we did not disclose that the study is about cognitive biases), they informed the researcher conducting the workshop that they had read Kahneman's famous book about cognitive biases [18]. As such, this participant had a high confidence in their knowledge of cognitive biases. Similarly, the workshop participant from pair P5 was the only practitioner who used a smaller amount of debiasing techniques than his control group colleague. Notably, the workshop group participant was the company's CTO. During the experiment, the researcher noticed that the attitude of the workshop participants indicated that they considered the workshop's contents trivial. The participant's confidence level was most likely heightened due to their high position in the company, where they were previously the head architect for many years. Therefore, we suggest that educators performing debiasing interventions should warn unusually confident individuals about their possibly higher susceptibility to biases.

Teaching suggestion 4: Limit low-quality counterarguments. In the case of practitioners, a slight increase in the use of biased counterarguments by workshop participants occurred. This may be because the workshop focused on negative factors, such as decision-related risks and drawbacks. As such, some participants may have over-focused on creating numerous counterarguments with less care about their quality. Educators may ask participants to limit the number of counterarguments to a set number of crucial ones during the workshop.

D. Implications for Researchers

Our study provides researchers with a valuable starting point, showcasing how a successful ADM debiasing intervention can be performed. However, we strongly advise that researchers consider the following issues. First, since confirmation bias was the only cognitive bias that did not decrease significantly, researchers should focus on finding methods of countering its impact. Since we suspect that routine, previous negative experiences and overconfidence may be the source of this problem, debiasing methods could consider routine interruption, discussing previous negative experiences, and warning overconfident individuals. Second, since practitioners' bias levels were higher and attachment to an existing architectural solution seems to be a strong factor affecting biases, future studies should focus on debiasing practitioners on real-life industrial systems. Finally, since the use of the "discussing risks" technique did not increase significantly, the workshop did not fully teach participants how to perform it. Researchers could attempt to change the workshop's contents related to teaching this technique or even consider replacing it with a different technique altogether.

Authorized licensed use limited to: Bibliothekssystem Universitaet Hamburg. Downloaded on December 11,2025 at 16:51:00 UTC from IEEE Xplore. Restrictions apply.

VI. Threats to validity

Our threats to validity are described based on the guidelines created by Wohlin et al. [41]:

Construct Validity – It is important to note that every human's thought process is nuanced and multifaceted, so the "biased statements" that we measured are a simplification of this process. As such, there exists the threat that our study over-simplified this complex phenomenon by, for example, not taking into account how several biases may impact decision-making simultaneously, and the relationships between them.

Internal Validity – First, our sample may be impacted by selection bias since our participants were volunteers. It is possible that individuals less interested in architectural decision-making may be impacted by cognitive biases and the workshop differently. Second, student participants discussed systems from a university project, while practitioners described real-life systems. Therefore, when comparing their outcomes, it must be noted that some result differences may be rooted in the nature of university projects and not the participant's experience level. We take this into account when discussing our results (see Section V-B). This was unavoidable since there was no different way of testing how students reason if they had no experience in designing real-world architectures. Third, while our experiment, which employed a matched pair design, did include randomized splitting into the control/workshop group, internal validity could have been enhanced even further by a fully random assignment design [42]. However, in that case, all the participants would have had to discuss the same task, making it impossible for them to use their knowledge about the system fully, thus limiting external validity. Finally, a key limitation of the experiment lies in the design of the control group. The control group participants were given no alternative intervention and simply took a break. This raises the possibility that the observed improvements may stem from any intervention rather than the debiasing workshop. Future studies should include a placebo [29] intervention for the control group participants to mitigate this threat.

Conclusion Validity – We have the threat of a relatively low number of experiments executed, which is due to the fact that it was hard to find participants, especially practitioners, willing to invest over two hours in a scientific experiment. Finding participants was particularly challenging since our experiment required (1) pairs of participants who worked together on the same project and (2), in the case of practitioners, sensitive data about their real-life architecture. This low sample size places a risk on the statistical significance of the findings. To counteract this threat, we performed the non-parametric Wilcoxon Signed Rank Test, which makes it possible to check the statistical significance of small paired samples. Still, a larger sample would provide more confidence in the findings.

External Validity – As common for a controlled experiment, the external validity is hampered by the strictly controlled setting. However, we tried to perform the experiment in the participants' preferred environment to lessen this effect. Additionally, the student participants have less domain knowledge than practitioners, which could lead to poor design decisions. We attempted to lessen this effect by having students discuss a project that they designed and implemented themselves during their coursework, which means that they had to analyze the project's domain beforehand. Finally, we could not research the long-term effect of the debiasing workshop. The workshop's beneficial impact may lessen over time. Future work could focus on studying the long-term effects of our workshop.

VII. Conclusion

We performed a debiasing workshop in order to explore its impact on students and practitioners, i.e. individuals at various career stages. The design of our study was based on the structure by Borowa et al. [14], with some adaptations (as explained in Section III). We divided participants into separate workshop (treatment) and control groups as it is common in similar studies [10], [23], striving for a higher level of validity than in a pretest-posttest experiment [17]. We also employed the think-aloud protocol study method suggested by Razavian et al. [9] as appropriate for researching behavioral aspects of ADM.

The research question that we aimed to answer was:

> **RQ: How does the proposed debiasing architectural decision-making workshop influence students and practitioners?**

Our main contribution includes the design and evaluation of a debiasing workshop. Through this evaluation, we found:

1) The use of the proposed debiasing techniques that were taught increased. In particular, the use of"discussing drawbacks" and "listing multiple solutions" techniques' increased significantly.
2) The occurrences of cognitive bias for all three researched biases (anchoring, confirmation bias, and optimism bias) decreased. This decrease was significant in the case of anchoring and optimism bias.
3) Practitioners were commonly more influenced by all researched biases than students, and as such, the improvement achieved in their case was more prominent.
4) A set of teaching suggestions based on factors that negatively impacted some of the study's participants.

Practitioners and educators intending to perform such a debiasing workshop can use the teaching materials we prepared for this study either directly or using them as a basis for their own materials [31]. **Researchers** may make use of this study as a basis for creating improved debiasing interventions and evaluate them by employing our experimental design.

For future work, we propose to focus on (1) longitudinal studies in a real-world setting to ensure that the debiasing treatment has a long-term impact, (2) further improvements meant to reduce the impact of confirmation bias, (3) an in-depth qualitative analysis of the biases in ADM.

VIII. Data availability

The experiment plan, workshop plan, workshop slides, participant questionnaire, coding results, and a coding guide with examples from the participants are available online [31].

Authorized licensed use limited to: Bibliothekssystem Universitaet Hamburg. Downloaded on December 11,2025 at 16:51:00 UTC from IEEE Xplore. Restrictions apply.

REFERENCES

[1] A. Tversky and Kahneman Daniel, "Judgment under uncertainty: Heuristics and biases," *Science*, 1974.

[2] G. R. Norman, S. D. Monteiro, J. Sherbino, J. S. Ilgen, H. G. Schmidt, and S. Mamede, "The causes of errors in clinical reasoning: cognitive biases, knowledge deficits, and dual process thinking," *Academic Medicine*, vol. 92, no. 1, pp. 23–30, 2017.

[3] S. Chattopadhyay, N. Nelson, A. Au, N. Morales, C. Sanchez, R. Pandita, and A. Sarma, "A Tale from the Trenches : Cognitive Biases and Software Development," in *International Conference on Software Engineering (ICSE)*, 2020, pp. 654–665.

[4] W. Stacy and J. Macmillan, "Cognitive Bias in Software Engineering," *Communications of the ACM*, vol. 38, no. 6, pp. 57–63, 1995.

[5] R. Mohanani, I. Salman, B. Turhan, P. Rodriguez, and P. Ralph, "Cognitive Biases in Software Engineering: A Systematic Mapping Study," *IEEE Transactions on Software Engineering*, vol. 5589, no. c, 2018.

[6] A. Tang, "Software designers, are you biased?" in *Proceedings of the 6th international workshop on sharing and reusing architectural knowledge*, 2011.

[7] H. van Vliet and A. Tang, "Decision making in software architecture," *Journal of Systems and Software*, vol. 117, pp. 638–644, 2016.

[8] A. Jansen and J. Bosch, "Software architecture as a set of architectural design decisions," in *5th Working IEEE/IFIP Conference on Software Architecture (WICSA'05)*. IEEE, 2005, pp. 109–120.

[9] M. Razavian, B. Paech, and A. Tang, "Empirical research for software architecture decision making: An analysis," *Journal of Systems and Software*, vol. 149, pp. 360–381, 2019.

[10] M. Razavian, A. Tang, R. Capilla, and P. Lago, "In two minds: how reflections influence software design thinking," *Journal of Software: Evolution and Process*, vol. 28, no. 6, pp. 394–426, 2016.

[11] A. Tang, F. Bex, C. Schriek, and J. M. E. van der Werf, "Improving software design reasoning–A reminder card approach," *Journal of Systems and Software*, vol. 144, no. April 2017, pp. 22–40, 2018.

[12] B. Fischhoff, "Debiasing. judgment under uncertainty: Heuristics and biases," *Judgment under uncertainty: Heuristics and biases*, pp. 422–444, 1982.

[13] M. Shepperd, C. Mair, and M. Jørgensen, "An Experimental Evaluation of a De-biasing Intervention for Professional Software Developers," in *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, 2018.

[14] K. Borowa, M. Jarek, G. Mystkowska, W. Paszko, and A. Zalewski, "Debiasing architectural decision-making: a workshop-based training approach," in *European Conference on Software Architecture (ECSA)*, 2022, pp. 1–8.

[15] M. Galster and D. Weyns, "Empirical research in software architecture—perceptions of the community," *Journal of Systems and Software*, vol. 202, p. 111684, 2023.

[16] S. Baltes and P. Ralph, "Sampling in software engineering research: A critical review and guidelines," *Empirical Software Engineering*, vol. 27, no. 4, p. 94, 2022.

[17] T. R. Knapp, "Why is the one-group pretest–posttest design still used?" *Clinical Nursing Research*, vol. 25, no. 5, pp. 467–472, 2016.

[18] D. Kahneman, "Think fast, think slow," *Farrar, Straus and Giroux, New York*, 2011.

[19] W. James, *Principles of Psychology (1980)*. Henry Holt and Company, 1931.

[20] U. Van Heesch, P. Avgeriou, and R. Hilliard, "A documentation framework for architecture decisions," *Journal of Systems and Software*, vol. 85, no. 4, pp. 795–820, 2012.

[21] O. Zimmermann, J. Koehler, F. Leymann, R. Polley, and N. Schuster, "Managing architectural decision models with dependency relations, integrity constraints, and production rules," *Journal of Systems and Software*, vol. 82, no. 8, pp. 1249–1267, 2009.

[22] A. Tang and R. Kazman, "Decision-Making Principles for Better Software Design Decisions," *IEEE Software*, vol. 38, no. 6, pp. 98–102, 2021.

[23] A. Tang, M. Razavian, B. Paech, and T. M. Hesse, "Human Aspects in Software Architecture Decision Making: A Literature Review," *Proceedings - 2017 IEEE International Conference on Software Architecture, ICSA 2017*, pp. 107–116, 2017.

[24] A. Manjunath, M. Bhat, K. Shumaiev, A. Biesdorf, and F. Matthes, "Decision Making and Cognitive Biases in Designing Software Architectures," *Proceedings - 2018 IEEE 15th International Conference on Software Architecture Companion, ICSA-C 2018*, pp. 52–55, 2018.

[25] K. Borowa, A. Zalewski, and S. Kijas, "The Influence of Cognitive Biases on Architectural Technical Debt," in *International Conference on Software Architecture (ICSA)*, 2021.

[26] A. Zalewski, K. Borowa, and A. Ratkowski, "On cognitive biases in architecture decision making," in *Software Architecture: 11th European Conference, ECSA 2017, Canterbury, UK, September 11-15, 2017, Proceedings 11*. Springer, 2017, pp. 123–137.

[27] R. S. Nickerson, "Confirmation bias: A ubiquitous phenomenon in many guises," *Review of general psychology*, vol. 2, no. 2, pp. 175–220, 1998.

[28] T. Sharot, "The optimism bias," *Current biology*, vol. 21, no. 23, pp. R941–R945, 2011.

[29] A. Myers and C. Hansen, *Experimental psychology*. Thomson Wadsworth, 2006.

[30] K. A. Ericsson and H. A. Simon, *Protocol analysis: Verbal reports as data, Rev. ed.* Cambridge, MA, US: The MIT Press, 1993.

[31] K. Borowa, R. Rebouças de Almeida, and M. Wiese, "Additional material for: Debiasing architectural decision-making: An experiment with students and practitioners," Jan. 2025. [Online]. Available: https://doi.org/10.5281/zenodo.14740330

[32] K. Borowa, R. Dwornik, and A. Zalewski, "Is knowledge the key? an experiment on debiasing architectural decision-making-a pilot study," in *Product-Focused Software Process Improvement: 22nd International Conference, PROFES 2021, Turin, Italy, November 26, 2021, Proceedings 22*. Springer, 2021, pp. 207–214.

[33] M. Soliman, M. Wiese, Y. Li, M. Riebisch, and P. Avgeriou, "Exploring web search engines to find architectural knowledge," in *2021 IEEE 18th International Conference on Software Architecture (ICSA)*. IEEE, 2021, pp. 162–172.

[34] H. Cervantes and R. Kazman, *Designing software architectures: a practical approach*. Addison-Wesley Professional, 2016.

[35] J. Saldaña, *The Coding Manual for Qualitative Researchers (2nd edition)*, 2013, vol. 12, no. 2.

[36] F. Wilcoxon, "Individual comparisons by ranking methods," in *Breakthroughs in statistics: Methodology and distribution*. Springer, 1992, pp. 196–202.

[37] G. Calikli and A. Bener, "Empirical analyses of the factors affecting confirmation bias and the effects of confirmation bias on software developer/tester performance," in *Proceedings of the 6th international conference on predictive models in software engineering*, 2010, pp. 1–11.

[38] H. Muccini *et al.*, "Group decision-making in software architecture: A study on industrial practices," *Information and software technology*, vol. 101, pp. 51–63, 2018.

[39] L. J. Gonçales, K. Farias, and B. C. da Silva, "Measuring the cognitive load of software developers: An extended systematic mapping study," *Information and Software Technology*, vol. 136, p. 106563, 2021.

[40] K. L. Blankenship, D. T. Wegener, R. E. Petty, B. Detweiler-Bedell, and C. L. Macy, "Elaboration and consequences of anchored estimates: An attitudinal perspective on numerical anchoring," *Journal of Experimental Social Psychology*, vol. 44, no. 6, pp. 1465–1476, 2008.

[41] C. Wohlin, P. Runeson, M. Höst, M. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering: An Introduction (International Series in Software Engineering)*. Springer Science & Business Media, 2000.

[42] "ACM SIGSOFT Empirical Standards for Software Engineering," Dec. 2024. [Online]. Available: https://www2.sigsoft.org/EmpiricalStandards/docs/standards

Authorized licensed use limited to: Bibliothekssystem Universitaet Hamburg. Downloaded on December 11,2025 at 16:51:00 UTC from IEEE Xplore. Restrictions apply.

Bibliography

[ABH14] Md Abdullah Al Mamun, Christian Berger, and Jorgen Jörgen Hansson. "Explicating, Understanding, and Managing Technical Debt from Self-Driving Miniature Car Projects". In: *2014 Sixth International Workshop on Managing Technical Debt*. Institute of Electrical and Electronics Engineers Inc., Dec. 2014, pp. 11–18. ISBN: 9781479967919. DOI: 10.1109/MTD.2014.15.

[AECV25] Joshua Aldrich Edbert, Zadia Codabux, and Roberto Verdecchia. *Racing Against the Clock: Exploring the Impact of Scheduled Deadlines on Technical Debt*. ADS Bibcode: 2025arXiv250504027A. May 2025. DOI: 10.48550/arXiv.2505.04027. (Visited on 06/02/2025).

[Aes02] Aesop. *Aesop's Fables*. Ed. by Laura Gibbs. Oxford University Press, 2002. ISBN: 9780199540754.

[Ahm+21] Sabreen Ahmadjee et al. "Assessing Smart Contracts Security Technical Debts". In: *Proceedings - 2021 IEEE/ACM International Conference on Technical Debt, TechDebt 2021* March 2021 (Mar. 2021), pp. 6–15. DOI: 10.1109/TechDebt52882.2021.00010. eprint: 2103.09595.

[Alv+16] Nicolli S.R. Alves et al. "Identification and management of technical debt: A systematic mapping study". In: *Information and Software Technology* 70 (2016), pp. 100–121. ISSN: 09505849. DOI: 10.1016/j.infsof.2015.10.008.

[Amp+15] Apostolos Areti Ampatzoglou et al. "The financial aspect of managing technical debt: A systematic literature review". In: *Information and Software Technology*. Vol. 64. 2015, pp. 52–73. DOI: 10.1016/j.infsof.2015.04.001.

[ANDG23] Abdullah Aldaeej, Anh Nguyen-Duc, and Varun Gupta. "A Lean Approach of Managing Technical Debt in Agile Software Projects – A Proposal and Empirical Evaluation". en. In: *Agile Processes in Software Engineering and Extreme Programming*. Cham: Springer Nature Switzerland, 2023, pp. 67–76. ISBN: 978-3-031-33976-9. DOI: 10.1007/978-3-031-33976-9_5.

[Apa+20] Cecilia Apa et al. "The perception and management of technical debt in software startups". In: *Fundamentals of Software Startups: Essential Engineering and Business Aspects* (2020), pp. 61–78. DOI: 10.1007/978-3-030-35983-6_4.

[Apo+16] Areti Apostolos et al. "The Perception of Technical Debt in the Embedded Systems Domain: An Industrial Case Study". In: *2016 IEEE 8th International Workshop on Managing Technical Debt (MTD)*. 2016, pp. 9–16. DOI: 10.1109/MTD.2016.8.

[Avg+16a] Paris Avgeriou et al. "Managing Technical Debt in Software Engineering". In: *Dagstuhl Reports* 6.4 (2016), pp. 110–138. ISSN: 01635948. DOI: 10.4230/DagRep.6.4.110. URL: http://www.dagstuhl.de/16162.

[Avg+16b] Paris Avgeriou et al. "Reducing Friction in Software Development". In: *IEEE Software* 33.1 (Jan. 2016), pp. 66–73. ISSN: 1937-4194. DOI: 10.1109/MS.2016.13.

[Avg+21] Paris C. Avgeriou et al. "An Overview and Comparison of Technical Debt Measurement Tools". In: *IEEE Software* 38.3 (2021), pp. 61–71. ISSN: 19374194. DOI: 10.1109/MS.2020.3024958.

[Avg+23] Paris Avgeriou et al. "Technical Debt Management: The Road Ahead for Successful Software Delivery". In: *2023 IEEE/ACM International Conference on Software Engineering: Future of Software Engineering (ICSE-FoSE)*. May 2023, pp. 15–30. DOI: 10.1109/ICSE-FoSE59343.2023.00007. (Visited on 03/14/2024).

[Avg+25] Paris Avgeriou et al. *Manifesto from Dagstuhl Perspectives Workshop 24452 – Reframing Technical Debt*. arXiv:2505.13009 [cs]. May 2025. DOI: 10.48550/arXiv.2505.13009. (Visited on 06/02/2025).

[Bal+20] Maria Teresa Baldassarre et al. "On the diffuseness of technical debt items and accuracy of remediation time when using SonarQube". In: *Information and Software Technology* 128 (2020), p. 106377. ISSN: 0950-5849. DOI: 10.1016/j.infsof.2020.106377.

[BAW25] Klara Borowa, Rodrigo Rebouças de Almeida, and Marion Wiese. "Debiasing Architectural Decision-Making: An Experiment With Students and Practitioners". In: *2025 IEEE 22nd International Conference on Software Architecture (ICSA)*. ISSN: 2835-7043. Mar. 2025, pp. 210–220. DOI: 10.1109/ICSA65012.2025.00029. (Visited on 12/11/2025).

[BCK12] Len Bass, Paul Clements, and Rick Kazman. *Software Architecture in Practice*. Addison-Wesley, 2012.

[Bec+01] Kent Beck et al. *Manifesto for Agile Software Development*. 2001. URL: https://agilemanifesto.org/ (visited on 12/22/2025).

[Bec+18] Christoph Becker et al. "Trade-off decisions across time in technical debt management: A systematic literature review". In: *Proceedings - International Conference on Software Engineering* (2018), pp. 85–94. ISSN: 02705257. DOI: 10.1145/3194164.3194171.

[Beh+17] Woubshet Nema Behutiye et al. *Analyzing the concept of technical debt in the context of agile software development: A systematic literature review*. 2017. DOI: 10.1016/j.infsof.2016.10.004.

[Ber+21] Clara Berenguer et al. "Technical Debt is not Only about Code and We Need to be Aware about It". In: 2021, pp. 1–12. DOI: 10.1145/3493244.3493285.

[Bes] *Merriam-Webster – Best practices*. URL: https://www.merriam-webster.com/dictionary/best%20practice (visited on 01/08/2026).

[Bes+17] Terese Besker et al. "An Investigation of Technical Debt in Automatic Production Systems". In: *Proceedings of the XP2017 Scientific Workshops*. XP '17. New York, NY, USA: Association for Computing Machinery, 2017. ISBN: 9781450352642. DOI: 10.1145/3120459.3120466.

[Bes+20] Terese Besker et al. "The influence of Technical Debt on software developer morale". In: *Journal of Systems and Software* 167 (2020), p. 110586. ISSN: 0164-1212. DOI: 10.1016/j.jss.2020.110586.

[Bha+20] Manoj Bhat et al. "The evolution of architectural decision making as a key focus area of software architecture research". In: *Proc. IEEE 17th Int. Conf. on Software Architecture, ICSA 2020* c (2020). DOI: 10.1109/ICSA47634.2020.00015.

[BHG13] Olga Baysal, Reid Holmes, and Michael W. Godfrey. "Developer dashboards: The need for qualitative analytics". In: *IEEE Software* 30.4 (2013), pp. 46–52. ISSN: 07407459. DOI: 10.1109/MS.2013.66.

[BM19] Simon Baars and Sander Meester. "CodeArena: Inspecting and Improving Code Quality Metrics using Minecraft". In: *2019 IEEE/ACM International Conference on Technical Debt (TechDebt)*. 2019, pp. 68–70. ISBN: 978172 8133713. DOI: 10.1109/TechDebt.2019.00023.

[BMB16] Terese Besker, Antonio Martini, and Jan Bosch. "A Systematic Literature Review and a Unified Model of ATD". In: *Proceedings - 42nd Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2016*. Aug. 2016, pp. 189–197. ISBN: 9781509028191. DOI: 10.1109/SEAA.2016.42.

[BMB19] Terese Besker, Antonio Martini, and Jan Bosch. "Technical Debt Triage in Backlog Management". In: *2019 IEEE/ACM International Conference on Technical Debt (TechDebt)*. TechDebt '19. IEEE Press, 2019, pp. 13–22. ISBN: 9781728133713. DOI: 10.1109/TechDebt.2019.00010.

[BMB20] Terese Besker, Antonio Martini, and Jan Bosch. "Carrot and Stick Approaches When Managing Technical Debt". In: *Proceedings of the 3rd International Conference on Technical Debt*. TechDebt '20. New York, NY, USA: Association for Computing Machinery, 2020, pp. 21–30. ISBN: 9781450379601. DOI: 10.1145/3387906.3388619.

[BMB22] Terese Besker, Antonio Martini, and Jan Bosch. "The use of incentives to promote technical debt management". In: *Information and Software Technology* 142 (2022), p. 106740. ISSN: 0950-5849. DOI: 10.1016/j.infsof.2021.106740.

[Bor+21a] Klara Borowa et al. "A Marketplace for Industrial Case Studies in Software Engineering". In: *8th ACM Celebration of Women in Computing: womENcourage (womENcourage 2021)*. 2021, pp. 21–22.

[Bor+21b] Nichlas Bødker Borup et al. "Deliberative Technical Debt Management: An Action Research Study". en. In: *Software Business*. Ed. by Xiaofeng Wang et al. Lecture Notes in Business Information Processing. Cham: Springer International Publishing, 2021, pp. 50–65. ISBN: 978-3-030-91983-2. DOI: 10.1007/978-3-030-91983-2_5.

[Bor+22] Klara Borowa et al. "Debiasing Architectural Decision-Making: A Workshop-Based Training Approach". en. In: *Software Architecture*. Ed. by Ilias Gerostathopoulos et al. Cham: Springer International Publishing, 2022, pp. 159–166. ISBN: 978-3-031-16697-6. DOI: 10.1007/978-3-031-16697-6_11.

[BS16] Max H. Bazerman and Ovul Sezer. "Bounded awareness: Implications for ethical decision making". In: *Organizational Behavior and Human Decision Processes* 136 (2016), pp. 95–105. ISSN: 07495978. DOI: 10.1016/j.obhdp.2015.11.004.

[Bus11] Frank Buschmann. "To pay or not to pay technical debt". In: *IEEE Software* 28.6 (Nov. 2011), pp. 29–31. ISSN: 07407459. DOI: 10.1109/MS.2011.150.

[BWA09] Finn Olav Bjørnson, Alf Inge Wang, and Erik Arisholm. "Improving the effectiveness of root cause analysis in post mortem analysis: A controlled experiment". en. In: *Information and Software Technology* 51.1 (Jan. 2009). ISSN: 09505849. DOI: 10.1016/j.infsof.2008.02.003.

[BZK21] Klara Borowa, Andrzej Zalewski, and Szymon Kijas. "The Influence of Cognitive Biases on Architectural Technical Debt". In: *International Conference on Software Architecture (ICSA)*. 2021, pp. 115–125. ISBN: 9781728162607. DOI: 10.1109/ICSA51549.2021.00019.

[Cam+13] John L. Campbell et al. "Coding In-depth Semistructured Interviews: Problems of Unitization and Intercoder Reliability and Agreement". In: *Sociological Methods and Research* 42.3 (2013), pp. 294–320. ISSN: 00491241. DOI: 10.1177/0049124113500475.

[CB07] Dolly Chugh and Max H. Bazerman. "Bounded awareness: What you fail to see can hurt you". In: *Mind and Society* 6.1 (2007), pp. 1–18. ISSN: 15937879. DOI: 10.1007/s11299-006-0020-4.

[Cha+18] Sofia Charalampidou et al. "Integrating traceability within the IDE to prevent requirements documentation debt". In: *Proceedings - 44th Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2018* (Aug. 2018), pp. 421–428. DOI: 10.1109/SEAA.2018.00075.

[Coh88] Jacob Cohen. *Statistical Power Analysis fot the Behavioral Sciences*. New York, New York, USA: Lawrene Erlbaum Associates, 1988. ISBN: 0805802835.

[Cre+22] Yania Crespo et al. "The role of awareness and gamification on technical debt management". In: *Information and Software Technology* 150.April (2022), p. 106946. ISSN: 09505849. DOI: 10.1016/j.infsof.2022.106946.

[CSS12] Bill Curtis, Jay Sappidi, and Alexandra Szynkarski. "Estimating the size, cost, and types of Technical Debt". In: *2012 Third International Workshop on Managing Technical Debt (MTD)*. 2012, pp. 49–53. DOI: 10.1109/MTD.2012.6226000.

[Cun92] Ward Cunningham. "The WyCash portfolio management system". In: *Proceedings of the Conference on Object-Oriented Programming Systems, Languages, and Applications, OOPSLA*. 2. 1992, pp. 29–30. ISBN: 0897916107. DOI: 10.1145/157710.157715.

[Dan+25] Léo-Paul Dana et al. "The art of the elevator pitch: state-of-the-art and research agenda". en. In: *International Entrepreneurship and Management Journal* 21.1 (Mar. 2025), p. 73. ISSN: 1555-1938. DOI: 10.1007/s11365-025-01083-8. (Visited on 02/15/2026).

[Das+22] Dipta Das et al. "Technical Debt Resulting from Architectural Degradation and Code Smells: A Systematic Mapping Study". In: *SIGAPP Appl. Comput. Rev.* 21.4 (Jan. 2022). Place: New York, NY, USA Publisher: Association for Computing Machinery, pp. 20–36. ISSN: 1559-6915. DOI: 10.1145/3512753.3512755.

[Dig+20] Georgios Digkas et al. *On the temporality of introducing code technical debt*. Vol. 1266 CCIS. Springer International Publishing, 2020, pp. 68–82. ISBN: 9783030587925. DOI: 10.1007/978-3-030-58793-2_6.

[Din+12] Torgeir Dingsøyr et al. "A decade of agile methodologies: Towards explaining agile software development". en. In: *Journal of Systems and Software* 85.6 (June 2012), pp. 1213–1221. ISSN: 01641212. DOI: 10.1016/j.jss.2012.02.033. (Visited on 12/29/2025).

[DMR22] Thober Detofeno, Andreia Malucelli, and Sheila Reinehr. "PriorTD: A Method for Prioritization Technical Debt". In: *Proceedings of the XXXVI Brazilian Symposium on Software Engineering*. SBES '22. New York, NY, USA: Association for Computing Machinery, Oct. 2022, pp. 230–240. ISBN: 978-1-4503-9735-3. DOI: 10.1145/3555228.3555238.

[DOVH19] Quang Huan Dong, Felix Ocker, and Birgit Vogel-Heuser. "Technical Debt as indicator for weaknesses in engineering of automated production systems". In: *Production Engineering* 13.3-4 (2019), pp. 273–282. ISSN: 18637353. DOI: 10.1007/s11740-019-00897-0.

[DRA21] Thober Detofeno, Sheila Reinehr, and Malucelli Andreia. "Technical Debt Guild: When experience and engagement improve Technical Debt Management". In: *Proceedings of the XX Brazilian Symposium on Software Quality*. SBQS '21. New York, NY, USA: Association for Computing Machinery, Dec. 2021, pp. 1–10. ISBN: 978-1-4503-9553-3. DOI: 10.1145/3493244.3493271.

[DS87] Michael E Dawson and Anne M Schell. "Human autonomic and skeletal classical conditioning: The role of conscious cognitive factors". In: (1987).

[EKD21] Neil Ernst, Rick Kazman, and Julien Delange. In: *Technical Debt in Practice*. London, England: The MIT Press Cambridge, Massachusetts, 2021, p. 6. ISBN: 9780262542111.

[Eli+15] Ulf Eliasson et al. "Identifying and visualizing Architectural Debt and its efficiency interest in the automotive domain: A case study". In: *2015 IEEE 7th International Workshop on Managing Technical Debt (MTD)*. 2015, pp. 33–40. DOI: 10.1109/MTD.2015.7332622.

[End00] Mica R. Endsley. "Direct Measurement of Situation Awareness: Validity and Use of SAGAT". In: *Situation Awareness Analysis and Measurement*. Ed. by M. R. Endsley and D. J Garland. Mahwah, NJ: Routledge, 2000. ISBN: 978-0-8058-2134-5.

[End88] Mica R. Endsley. "Design and Evaluation for Situation Awareness Enhancement". In: *Proceedings of the Human Factors Society Annual Meeting* 32.2 (1988), pp. 97–101. DOI: https://doi.org/10.1177/154193128803200221.

[Ern+15] Neil A. Ernst et al. "Measure it? Manage it? Ignore it? Software practitioners and technical debt". In: *10th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, ESEC/FSE 2015 - Proceedings*. ESEC/FSE 2015 (2015), pp. 50–60. DOI: 10.1145/2786805.2786848.

[ES93] Karl Anders Ericsson and Herbert Alexander Simon. *Protocol analysis: Verbal reports as data, Rev. ed.* Cambridge, MA, US: The MIT Press, 1993, pp. liii, 443–liii, 443. ISBN: 0-262-05047-1 (Hardcover); 0-262-55023-7 (Paperback).

[Fon+17] Francesca Arcelli Fontana et al. "Technical Debt in Agile Development". In: *ACM SIGSOFT Software Engineering Notes* 42.3 (Sept. 2017), pp. 18–21. ISSN: 0163-5948. DOI: 10.1145/3127360.3127372.

[Fow09] Martin Fowler. *Technical Debt Quadrant*. 2009. URL: https://martinfowler.com/bliki/TechnicalDebtQuadrant.html (visited on 07/13/2022).

[FR15] Davide Falessi and Andreas Reichel. "Towards an open-source tool for measuring and visualizing the interest of technical debt". In: *2015 IEEE 7th International Workshop on Managing Technical Debt (MTD)*. Oct. 2015, pp. 1–8. DOI: 10.1109/MTD.2015.7332618. (Visited on 07/26/2024).

[Fre+20a] Sávio Freire et al. "Actions and impediments for technical debt prevention: Results from a global family of industrial surveys". In: *Proceedings of the ACM Symposium on Applied Computing*. SAC '20. New York, NY, USA: Association for Computing Machinery, 2020, pp. 1548–1555. ISBN: 9781450368667. DOI: 10.1145/3341105.3373912. URL: https://doi.org/10.1145/3341105.3373912.

[Fre+20b] Sávio Freire et al. "Surveying Software Practitioners on Technical Debt Payment Practices and Reasons for Not Paying off Debt Items". In: *Proceedings of the Evaluation and Assessment in Software Engineering*. EASE '20. New York, NY, USA: Association for Computing Machinery, 2020, pp. 210–219. ISBN: 978145037 7317. DOI: 10.1145/3383219.3383241.

[Fre+21] Sávio Savio Freire et al. "How Experience Impacts Practitioners' Perception of Causes and Effects of Technical Debt". In: *2021 IEEE/ACM 13th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*. 2021, pp. 21–30. ISBN: 9781665414098. DOI: 10.1109/CHASE52884.2021.00011.

[Fre+23] Sávio Freire et al. "Software practitioners' point of view on technical debt payment". In: *Journal of Systems and Software* 196 (2023), p. 111554. ISSN: 0164-1212. DOI: 10.1016/j.jss.2022.111554.

[FS+17] Carlos Fernández-Sánchez et al. "An Open Tool for Assisting in Technical Debt Management". In: *2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. Aug. 2017, pp. 400–403. DOI: 10.1109/SEAA.2017.60. (Visited on 07/26/2024).

[Fü21] Meik Führing. *Change Management Essentials for Managers and Smart Leaders*. Baden-Baden, Germany: Nomos Verlagsgesellschaft, 2021. ISBN: 978-3-95710-293-5. DOI: 10.5771/9783957102935.

[Gan14] Lakshmi Ganesh. "Board game as a tool to teach software engineering concept - Technical debt". In: *Proceedings - IEEE 6th International Conference on Technology for Education, T4E 2014*. IEEE, 2014, pp. 44–47. ISBN: 978-1-4799-6489-5. DOI: 10.1109/T4E.2014.28.

[Gar+06] D R Garrison et al. "Revisiting methodological issues in transcript analysis: Negotiated coding and reliability". In: *Internet and Higher Education* 9.1 (2006), pp. 1–8. ISSN: 10967516. DOI: 10.1016/j.iheduc.2005.11.001.

[Git] *GitLab | The most-comprehensive AI-powered DevSecOps platform*. 2025. URL: https://about.gitlab.com/ (visited on 10/22/2025).

[Goo] *Google Forms: Online Form Builder*. en. URL: https://workspace.google.com/products/forms/ (visited on 01/07/2026).

[GSS16] Yuepu Guo, Rodrigo Oliveira Spínola, and Carolyn Seaman. "Exploring the costs of technical debt management – a case study". In: *Empirical Software Engineering* 21.1 (2016), pp. 159–182. ISSN: 15737616. DOI: 10.1007/s10664-014-9351-7.

[Guo+11] Yuepu Guo et al. "Tracking technical debt - An exploratory case study". In: *2011 27th IEEE International Conference on Software Maintenance (ICSM)*. 2011, pp. 528–531. ISBN: 9781457706646. DOI: 10.1109/ICSM.2011.6080824.

[Guo+16] Yuepu Guo et al. "Costs and obstacles encountered in technical debt management – A case study". In: *Journal of Systems and Software* 120 (2016), pp. 156–169. ISSN: 01641212. DOI: 10.1016/j.jss.2016.07.008.

[Gup+16] Rajeev Kumar Gupta et al. "Pragmatic approach for managing technical debt in legacy software project". In: *Proceedings of the 9th India Software Engineering Conference*. Vol. 18-20-Febr. ISEC '16. New York, NY, USA: Association for Computing Machinery, 2016, pp. 170–176. ISBN: 9781450340182. DOI: 10.1145/2856636.2856655.

[HC12] Alan Hevner and Samir Chatterjee. *Design Research in Information Systems - Theory and Practice*. Vol. 22. 2012, pp. 63–86. ISBN: 9781441956521. DOI: 10.1007/978-1-4419-5653-8.

[He+16] Xiao He et al. "Technical Debt in MDE: A Case Study on GMF/EMF-Based Projects". In: *Proceedings of the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems*. MODELS '16. New York, NY, USA: Association for Computing Machinery, 2016, pp. 162–172. ISBN: 9781450343213. DOI: 10.1145/2976767.2976806.

[HL13] Johannes Holvitie and Ville Leppänen. "DebtFlag: Technical debt management with a development environment integrated tool". In: *2013 4th International Workshop on Managing Technical Debt (MTD)*. May 2013, pp. 20–27. DOI: 10.1109/MTD.2013.6608674.

[Hol+18] Johannes Holvitie et al. "Technical debt and agile software development practices and processes: An industry practitioner survey". In: *Information and Software Technology* 96 (2018), pp. 141–160. ISSN: 0950-5849. DOI: 10.1016/j.infsof.2017.11.015.

[ISO23] ISO/IEC. *ISO 25010-2023 | Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models*. 2023.

[JG23] Ciera Jaspan and Collin Green. "Defining, Measuring, and Managing Technical Debt". In: *IEEE Software* 40.3 (May 2023). Conference Name: IEEE Software, pp. 15–19. ISSN: 1937-4194. DOI: 10.1109/MS.2023.3242137. (Visited on 09/23/2024).

[Jir] *Jira | Issue & Project Tracking Software | Atlassian*. 2025. URL: https://www.atlassian.com/software/jira (visited on 10/22/2025).

[JT22] Helvio Jeronimo Junior and Guilherme Horta Travassos. "Consolidating a common perspective on Technical Debt and its Management through a Tertiary Study". In: *Information and Software Technology* 149.October 2020 (2022), p. 106964. ISSN: 09505849. DOI: 10.1016/j.infsof.2022.106964.

[Kat+22] Andrej Katin et al. "On the Technical Debt Prioritization and Cost Estimation with SonarQube Tool". en. In: *Proceedings on 18th International Conference on Industrial Systems – IS'20*. Cham: Springer International Publishing, 2022, pp. 302–309. ISBN: 978-3-030-97947-8. DOI: 10.1007/978-3-030-97947-8_40.

[KKT91] Daniel Kahneman, Jack L. Knetsch, and Richard H. Thaler. "Anomalies: The endowment effect, loss aversion, and status quo bias". In: *Journal of Economic Perspectiv* 5.1 (1991), pp. 193–206. DOI: 10.1017/CBO9780511803475.009.

[Klu+25] Jil Kluender et al. "How to Mitigate Technical Debt in Startups: Extending the Entrepreneurial Software Engineering Model". In: Gesellschaft für Informatik e.V., 2025. URL: https://dl.gi.de/server/api/core/bitstreams/155fdd95-f996-4b17-b479-8080b177b915/content (visited on 12/11/2025).

[KM20] Pawel Klimczyk and Lech Madeyski. "Technical debt aware estimations in software engineering: A systematic mapping study". In: *E-Informatica Software Engineering Journal* 14.1 (2020), pp. 61–76. ISSN: 20844840. DOI: 10.37190/e-Inf200102.

[KNO12] Philippe Kruchten, Robert L. Nord, and Ipek Ozkaya. "Technical debt: From metaphor to theory and practice". In: *IEEE Software* 29.6 (Nov. 2012), pp. 18–21. ISSN: 07407459. DOI: 10.1109/MS.2012.167.

[KNO19] Philippe Kruchten, Robert Nord, and Ipek Ozkaya. *Managing Technical Debt: Reducing Friction in Software Development*. Software Engineering Institute, Carnegie Mellon University, 2019.

[Kpi] *Cambridge Dictionary – Key Performance Indicator*. URL: https://dictionary.cambridge.org/dictionary/english/key-performance-indicator (visited on 01/08/2026).

[KR11] Michael Kaiser and Guy Royse. "Selling the Investment to Pay Down Technical Debt: The Code Christmas Tree". In: *2011 Agile Conference*. Aug. 2011, pp. 175–180. DOI: 10.1109/AGILE.2011.50.

[LAL15] Zengyang Li, Paris Avgeriou, and Peng Liang. "A systematic mapping study on technical debt and its management". In: *Journal of Systems and Software* 101 (Mar. 2015), pp. 193–220. ISSN: 01641212. DOI: 10.1016/j.jss.2014.12.027.

[LCP20] Bjoern Arild Lunde and Ricardo Colomo-Palacios. "Continuous practices and technical debt: a systematic literature review". In: *2020 20th International Conference on Computational Science and Its Applications (ICCSA)*. 2020, pp. 40–44. DOI: 10.1109/ICCSA50381.2020.00018.

[Lei+24] Gilberto de Sousa Leite et al. "Technical Debt Management in Agile Software Development: A Systematic Mapping Study". In: *Proceedings of the XXIII Brazilian Symposium on Software Quality*. SBQS '24. New York, NY, USA: Association for Computing Machinery, Dec. 2024, pp. 309–320. ISBN: 979-8-4007-1777-2. DOI: 10.1145/3701625.3701669. (Visited on 01/07/2025).

[Len+19] Valentina Lenarduzzi et al. "An Empirical Study on Technical Debt in a Finnish SME". In: *2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. Vol. 2019-Septe. 2019, pp. 1–6. ISBN: 9781728129686. DOI: 10.1109/ESEM.2019.8870169.

[Len+20] Valentina Lenarduzzi et al. "A systematic literature review on Technical Debt prioritization: Strategies, processes, factors, and tools (journal version)". In: *The Journal ofSystems & Software journal* 171.171 (2020). DOI: 10.1016/j.jss.2020.110827.

[Len+21] Valentina Lenarduzzi et al. "A systematic literature review on Technical Debt prioritization: Strategies, processes, factors, and tools". In: *Journal of Systems and Software* 171 (Jan. 2021), p. 110827. ISSN: 01641212. DOI: 10.1016/j.jss.2020.110827.

[LI12] Jean-Louis Letouzey and Michel Ilkiewicz. "Managing Technical Debt with the SQALE Method". In: *IEEE Software* 29.6 (Nov. 2012), pp. 44–51. ISSN: 15227383. DOI: 10.1109/MS.2012.129.

[Lim] *LimeSurvey — Free Online Survey Tool*. en-us. URL: https://www.limesurvey.org/ (visited on 01/07/2026).

[Liu+20] Jiakun Liu et al. "Is Using Deep Learning Frameworks Free? Characterizing Technical Debt in Deep Learning Frameworks". In: *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering in Society*. New York, NY, USA: Association for Computing Machinery, 2020, pp. 1–10. ISBN: 9781450371244. DOI: 10.1145/3377815.3381377.

[LSA22] Yikun Li, Mohamed Soliman, and Paris Avgeriou. "Identifying self-admitted technical debt in issue tracking systems using machine learning". en. In: *Empirical Software Engineering* 27.6 (July 2022), p. 131. ISSN: 1573-7616. DOI: 10.1007/s10664-022-10128-3. (Visited on 04/04/2023).

[Max] *MAXQDA | All-In-One Tool for Qualitative Data Analysis & Mixed Methods - MAXQDA*. URL: https://www.maxqda.com/ (visited on 10/06/2022).

[Max18] Dominik Maximini. *The Scrum Culture*. en. Management for Professionals. Cham: Springer International Publishing, 2018. ISBN: 978-3-319-73841-3 978-3-319-73842-0. DOI: 10.1007/978-3-319-73842-0. (Visited on 07/26/2024).

[MB15] Antonio Martini and Jan Bosch. "The Danger of Architectural Technical Debt: Contagious Debt and Vicious Circles". In: *Proceedings - 12th Working IEEE/IFIP Conference on Software Architecture, WICSA 2015*. 2015, pp. 1–10. ISBN: 978147991 9222. DOI: 10.1109/WICSA.2015.31.

[MB16] Antonio Martini and Jan Bosch. "An Empirically Developed Method to Aid Decisions on Architectural Technical Debt Refactoring: AnaConDebt". In: *2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C)*. 2016, pp. 31–40. DOI: 10.1145/2889160.2889224.

[MB17] Antonio Martini and Jan Bosch. "Revealing social debt with the CAFFEA framework: An antidote to architectural debt". In: *Proceedings - 2017 IEEE International Conference on Software Architecture Workshops, ICSAW 2017: Side Track Proceedings*. IEEE, Apr. 2017, pp. 179–181. ISBN: 9781509047932. DOI: 10.1109/ICSAW.2017.42. URL: http://ieeexplore.ieee.org/document/7958479/.

[MBB18] Antonio Martini, Terese Besker, and Jan Bosch. "Technical Debt tracking: Current state of practice: A survey and multiple case study in 15 large organizations". In: *Science of Computer Programming* 163 (2018), pp. 42–61. ISSN: 01676423. DOI: 10.1016/j.scico.2018.03.007.

[MBC14] Antonio Martini, Jan Bosch, and Michel Chaudron. "Architecture technical debt: Understanding causes and a qualitative model". In: *Proceedings - 40th Euromicro Conference Series on Software Engineering and Advanced Applications, SEAA 2014*. 2014, pp. 85–92. ISBN: 978-1-4799-5794-1. DOI: 10.1109/SEAA.2014.65.

[McC08] Steve McConnell. "Managing technical debt". In: *Construx Inc.* (2008). URL: http://www.construx.com/uploadedfiles/resources/whitepapers/ManagingTechnicalDebt.pdf.

[Men+19] Thiago S. Mendes et al. "VisminerTD: a tool for automatic identification and interactive monitoring of the evolution of technical debt items". In: *Journal of the Brazilian Computer Society* 25.1 (2019). ISSN: 16784804. DOI: 10.1186/s13173-018-0083-1.

[Mer] *Merriam Webster Dictionary – Communication*. en. URL: https://www.merriam-webster.com/dictionary/communication (visited on 01/07/2026).

[Mey14] Bertrand Meyer. *Agile!: The Good, the Hype and the Ugly*. en. Cham: Springer International Publishing, 2014. ISBN: 978-3-319-05154-3 978-3-319-05155-0. DOI: 10.1007/978-3-319-05155-0. (Visited on 07/26/2024).

[MH12] Viljan Mahnič and Tomaž Hovelja. "On using planning poker for estimating user stories". In: *Journal of Systems and Software*. Selected papers from the 2011 Joint Working IEEE/IFIP Conference on Software Architecture (WICSA 2011) 85.9 (Sept. 2012), pp. 2086–2095. ISSN: 0164-1212. DOI: 10.1016/j.jss.2012.04.005. (Visited on 02/15/2026).

[Mic25a] Microsoft. *Azure DevOps Services | Microsoft Azure*. en-US. 2025. URL: https://azure.microsoft.com/en-us/products/devops (visited on 10/22/2025).

[Mic25b] Microsoft. *Free Online Spreadsheet Software: Excel | Microsoft 365*. en-US. 2025. URL: https://www.microsoft.com/en-us/microsoft-365/excel (visited on 03/04/2025).

[Mic25c] Microsoft. *Power BI - Data Visualization | Microsoft Power Platform*. en-US. 2025. URL: https://www.microsoft.com/en-us/power-platform/products/power-bi (visited on 03/04/2025).

[MO20] Somayeh Malakuti and Sergey Ostroumov. "The quest for introducing technical debt management in a large-scale industrial company". In: *Software Architecture. ECSA 2020. Lecture Notes in Computer Science*. Vol. 12292. Springer International Publishing, 2020, pp. 296–311. ISBN: 9783030589226. DOI: 10.1007/978-3-030-58923-3_20. URL: http://dx.doi.org/10.1007/978-3-030-58923-3{_}20.

[Moe13] Nils Brede Moe. "Key Challenges of Improving Agile Teamwork". en. In: vol. 149. Lecture Notes in Business Information Processing. Springer Berlin Heidelberg, 2013. DOI: 10.1007/978-3-642-38314-4_6.

[Mor+12] J. David Morgenthaler et al. "Searching for build debt: Experiences managing technical debt at Google". In: *2012 3rd International Workshop on Managing Technical Debt, MTD 2012 - Proceedings* (2012), pp. 1–6. DOI: 10.1109/MTD.2012.6225994.

[Mor+21] Rui S Moreira et al. "Trends and Applications in Information Systems and Technologies". In: 1367 (2021), pp. 483–493. DOI: 10.1007/978-3-030-72660-7.

[MSB21] Haris Mumtaz, Paramvir Singh, and Kelly Blincoe. "A systematic mapping study on architectural smells detection". en. In: *Journal of Systems and Software* 173 (Mar. 2021). ISSN: 01641212. DOI: 10.1016/j.jss.2020.110885.

[MTR20] Vladimir Mandić, Nebojša Taušan, and Robert Ramač. "The Prevalence of the Technical Debt Concept in Serbian IT Industry: Results of a National-Wide Survey". In: *Proceedings of the 3rd International Conference on Technical Debt*. TechDebt '20. New York, NY, USA: Association for Computing Machinery, 2020, pp. 77–86. ISBN: 9781450379601. DOI: 10.1145/3387906.3388622.

[MW21] Whitney Rachel Morgan and Erik Scott Wright. "Ten simple rules for hitting a home run with your elevator pitch". In: *PLoS Computational Biology* 17.3 (Mar. 2021), e1008756. ISSN: 1553-734X. DOI: 10.1371/journal.pcbi.1008756. (Visited on 01/12/2026).

[MW47] H. B. Mann and D. R. Whitney. "On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other". In: *The Annals of Mathematical Statistics* 18.1 (1947), pp. 50–60. ISSN: 0003-4851. DOI: 10.1214/aoms/1177730491.

[NVK11] Ariadi Nugroho, Joost Visser, and Tobias Kuipers. "An Empirical Model of Technical Debt and Interest". In: *Proceedings of the 2nd Workshop on Managing Technical Debt*. Series Title: MTD '11 ISSN: 02705257. New York, NY, USA: Association for Computing Machinery, 2011, pp. 1–8. ISBN: 978-1-4503-0586-0. DOI: 10.1145/1985362.1985364.

[Ock+19] Felix Ocker et al. "Increasing Awareness for Potential Technical Debt in the Engineering of Production Systems". In: *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*. Vol. 1. Institute of Electrical and Electronics Engineers Inc., July 2019, pp. 478–484. ISBN: 978-1-72812-927-3. DOI: 10.1109/INDIN41052.2019.8972268.

[OGS15] Frederico Oliveira, Alfredo Goldman, and Viviane Santos. "Managing Technical Debt in Software Projects Using Scrum: An Action Research". In: *Proceedings - 2015 Agile Conference, Agile 2015*. 2015, pp. 50–59. ISBN: 978-1-4673-7153-7. DOI: 10.1109/Agile.2015.7.

[Pea01] Karl Pearson. "I. Mathematical contributions to the theory of evolution. —VII. On the correlation of characters not quantitatively measurable". In: *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character* 195.262-273 (1901), pp. 1–47. ISSN: 0264-3952. DOI: 10.1098/rsta.1900.0022.

[Pef+07] Ken Peffers et al. "A Design Science Research Methodology for Information Systems Research". In: *Journal of Management Information Systems* 24.3 (2007), pp. 45–77. ISSN: 0742-1222. DOI: 10.2753/MIS0742-1222240302.

[Pér+21] Boris Pérez et al. "Technical debt payment and prevention through the lenses of software architects". In: *Information and Software Technology* 140.December 2020 (2021), p. 106692. ISSN: 0950-5849. DOI: 10.1016/j.infsof.2021.106692.

[Per+23] Judith Perera et al. *Quantifying Technical Debt: A Systematic Mapping Study and a Conceptual Model*. arXiv:2303.06535 [cs]. Mar. 2023. URL: http://arxiv.org/abs/2303.06535 (visited on 03/16/2023).

[Per+24] Judith Perera et al. "A Systematic Mapping Study Exploring Quantification Approaches to Code, Design, and Architecture Technical Debt". In: *ACM Trans. Softw. Eng. Methodol.* (July 2024). Just Accepted. ISSN: 1049-331X. DOI: 10.1145/3675393. (Visited on 08/05/2024).

[PWL18] Alexander Poth, Mark Werner, and Xinyan Lei. "How to Deliver Faster with CI/CD Integrated Testing Services?" en. In: *Systems, Software and Services Process Improvement*. Ed. by Xabier Larrucea et al. Cham: Springer International Publishing, 2018, pp. 401–409. ISBN: 978-3-319-97925-0. DOI: 10.1007/978-3-319-97925-0_33.

[Ram+21] Robert Ramač et al. "Prevalence, Common Causes and Effects of Technical Debt: Results from a Family of Surveys with the IT Industry". In: *Journal of Systems and Software* (2021). ISSN: 01641212. DOI: 10.1016/j.jss.2021.111114. arXiv: 2109.13771. URL: http://arxiv.org/abs/2109.13771.

[RCM22] Barbara Russo, Matteo Camilli, and Moritz Mock. "WeakSATD: Detecting Weak Self-admitted Technical Debt". In: *2022 IEEE/ACM 19th International Conference on Mining Software Repositories (MSR)*. 2022, pp. 448–453. DOI: 10.1145/3524842.3528469.

[Reb+18] Rodrigo Reboucas De Almeida et al. "Aligning technical debt prioritization with business objectives: A multiple-case study". In: *Proceedings - 2018 IEEE International Conference on Software Maintenance and Evolution, ICSME 2018* (2018), pp. 655–664. ISSN: 2576-3148. DOI: 10.1109/ICSME.2018.00075. eprint: 1807.05582.

[Reb19] Rodrigo Reboucas De Almeida. "Business-Driven Technical Debt Prioritization". In: *Proceedings - 2019 IEEE International Conference on Software Maintenance and Evolution, ICSME 2019* (2019), pp. 605–609. ISSN: 2576-3148. DOI: 10.1109/ICSME.2019.00096. eprint: 1908.01347.

[Reb+21] Rodrigo Rebouças de Almeida et al. "Business-Driven Technical Debt Prioritization - An Industrial Case Study". In: *2021 IEEE/ACM International Conference on Technical Debt (TechDebt)*. 2021, pp. 74–83. ISBN: 9781665414050. DOI: 10.1109/TechDebt52882.2021.00017. eprint: 2010.09711.

[Rib+16] Leilane Ferreira Ribeiro et al. "Decision criteria for the payment of technical debt in software projects: A systematic mapping study". In: *ICEIS 2016 - Proceedings of the 18th International Conference on Enterprise Information Systems*. Vol. 1. 2016, pp. 572–579. ISBN: 9789897581878. DOI: 10.5220/0005914605720579.

[Ric15] Mark Richards. *Software Architecture Patterns*. en. O'Reilly Media, 2015. ISBN: 978-1-4919-2424-2.

[Rio+18a] Nicolli Rios et al. "A study of factors that lead development teams to incur technical debt in software projects". In: *Proceedings - 44th Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2018*. IEEE, Aug. 2018, pp. 429–436. ISBN: 9781538673829. DOI: 10.1109/SEAA.2018.00076.

[Rio+18b] Nicolli Rios et al. "The Most Common Causes and Effects of Technical Debt: First Results from a Global Family of Industrial Surveys". In: *International Symposium on Empirical Software Engineering and Measurement (ESEM'18)*. ESEM '18. Oulu, Finnland: Association for Computing Machinery, 2018. ISBN: 9781450358231. DOI: 10.1145/3239235.3268917. URL: https://doi.org/10.1145/3239235.3268917.

[Rio+19] Nicolli Rios et al. "Supporting Analysis of Technical Debt Causes and Effects with Cross-Company Probabilistic Cause-Effect Diagrams". In: *2019 IEEE/ACM International Conference on Technical Debt (TechDebt)*. 2019, pp. 3–12. ISBN: 9781728133713. DOI: 10.1109/TechDebt.2019.00009.

[Rio+20] Nicolli Rios et al. "The practitioners' point of view on the concept of technical debt and its causes and consequences: a design for a global family of industrial surveys and its first results from Brazil". In: *Empirical Software Engineering* 25.5 (2020), pp. 3216–3287. ISSN: 15737616. DOI: 10.1007/s10664-020-09832-9.

[Rio+21] Nicolli Rios et al. "On the Relationship Between Technical Debt Management and Process Models". In: *IEEE Software* 7459.c (2021), p. 0. ISSN: 19374194. DOI: 10.1109/MS.2021.3058652.

[RK19] Narayan Ramasubbu and Chris F. Kemerer. "Integrating technical debt management and software quality management processes: A framework and field test". In: *Test Engineering and Management* 45.7-8 (May 2019), p. 883. ISSN: 019341 20. DOI: 10.1145/3180155.3182529.

[RMS18] Nicolli Rios, Manoel Gomes de Mendonça Neto, and Rodrigo Oliveira Spínola. *A tertiary study on technical debt: Types, management strategies, research trends, and base information for practitioners*. 2018. DOI: 10.1016/j.infsof.2018.05.010. URL: https://doi.org/10.1016/j.infsof.2018.05.010.

[RR22] Paula Rachow and Matthias Riebisch. "An architecture smell knowledge base for managing architecture technical debt". In: *Proceedings of the International Conference on Technical Debt*. TechDebt '22. New York, NY, USA: Association for Computing Machinery, Aug. 2022, pp. 1–10. ISBN: 978-1-4503-9304-1. DOI: 10.1145/3524843.3528092. (Visited on 07/22/2024).

[Run+12] Per Runeson et al. *Case Study Research in Software Engineering*. John Wiley and Sons Ltd, 2012, p. 241. ISBN: 9781118104354. DOI: 10.1007/978-3-030-49392-9.

[RWH23] Paula Rachow, Marion Wiese, and André van Hoorn. "Taxonomy of Architecture Maintainability Smells". In: *2023 30th Asia-Pacific Software Engineering Conference (APSEC)*. ISSN: 2640-0715. Dec. 2023, pp. 31–40. DOI: 10.1109/APSEC60848.2023.00013. (Visited on 12/11/2025).

[RZN17] Junior Cesar Rocha, Vanius Zapalowski, and Ingrid Nunes. "Understanding Technical Debt at the Code Level from the Perspective of Software Developers". In: *Proceedings of the 31st Brazilian Symposium on Software Engineering*. SBES'17 i. New York, NY, USA: Association for Computing Machinery, 2017, pp. 64–73. ISBN: 9781450353267. DOI: 10.1145/3131151.3131164. URL: https://doi.org/10.1145/3131151.3131164.

[Saa+19] Nyyti Saarimaki et al. "On the Accuracy of SonarQube Technical Debt Remediation Time". In: *2019 45th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. Aug. 2019, pp. 317–324. DOI: 10.1109/SEAA.2019.00055.

[Sal13] Johnny Saldaña. *The Coding Manual for Qualitative Researchers (2nd edition)*. 2nd Editio. London, England: SAGE publications, 2013. ISBN: 9781446247365. DOI: 10.1108/qrom-08-2016-1408.

[San+13] Paulo Sérgio Medeiros dos Santos et al. "Visualizing and managing technical debt in agile development: An experience report". In: *Lecture Notes in Business Information Processing* 149 (2013), pp. 121–134. ISSN: 18651348. DOI: 10.1007/978-3-642-38314-4_9.

[Sar+21] Diego Saraiva et al. "Technical Debt Tools: A Systematic Mapping Study". In: April. 2021, pp. 88–98. ISBN: 0000000302. DOI: 10.5220/0010459100880098.

[Sch13a] Klaus Schmid. "A formal approach to technical debt decision making". In: *QoSA 2013 - Proceedings of the 9th International ACM Sigsoft Conference on the Quality of Software Architectures*. QoSA '13 (2013), pp. 153–162. DOI: 10.1145/2465478.2465492. URL: https://doi.org/10.1145/2465478.2465492.

[Sch13b] Klaus Schmid. "On the Limits of the Technical Debt Metaphor: Some Guidance on Going Beyond". In: *Fourth Workshop on Managing Technical Debt, Workshop at the International Conference on Software Engineering*. IEEE, 2013, pp. 63–66. ISBN: 9781467364430. DOI: 10.1109/MTD.2013.6608681.

[SCW20] Marek G. Stochel, Piotr Cholda, and Mariusz R. Wawrowski. "On Coherence in Technical Debt Research : Awareness of the Risks Stemming from the Metaphorical Origin and Relevant Remediation Strategies". In: *Proceedings - 46th Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2020*. 2020, pp. 367–375. ISBN: 978-1-7281-9532-2. DOI: 10.1109/SEAA51224.2020.00067.

[SE20] Phani Sekhar Emmanni. "Implementing CI / CD Pipelines for Enhanced Efficiency in IT Projects". en. In: *International Journal of Science and Research (IJSR)* 9.9 (Sept. 2020), pp. 1616–1619. ISSN: 23197064. DOI: 10.21275/SR24402001528. (Visited on 12/22/2025).

[Sea+12] Carolyn Seaman et al. "Using technical debt data in decision making: Potential decision approaches". In: *2012 Third International Workshop on Managing Technical Debt (MTD)*. June 2012, pp. 45–48. DOI: 10.1109/MTD.2012.6225999.

[Ser17] Olivier Serrat. "The Five Whys Technique". en. In: *Knowledge Solutions: Tools, Methods, and Approaches to Drive Organizational Performance*. Ed. by Olivier Serrat. Singapore: Springer, 2017, pp. 307–310. ISBN: 978-981-10-0983-9. DOI: 10.1007/978-981-10-0983-9_32. (Visited on 07/25/2025).

[SF15] Klaas Jan Stol and Brian Fitzgerald. "Theory-oriented software engineering". In: *Science of Computer Programming* 101 (2015), pp. 79–98. ISSN: 01676423. DOI: 10.1016/j.scico.2014.11.010.

[SF18] Klaas Jan Stol and Brian Fitzgerald. "The ABC of software engineering research". In: *ACM Transactions on Software Engineering and Methodology* 27.3 (2018). ISSN: 15577392. DOI: 10.1145/3241743.

[SG11] Carolyn Seaman and Yuepu Guo. "Measuring and Monitoring Technical Debt". In: *Advances in Computers*. Ed. by Marvin V. Zelkowitz. Vol. 82. Elsevier, Jan. 2011, pp. 25–46. DOI: 10.1016/B978-0-12-385512-1.00002-5. (Visited on 07/19/2024).

[SIG25] ACM SIGSOFT. *ACM Special Interest Group on Software Engineering's Empirical Standards for Action Research*. en. Mar. 2025. URL: https://www2.sigsoft.org/EmpiricalStandards/docs/standards?standard=ActionResearch# (visited on 03/04/2025).

[Sne14] Harry M. Sneed. "Dealing with Technical Debt in agile development projects". In: *Lecture Notes in Business Information Processing* 166 LNBIP (2014), pp. 48–62. ISSN: 18651348. DOI: 10.1007/978-3-319-03602-1_4.

[Sol+21a] Mohamed Soliman et al. "Architectural design decisions that incur technical debt — An industrial case study". In: *Information and Software Technology* 139 (2021), p. 106669. ISSN: 0950-5849. DOI: 10.1016/j.infsof.2021.106669.

[Sol+21b] Mohamed Soliman et al. "Exploring Web Search Engines to Find Architectural Knowledge". In: *International Conference on Software Architecture 2021 (ICSA2021)*. 2021, pp. 162–172. ISBN: 978-1-7281-6260-7. DOI: 10.1109/ICSA51549.2021.00023.

[Son25] Sonar. *Better Code & Better Software | Ultimate Security and Quality | Sonar*. 2025. URL: https://www.sonarsource.com/ (visited on 03/04/2025).

[Sps] *IBM SPSS Statistics*. URL: https://www.ibm.com/products/spss-statistics (visited on 01/08/2026).

[Squ] *SQUALE – Software QUALity Enhancement*. en. URL: http://www.squale.org/ (visited on 01/07/2026).

[SRB23] Armando Sousa, Lincoln Rocha, and Ricardo Britto. "Architectural Technical Debt - A Systematic Mapping Study". In: *Proceedings of the XXXVII Brazilian Symposium on Software Engineering*. SBES '23. New York, NY, USA: Association for Computing Machinery, Sept. 2023, pp. 196–205. ISBN: 979-8-4007-0787-2. DOI: 10.1145/3613372.3613399. (Visited on 10/02/2023).

[SSS15] Tushar Sharma, Girish Suryanarayana, and Ganesh Samarthyam. "Challenges to and Solutions for Refactoring Adoption: An Industrial Perspective". In: *IEEE Software* 32.6 (Nov. 2015), pp. 44–51. ISSN: 1937-4194. DOI: 10.1109/MS.2015.105.

[Sta20] Miroslaw Staron. *Action Research in Software Engineering: Theory and Applications*. en. Cham: Springer International Publishing, 2020. ISBN: 978-3-030-32609-8 978-3-030-32610-4. DOI: 10.1007/978-3-030-32610-4. (Visited on 03/14/2023).

[STAF21] Irene Sala, Antonela Tommasel, and Francesca Arcelli Fontana. "DebtHunter: A Machine Learning-Based Approach for Detecting Self-Admitted Technical Debt". In: *Evaluation and Assessment in Software Engineering*. New York, NY, USA: Association for Computing Machinery, 2021, pp. 278–283. ISBN: 978-1-4503-9053-8. DOI: 10.1145/3463274.3464455.

[SUW22] Lakmal Silva, Michael Unterkalmsteiner, and Krzysztof Wnuk. "MultiDimEr: a multi-dimensional bug analyzEr". In: *Proceedings of the International Conference on Technical Debt*. TechDebt '22. New York, NY, USA: Association for Computing Machinery, Aug. 2022, pp. 66–70. ISBN: 978-1-4503-9304-1. DOI: 10.1145/3524843.3528099. (Visited on 07/26/2024).

[SWA17] Ben Stopford, Ken Wallace, and John Allspaw. "Technical Debt: Challenges and Perspectives". In: *IEEE Software* 34.4 (2017), pp. 79–81. ISSN: 1937-4194. DOI: 10.1109/MS.2017.99.

[Tab] *TABLETOPIA – Play 255*. en. URL: https://tabletopia.com/ (visited on 01/07/2026).

[Tan+17] Antony Tang et al. "Human Aspects in Software Architecture Decision Making - A Literature Review". In: *Int. Conf. on Software Architecture (ICSA2017)*. 2017. DOI: 10.1109/ICSA.2017.15.

[TAV13] Edith Tom, Aybüke Aurum, and Richard Vidgen. "An exploration of technical debt". In: *Journal of Systems and Software* 86.6 (2013), pp. 1498–1516. ISSN: 01641212. DOI: 10.1016/j.jss.2012.12.052.

[Teca] *Digital TechDebt Game on Tabletopia*. URL: https://tabletopia.com/workshop/games/techdebts/1-7players/test.

[Tecb] *Digital TechDebt Game on Tabletopia*. URL: https://tabletopia.com/workshop/games/techdebts/1-6players-1/test.

[Tecc] *Tabletop TechDebt Game on GitHub*. URL: https://github.com/TechDebtGame/TechDebtGame.

[Ton+17] Graziela Simone Tonin et al. "Effects of Technical Debt Awareness: A Classroom Study". In: *18th International Conference, XP 2017*. Ed. by Hubert Baumeister, Horst Lichter, and Matthias Riebisch. Vol. 283. 2017, pp. 84–100. ISBN: 978-3-319-57633-6. DOI: 10.1007/978-3-319-57633-6_6.

[TS10] Christoph Treude and Margaret Anne Storey. "Awareness 2.0: Staying aware of projects, developers and tasks using dashboards and feeds". In: *Proceedings - International Conference on Software Engineering* 1 (2010), pp. 365–374. ISSN: 02705257. DOI: 10.1145/1806799.1806854.

[Tso+20] Dimitrios Tsoukalas et al. "Technical debt forecasting: An empirical study on open-source repositories". In: *Journal of Systems and Software* 170 (2020), p. 110777. ISSN: 0164-1212. DOI: 10.1016/j.jss.2020.110777.

[Ver+21] Roberto Verdecchia et al. "Building and evaluating a theory of architectural technical debt in software-intensive systems". In: *Journal of Systems and Software* 176 (2021), p. 110925. ISSN: 01641212. DOI: 10.1016/j.jss.2021.110925.

[VHB21] Birgit Vogel-Heuser and Fandi Bi. "Interdisciplinary effects of technical debt in companies with mechatronic products — a qualitative study". In: *Journal of Systems and Software* 171 (2021), p. 110809. ISSN: 0164-1212. DOI: 10.1016/j.jss.2020.110809.

[VHN17] Birgit Vogel-Heuser and Eva-Maria Maria Neumann. "Adapting the concept of technical debt to software of automated Production Systems focusing on fault handling, mode of operation and safety aspects". In: *IFAC-PapersOnLine* 50.1 (2017), pp. 5887–5894. ISSN: 24058963. DOI: 10.1016/j.ifacol.2017.08.1308.

[WB23] Marion Wiese and Klara Borowa. "IT managers' perspective on Technical Debt Management". en. In: *Journal of Systems and Software* 202 (Apr. 2023), p. 111700. ISSN: 01641212. DOI: 10.1016/j.jss.2023.111700. (Visited on 04/12/2023).

[WBH23] Marion Wiese, Axel-Frederik Brand, and André van Hoorn. "Learning from Each Other: How Are Architectural Mistakes Communicated in Industry?" en. In: *Software Architecture*. Cham: Springer Nature Switzerland, 2023, pp. 319–336. ISBN: 978-3-031-42592-9. DOI: 10.1007/978-3-031-42592-9_22.

[WG16] Rainer Weinreich and Iris Groher. "Software architecture knowledge management approaches and their support for knowledge management activities: A systematic literature review". en. In: *Information and Software Technology* 80 (Dec. 2016). ISSN: 0950-5849. DOI: 10.1016/j.infsof.2016.09.007. (Visited on 05/08/2023).

[Wie20] Marion Wiese. "Introducing a Framework for Managing Technical Debt Developed by Practitioners". In: *22.Workshop Software-Reengineering & -Evolution2020, WSRE 2020*. Issue: 22. 2020, pp. 37–38.

[Wie+22] Marion Wiese et al. "Preventing technical debt with the TAP framework for Technical Debt Aware Management". In: *Information and Software Technology* (2022), p. 106926. ISSN: 0950-5849. DOI: 10.1016/j.infsof.2022.106926.

[Wie+25] Marion Wiese et al. "The TechDebt Game - Enabling Discussions about Technical Debt". In: *2025 IEEE/ACM International Conference on Technical Debt (TechDebt)*. Apr. 2025, pp. 23–33. DOI: 10.1109/TechDebt66644.2025.00010. (Visited on 08/26/2025).

[Wie26] Marion Wiese. *A Practical Guide to Establishing Technical Debt Management*. 2026. DOI: http://arxiv.org/abs/2601.11430. (Visited on 01/15/2026).

[Wie+26] Marion Wiese et al. "Establishing technical debt management — A five-step workshop approach and an action research study". In: *Journal of Systems and Software* 231 (Jan. 2026), p. 112606. ISSN: 0164-1212. DOI: 10.1016/j.jss.2025.112606. (Visited on 10/02/2025).

[Wil92] Frank Wilcoxon. "Individual comparisons by ranking methods". In: *Breakthroughs in statistics: Methodology and distribution*. Springer, 1992, pp. 196–202.

[Woh+12] Claes Wohlin et al. *Experimentation in software engineering*. Vol. 9783642290. 2012, pp. 1–236. ISBN: 9783642290442. DOI: 10.1007/978-3-642-29044-2.

[WRS21] Marion Wiese, Matthias Riebisch, and Julian Schwarze. "Preventing Technical Debt by Technical Debt Aware Project Management". In: *2021 IEEE/ACM International Conference on Technical Debt (TechDebt)*. arXiv: 2103.10317v1. 2021, pp. 84–93. DOI: 10.1109/TechDebt52882.2021.00018. (Visited on 03/19/2021).

[WSB26] Marion Wiese, Kamila Serwa, and Eva Bittner. "A Practical Guide for Establishing a Technical Debt Management Process". In: *2026 IEEE/ACM International Conference on Technical Debt (TechDebt)*. 2026. DOI: 10.1145/3794915.3795789. (Visited on 03/19/2021).

[Yan+19] Ye Yang et al. "Towards a taxonomy of technical debt for COTS-intensive cyber physical systems". In: *Procedia Computer Science* 153 (2019), pp. 108–117. ISSN: 1877-0509. DOI: 10.1016/j.procs.2019.05.061.

[YH+15] Jesse Yli-Huumo et al. "The Relationship Between Business Model Experimentation and Technical Debt". In: *Lecture Notes in Business Information Processing* 210 (2015), pp. 5–6. ISSN: 18651348. DOI: 10.1007/978-3-319-19593-3.

[YH+16] Jesse Yli-Huumo et al. "Developing processes to increase technical debt visibility and manageability – An action research study in industry". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 10027 LNCS. 2016, pp. 368–378. ISBN: 9783319490939. DOI: 10.1007/978-3-319-49094-6_24.

[YHMS16] Jesse Yli-Huumo, Andrey Maglyas, and Kari Smolander. "How do software development teams manage technical debt? – An empirical study". In: *Journal of Systems and Software* 120 (2016), pp. 195–218. ISSN: 0164-1212. DOI: 10.1016/j.jss.2016.05.018.

[Zam+21] Fiorella Zampetti et al. "Self-admitted technical debt practices: a comparison between industry and open-source". en. In: *Empirical Software Engineering* 26.6 (2021), p. 131. ISSN: 1573-7616. DOI: 10.1007/s10664-021-10031-3.

[Zaz+13] Nico Zazworka et al. "A case study on effectively identifying technical debt". In: *ACM International Conference Proceeding Series*. 2013, pp. 42–47. ISBN: 978-1-4503-1848-8. DOI: 10.1145/2460999.2461005.

[ZBA22] Ioannis Zozas, Stamatia Bibi, and Apostolos Ampatzoglou. "Forecasting the Principal of Code Technical Debt in JavaScript Applications". In: *IEEE Transactions on Software Engineering* (2022), pp. 1–15. DOI: 10.1109/TSE.2022.3222318.

[Zho+23] Xiaozhou Zhou et al. "Research on multilevel situation awareness changes under the cumulative effect of mental fatigue". In: *Cognition, Technology & Work* (2023), pp. 1–13. DOI: https://doi.org/10.1007/s10111-023-00723-99.

Appendix A

Appendix A – TDM Guide

The following Guideline provides practitioners with a guide for establishing a TDM process. It was published in the additional material to:

TDM Guide from the additional material of publication No. 7 *(Practical Guide)*	
Title	A Practical Guide for Establishing a Technical Debt Management Process
Authors	Wiese M.
Year	2026
Reference	[Wie26], additional material of publication [WSB26]

TABLE A.1: TDM Guide from the additional material of publication No. 7 *(Practical Guide)*

A Practical Guide to Establishing Technical Debt Management

A Whitepaper of

Marion Wiese

January 21, 2026

Inhalt

1 Introduction **3**
- 1.1 How to read the guide . . . 3
- 1.2 Terminology . . . 3

2 Fundamentals of Technical Tebt **5**
- 2.1 Defining the Term Technical Debt . . . 5
- 2.2 Technical Debt Types . . . 6
- 2.3 Cause-and-Consequences Model of Technical Debt . . . 6
  - 2.3.1 Visibility . . . 7
  - 2.3.2 Categories, Chains, and Cycles . . . 7
  - 2.3.3 Fowler Quadrant . . . 9
- 2.4 Activities for Managing Technical Debt . . . 10

3 Guide to Technical Debt Management **11**
- 3.1 Organizational Matters . . . 11
  - 3.1.1 Responsibilities – TD Manager . . . 11
  - 3.1.2 Tool Support . . . 11
- 3.2 Preventing Technical Debt . . . 12
  - 3.2.1 Indicators of Unconscious Incurrence of Technical Debt . . . 13
- 3.3 Identifying Technical Debt . . . 14
  - 3.3.1 Static Code Analysis . . . 16
  - 3.3.2 Indicators for Technical Debt . . . 17
  - 3.3.3 If in doubt... . . . 18
- 3.4 Managing Technical Debt . . . 18
  - 3.4.1 Document Technical Debt . . . 18
  - 3.4.2 Prioritize and Resubmit Technical Debt . . . 18
  - 3.4.3 Repay Technical Debt . . . 26
  - 3.4.4 Visualize Technical Debt in the Backlog . . . 28
- 3.5 Getting Started . . . 31

4 Typical Challenges in Managing Technical Debt **33**
- 4.1 Typical Mistakes . . . 33
  - 4.1.1 We all remember it. . . . 33
  - 4.1.2 The attributes all make sense – we use them all. . . . 33
  - 4.1.3 Forgetting the new attributes for non-technical debt issues . . . 33
  - 4.1.4 Alternatives? There aren't any. . . . 34
  - 4.1.5 Unconsciously accumulating technical debt: Planning Poker is just annoying . . . 34
  - 4.1.6 Unconsciously taking on technical debt: But that has to be included in the sprint now. . . . 34
  - 4.1.7 Consequence of technical debt: Poor maintainability . . . 35
  - 4.1.8 The components "Test," "Documentation," and "Logging" . . . 35
  - 4.1.9 Nothing is 'very' . . . 35
  - 4.1.10 Resubmission: As a precaution, next week . . . 36
- 4.2 Typical Questions . . . 36

4.2.1 Too many issues in the backlog—how do I get started? 36
4.2.2 How can I find alternatives? 36
4.2.3 Can I use other terms? 36
4.2.4 There are several consequences. How do I assess this? 37
4.2.5 How do I handle ongoing or recurring maintenance tasks such as updates? 37

5 Outlook and Scaling of Technical Debt Management 38

1 Introduction

This white paper provides an overview of the topic of "technical debt" and presents an approach for managing technical debt in teams. The white paper is based on the results of my dissertation (Wiese et al. [1, 2, 3, 4]), which aimed to translate scientific findings into practical guidance. To this end, I collaborated with other researchers to support three teams from different companies in adapting and establishing a technical debt management system tailored to their specific needs. Research findings were supplemented with details or additional approaches. Research results that were less practical were discarded.

The result is a guide on establishing technical debt management within a team. The guide is intended to provide orientation and not be a rigid framework.

We distinguish between "best practices" and "nice-to-haves." "Best practices" are understood to be all approaches that were adopted by all three teams. "Nice-to-haves" were used by at least one team. In many places, it is explicitly mentioned that the team should decide together how to design the process. This also applies, of course, to all areas where this was not explicitly mentioned.

This white paper explicitly does not cover the establishment of technical debt management across the entire company, but provides suggestions for this at the end.

1.1 How to read the guide

The white paper is written in a way that allows practitioners to quickly grasp the essential aspects and gain initial insights into the topic by skimming through it. In addition, all essential instructions for action have been highlighted in the margins.

Chapter 2 provides a rough overview of the basics of technical debt that are not explicitly part of the guide. If you are already familiar with the basics of technical debt, you can skip this chapter.

Chapter 3 is the core of this white paper and provides guidance on introducing a process for managing technical debt. In this chapter, we draw a fundamental distinction between "best practices," i.e., process elements that have been established by all three teams, and optional additions, i.e., process elements that have been used by some of the teams.

In Chapter 4, we list the problems that typically arise when introducing the process and provide tips on how to avoid them or which aspects require special attention. For a successful introduction, we recommend reading this chapter as well, as it contains many references to typical errors that might otherwise go unnoticed.

Ultimately, in Chapter 5, we provide an outlook on how the process can be extended to an entire company, identify existing obstacles, and discuss the potential advantages this approach could bring. This chapter is particularly interesting for managers or software architects who are considering a company-wide approach.

1.2 Terminology

This white paper uses some terms without further explanation. We have included an overview of these terms here. As these are common terms in the field of IT, the list serves primarily to clearly define terms and avoid ambiguities. A computer scientist working in an agile environment should be able to read the text without these explanations.

Term	Explanation
(The) Team	The team for which technical debt management is being introduced. Includes: developers, at least one person responsible for architecture, and the product owner.
User	Users of the IT system. In a company, this is usually not the person who pays for the IT system.
Customer	Company or an individual who pays for the system.
Functional Requirement	The desire (usually from customers/users) to change system functionalities or add new ones.
Quality Attribute	Implicit requirements for the system that are not described by the customer/user but are expected (e.g., performance, availability). The typical quality attributes are described by ISO 25010.
Non-function. Requirement	Requirements for quality attributes and other technical requirements (e.g., use of certain technologies based on company guidelines or existing expertise).
Issue	Requirement for a system that is stored in a tool. Various attributes are determined or recorded for each issue, e.g., assignee, open date, title, description, effort estimate, and keywords.
Issue Tracking Tool	A tool that stores issues with their attributes and allows them to be recreated/edited. The issues are displayed as a list or sorted by status in a "board." The list can be sorted and filtered. Typical examples are: Jira, AzureDevOps, GitLab Issues, GitHub Issues.
Issue Types	There are different types of issues. In addition to technical requirements (e.g., defined as user stories), there are usually bugs/errors. There are also issue types at other hierarchical levels: epics are a collection of issues that are related in terms of content, while tasks or subtasks are components of issues.
User Story	A way of describing functional requirements. Includes the requester, the requirement, and the rationale for the requirement. The term is often used (somewhat inaccurately) in place of the term "functional requirement."
Sprint	Periods of a few weeks (often 2 weeks). A goal is set for this period and the issues to be dealt with are planned.
Backlog	Planning tool for collecting issues per system/project. Issues can have different statuses (e.g., new, refined, in progress, in testing, completed).
Refinement	Meeting of the entire team to discuss and evaluate new issues. For each issue, implementation ideas are documented, the effort required is evaluated, and other attributes are estimated/determined.
Planning	Meeting of the entire team to discuss which issues should be addressed in the upcoming sprint.
Product Owner (PO)	The person in the team who collects technical requirements, prioritizes them, and has them evaluated by the team during refinement. The product owner determines the order of issues in the backlog and thus their priority.
DoR – Definition of Ready	Defines the requirements an issue must meet to be included in a sprint. For example, estimation is available, acceptance criteria are defined, and dependencies on other teams are clarified.
SP – Story Points	Type of effort estimation; often used in agile environments; an abstract or relative type of estimation. The complexity of the task is compared with the complexity of a similar task and assessed as equal, greater, or lesser.
PD – Person days	Type of effort estimation; often used in traditional methods; a concrete type of estimation. The effort is estimated in a real unit, i.e., in days.
Risk (vs. Drawback)	A risk is a drawback that has a probability associated with it. Example: "The license costs are very high" is a drawback. "We are dependent on the tool, and the license costs could increase" is a risk.
CoP – Community of Practice	Regular, voluntary meetings of people from a company to discuss a specific topic.

Table 1: Terms used in this white paper without further explanation.

2 Fundamentals of Technical Tebt

In this chapter, we provide a brief overview of the relevant basics on the topic of technical debt.

2.1 Defining the Term Technical Debt

The term was originally established by Ward Cunningham. In his paper, Cunningham wrote:

> *"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. [5]*

The term "technical debt" is therefore a (technical) metaphor for financial debt, and its terminology is based on terms familiar from the banking sector, such as interest or repayment. Interest is the additional expense that arises when technical debt is not repaid. Repayment is the expense that must be incurred to remove technical debt from a system. Interest in particular is often forgotten in the discussion about technical debt, or is difficult to grasp.

After some discussion about the exact definition, a group of researchers came up with the following definition:

> *In software-intensive systems, technical debt is a collection of design or implementation constructs that are expedient in the short term, but set up a technical context that can make future changes more costly or impossible. Technical debt presents an actual or contingent liability whose impact is limited to internal system qualities, primarily maintainability and evolvability. [6]*

The part entitled "future changes" is particularly relevant here, as it explains two aspects of technical debt:

- Technical debt is only a problem if you plan to change the system or the component affected by the debt. If the code is no longer to be changed, for example, in a legacy system that is to be replaced and will no longer be modified, then the technical debt in this code/system does not pose a problem.
- Over the past decade, agility has become a major topic in software development. A core element of agility is "embrace change," i.e., welcoming constant change in an IT system. This is also reflected in "continuous integration" and "continuous deployment" efforts. The code base and everything around it are now changing, in some cases, by the minute. That's why technical debt has become a problem in recent years. The consequences of technical debt become apparent when you want to change a system, which we now do every minute.

Another way to classify technical debt is based on the "four colors of the backlog" according to Kruchten et al. [7]. According to this classification, issues in the backlog of an issue tracking system can be differentiated according to whether they represent a positive or negative value for the system and whether they are visible or invisible to customers or users. This results in the classification shown in Figure 1.

In the scientific context, the term "technical debt item" has become established as a singular term for technical debt in order to distinguish between a specific problem in the code and a diffuse "technical debt in the system."

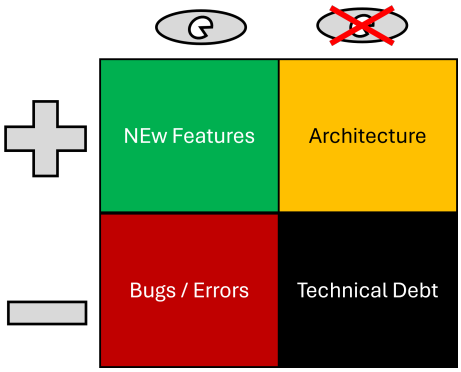


Figure 1: The four colors (classes) of the backlog according to Kruchten et al. [7]

2.2 Technical Debt Types

Technical debt is often only known as problematic code locations. TD exists, but not only in code, but also in many other software artifacts. Table 2 provides an overview of these different types.

Type	Example
Code	Unreadable code; code duplication
Architecture	Outdated architectural patterns, such as shared databases
Documentation	missing or outdated documentation
Test	Lack of test environment; insufficient test coverage
Infrastructure	Outdated servers; operating systems not up to date
Requirements	Non-functional requirements not taken into account
Build / Automation	No automated deployment; unstructured deployment pipeline
Security	Passwords in plain text; unsecured interfaces
Social	Communication barriers caused by the company's organizational structure; error-intolerant corporate culture
Versioning	Illegible code due to A/B testing or feature flags; problems with or lack of API versioning
Update	Outdated 3rd party libraries
Hardware	Use of outdated, less powerful components, e.g., CPUs in the "embedded" sector
and more	In science, further types of debt are still being identified and named

Table 2: Various TD types according to Li et al. [8]

2.3 Cause-and-Consequences Model of Technical Debt

Technical debt follows a cause-and-consequences model, i.e., there are factors that can trigger technical debt and consequences of technical debt that can lead to problems and visible symptoms. A special feature is that even the absence of action can lead to TD, e.g., if third-party libraries are not updated.

2.3.1 Visibility

A fundamental problem with technical debt is that different stakeholders have different perspectives on the situation. Figure 2 illustrates these perspectives:

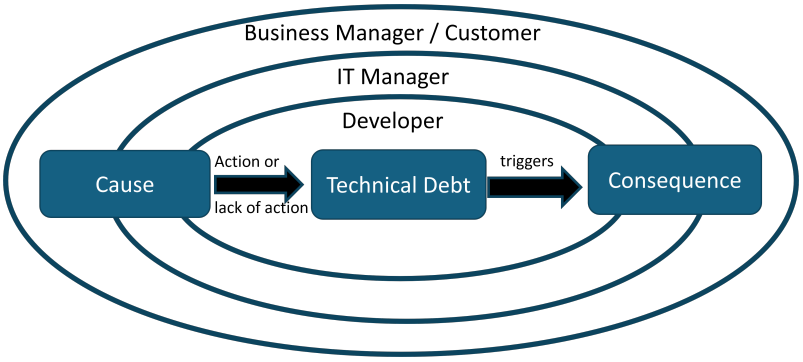


Figure 2: Visibility of technical debt for various stakeholders according to Wiese and Borowa [1]

- **Developers** have a strong focus on technical debt. They recognize the direct causes, such as deadline pressure or the lack of a specialist with the necessary knowledge. They recognize the direct consequences, such as development taking longer than estimated and missing deadlines. They often lack the bigger picture, i.e., the answers to the questions: Where does the deadline pressure come from? What consequences does the project delay have on the business?
- **IT managers** play a central mediating role. IT managers often understand technical debt themselves, but do not feel its effects directly. They are often much better able to answer the above questions than developers and therefore have a more complete picture.
- The **business managers**, such as customers or technical management, are very familiar with the causes and consequences. They see that they create time pressure. They realize that there are few good developers on the job market. Business stakeholders also see the consequences, e.g., missed deadlines or faulty systems and system failures. The problem is that they do not see the technical debt in the middle and therefore cannot connect the causes and consequences. For them, the middle circle in Figure 2 is not visible.

2.3.2 Categories, Chains, and Cycles

The model from Figure 3 shows a more detailed version of the cause-and-effect model. The causes and effects are classified into categories (see Table 3). In addition, the arrows indicate that there is usually not just one cause or one effect, but rather that these are often interconnected. Cyclical dependencies are also common. The inner cycles represent the "broken window" phenomenon[1], i.e., in an already dirty environment (with broken windows), the probability is higher that further uncleanliness will be added (further windows will be broken). In terms of technical debt, this means that places that already contain a lot of technical debt are most likely to accumulate further technical debt. The outer cycles result directly from the fact that the same categories can be found in both the causes and the effects.

[1] `https://de.wikipedia.org/wiki/Broken-Windows-Theorie`

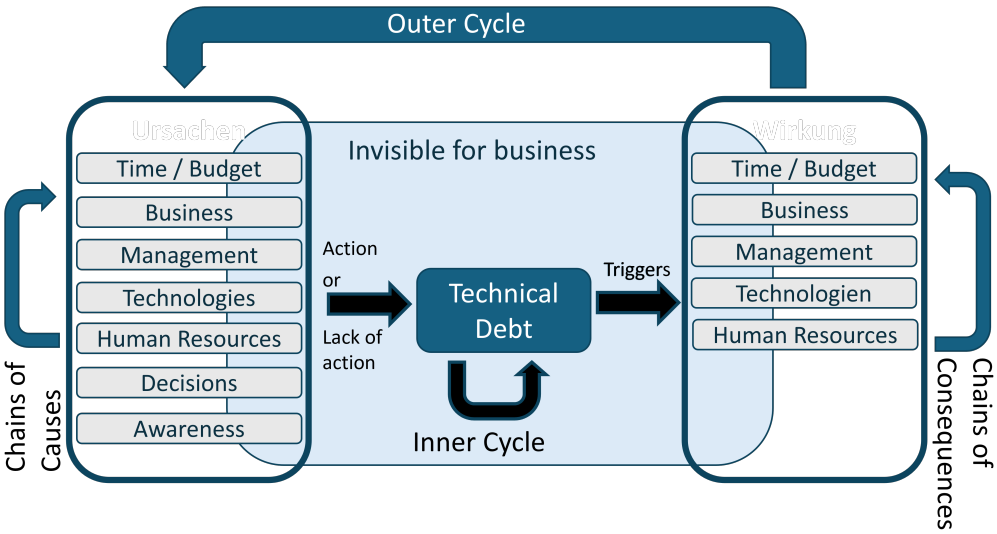


Figure 3: Causes, effects, chains, and vicious circles of technical debt according to Wiese and Borowa [1]

Category	Example of a cause	Example of an effect
Time/Budget	A project deadline that is set too tight	A project cannot meet the deadline
Business	The business strategy changes in the middle of the project, affecting the project	Customers are frustrated with product quality and are switching to a competing product
Management	Project deadlines are not planned with sufficient buffer time	Projects are delayed because developers have miscalculated or technical debt must first be cleared
Technology	The use of outdated technologies prevents the use of modern methods	Deployment dependencies prevent early rollout of a function that has already been implemented
Human Resources	Too many junior developers or developers who are not familiar with the system	Developers are frustrated by the consequences of technical debt and are leaving the company
(Faulty) Decisions	Management decision to implement a workaround instead of the correct solution; developers do not adhere to common code patterns	
Awareness	If a stakeholder is completely unaware that such a thing as technical debt exists, there is no reason for them to implement or have implemented a more complex solution	

Table 3: Different categories of causes and effects of technical debt according to Wiese and Borowa [1]

Examples:

- **Chain of causes:** A new law must be taken into account in the system and has a fixed date on which it will take effect (business – time). The project manager decides to implement a change in the code in a fixed manner rather than flexibly (management – decision).

- **Chain of consequences:** During development, a developer discovers that technical debt requires a redesign of a piece of code, changing the estimated effort from 2 days to 2 weeks (technology – time). The project manager is unable to compensate for this additional effort in the project plan and delivers the desired change with a week's delay (time – management). The customer is frustrated and considers switching products (time – business).
- **Inner cycle:** There is no test concept, and no test suite has been selected and is in use. Each new function is not tested, and each new function creates new test debt.
- **External cycle:** A shortage of good developers leads to avoidable problems in the code and reduced readability. Due to high staff turnover, there are only a few developers who are truly familiar with the code. The poor overall situation in terms of implementation quality leads to frequent failures, resulting in overtime and weekend work for developers. Well-qualified developers look for other employers.

2.3.3 Fowler Quadrant

M. Fowler provides another simple explanation of the causes of technical debt in his blog[2]. The quadrant he has created differentiates between technical debt that is consciously accepted and technical debt that is unconsciously incurred. It also differentiates between careless and wise decisions that can lead to technical debt.

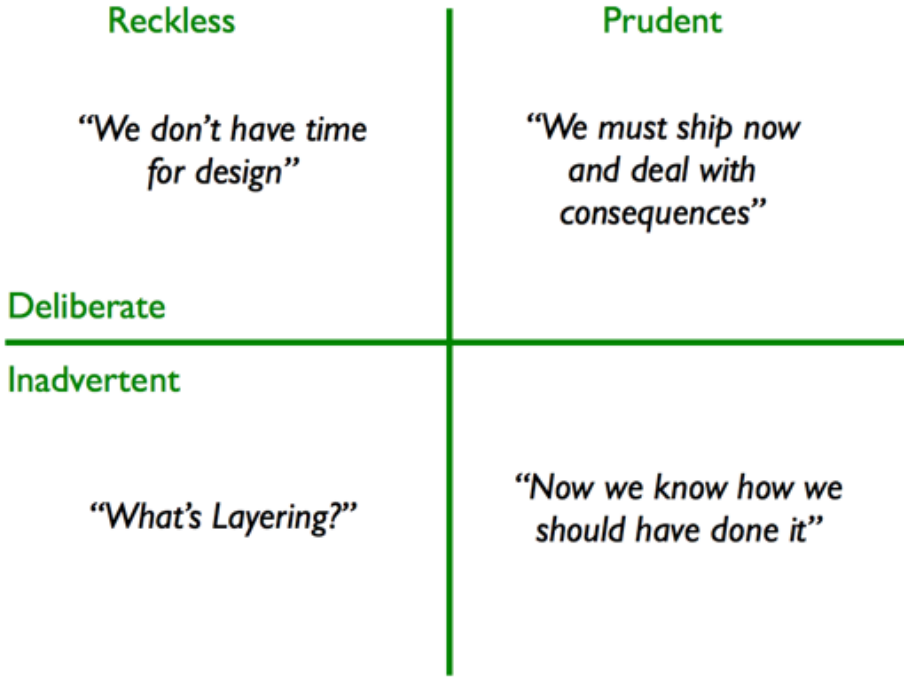


Figure 4: Causes of technical debt according to M. Fowler [9]

[2] https://martinfowler.com/

2.4 Activities for Managing Technical Debt

There are various aspects to managing technical debt. These are briefly explained in Table 4 and serve as the basis for the guidelines in Chapter 3.

Activity	Explanation
Identification	Identifying technical debt, e.g., through static code analysis, but also determining whether an issue from the backlog is a technical debt
Prevention	Prevent technical debt from being incurred, e.g., by making more conscious decisions and preventing carelessness
Documentation	Collecting information on each technical debt (e.g., description, list of tasks for dismantling, assessment of consequences) in order to gain an overview and be able to evaluate the technical debts
Measurement	Determining or estimating quantifiable characteristics of technical debt (e.g., dismantling costs or interest burden) in order to be able to evaluate the technical debt
Prioritization	Use the assessment (measurements) of technical debt to prioritize it against each other or against functional requirements
Repayment	Methods for structuring and sensibly organizing the repayment of technical debt
Monitoring (&Visualization)	Monitoring the system status in terms of technical debt using the recorded and measured properties

Table 4: Activities for managing technical debt according to Li et al. [8]

3 Guide to Technical Debt Management

In this chapter, we first explain a few organizational prerequisites for managing technical debt (Chapter 3.1). We then describe how to avoid technical debt (Chapter 3.2) and the elements needed to manage it (Chapter 3.4). Finally, we provide step-by-step instructions on how to get started with the process (Chapter 3.5).

3.1 Organizational Matters

Two key considerations for the organization are: firstly, the personnel requirements (Chapter 3.1.1) and secondly, the technological requirements (Chapter 3.1.2) for the process.

3.1.1 Responsibilities – TD Manager

> From the outset, there should be a person responsible for the process who acts as a so-called "technical debt manager" ("TD Manager").

The tasks of the TD Manager are:

- Coordination of the implementation process
- Technological responsibility for making changes to the issue tracking tool (changing existing issue types, creating new issue types)
- Technological responsibility for coordinating visualization and interfaces
- "Voice of reason," who repeatedly reminds everyone of the process and also considers the issue of "technical debt" in all discussions (e.g., to avoid it in decision-making situations).
- Contact person for technical debt for other teams within the company and participation in a community of practice for this topic.

3.1.2 Tool Support

> It is advisable to use the tool that is also used for work organization in everyday life. In an agile environment, this is the issue tracking tool used for the backlog (e.g., Jira, Azure DevOps, GitLab Issues).

> In addition to issue types for **functional changes** (e.g., user stories) and **bugs**, the selected issue tracking tool should also include a separate type for **technical debt**.

Other issue types are possible, but they are usually at a different level of abstraction, such as epics as a bracket around several issues or subtasks, and test cases as parts of an issue. In the following chapters, we describe how the different issue types should be changed to enable technical debt management.

Since issue tracking tools currently do not offer good support for visualization, it may be necessary to use an additional visualization tool (e.g., Microsoft Power BI[3], https://www.microsoft.com/en-us/power-platform/products/power-bi/, Tableau[4]).https://www.tableau.com/). It is advisable to

[3] `https://www.microsoft.com/de-de/power-platform/products/power-bi?market=de`

[4] `https://www.tableau.com/de-de`

ask your company which tool(s) are already in use there.

3.2 Preventing Technical Debt

Before addressing the management and repayment of technical debt, it is best to try to prevent it as much as possible.

To prevent technical debt, it is essential to examine its underlying causes more closely. As can be seen from the causes of technical debt, it often arises as a result of a decision (see Chapter 2.3). However, this decision can be made consciously or unconsciously (see Figure 4).

> The goal of avoidance is therefore not only to make better/smarter decisions (Table 6), but also to be aware of the decisions (Table 5). These decisions are not made when discussing and evaluating technical debt issues, but rather for all issues, especially those issues that aim to make functional changes. To avoid technical debt, it therefore makes sense to adapt the issue template for functional requirements.

The bug template or epics can also benefit from this change, but this depends heavily on how these issue types are used.

Below, we present adjustments that help the team avoid technical debt **in all issue types**.

For factors that serve to make the decision conscious (Table 5), checkboxes are sufficient, as the aim is simply to remind the team of certain facts. Instead of a checkbox, the question can also be included in a Definition of Ready (DoR) if this is used consistently by the team.

For the factors in decision-making (Table 6), it is up to the team to decide whether the findings are stored as free text in the issue or whether only a check box is used to remind them to discuss these topics.

Attribute	Explanation	Field / Attribute type
Talked about TD	The check box serves as a reminder that technical debt should generally be discussed in the context of the issue. It is checked off when the team determines that the issue does not generate technical debt, or after an issue of the type "technical debt for dismantling" has been created for deliberately incurred technical debt.	check box
Is this issue a repayment of technical debt?	This question serves as a safeguard for teams that cannot quickly and easily determine whether new issues involve technical debt reduction. If the question is answered with "No," it is marked as answered. If the question is answered with "Yes," the type and template for this issue are switched.	DoR or check box
Does this issue create technical debt?	This check box serves as a reminder that the question should be answered during the discussion of the issue. If the question is answered with "No," the question is marked as answered. If the question is answered with **"Yes'**, a discussion will take place as to whether it is wise to accept technical debt here (see Table 6) and, if necessary, **another issue for dismantling this technical debt** will be created, evaluated, and linked to this issue.	DoR or check box

Table 5: Methods for making conscious decisions regarding technical debt. (Use of the first or the last two attributes)

Attribute	Explanation	Field / Attribute type
Quality attributes	For each issue, you should consider whether a quality attribute is particularly affected by the implementation. If quality attributes are impaired, e.g., performance could deteriorate, this provides a direct transition to the next point.	label field or free text
Drawbacks	Discussing and, if necessary, recording the drawbacks of a solution helps to raise awareness of these drawbacks so that they can be addressed in the workplace. It also helps to think about other solutions (alternatives, see below) that do not have these drawbacks.	Check box or free text
Risks	Discussing and, if necessary, recording the risks of a solution is helpful in developing mitigation strategies for these risks. However, it is also helpful to identify other solution options (alternatives, see below) that do not carry these risks.	Check box or free text
Alternatives	Comparing different alternatives helps you decide on a solution that carries the least risk of creating technical debt, either immediately or over time.	Check box or free text

Table 6: Methods for making wise decisions regarding technical debt

3.2.1 Indicators of Unconscious Incurrence of Technical Debt

When refining or planning issues, there are various indicators that may suggest that there is a risk of (unconsciously) incurring technical debt. We have included a list of such indicators here, which can help readers remain alert in certain situations and, if necessary, steer the team's discussion toward this issue.

Statements that indicate unconscious incurrence of technical debt:

- "I don't quite understand that yet, but it's not important anyway."
- "If we can't do it that way, we'll have to see how it fits.‘'
- "First, let's come up with a small solution.‘'
- "We'll do that later.‘'
- "That is extra bells and whistles." (i.e., some special solution for a unique situation/customer)

Circumstances that indicate the unconscious incurring of technical debt:

- The content of the issue is not yet clear, but the issue will be included in the sprint. Specifications still need to be clarified.
- The solution approach for the issue is not yet clear, but the issue will be included in the sprint.
- The process is not being followed; issues that have not (yet) been discussed in refinement are being included in the sprint
- The developer who is supposed to implement the issue is not involved in the refinement.
- Acceptance criteria are not specified.
- No planning poker was performed, resulting in a distorted and possibly too low estimate.
- The product owner decides on the effort or provides their own estimate to the developers, resulting in a distorted and possibly too low estimate.

- Creation of a ticket for testing so that the functional ticket can be closed even though the test is still missing.
- Recognition that technical debt is being incurred (e.g., code duplication), but discussion of this is discontinued.
- Decisions in favor of a technology, even though the requirements are not yet clear.
- Providing information to users, even though the technical details have not yet been clarified. (Is this possible within the time frame? Is it even possible at all?)
- Installation of special features, wanting to fulfill all user requests, even if they are very complex, not disclosing the financial consequences of their requests to users.

3.3 Identifying Technical Debt

The first and challenging task in managing technical debt is determining whether an issue is technical debt in the first place.

The definitions presented in Chapter 2 can help with this, but they do not apply to all practical situations, for example, when the team does not have direct customers but provides a central service for other development teams. It is also usually not enough to simply look at whether the team incurs any costs ("additional costs'', "interest'') as a result of not implementing the issue.

One method that works relatively well and is easy to use is to focus on two questions:

- Who wants this issue to be addressed? (Who is suffering from the issue?)
- Who is willing to pay for the issue to be addressed?

Figure 5 shows a matrix with the resulting classification.

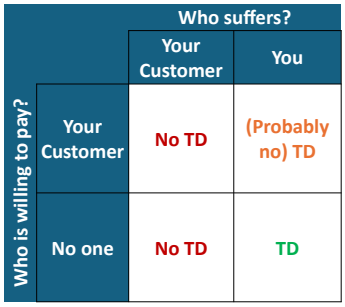


Figure 5: A decision matrix for identifying technical debt

However, when scaling this matrix to include multiple stakeholders, not all decisions are straightforward, as shown in Figure 6. The team has to decide what to interpret as TD and what not to. The presented matrix can support this discussion.

In addition, the simple flowchart in Figure 7 helps with classification. We begin with the fundamental idea that resources for reducing technical debt are scarce. If you imagine the resources for the different issue types as pots, you want as little as possible to end up in your own pot for technical debt. This raises the question of whether – even if the team would benefit from reducing technical debt –another pot could be used to pay for it. This also means that technical

Who is willing to pay?	Who suffers?			
	Your Customer	Your Company	Your IT unit	You (Your Team)
Your Customer	No TD	(Probably no) TD	(Probably no) TD	(Probably no) TD
Your Company	No TD	(Probably no) TD	Probably TD	Probably TD
Your IT unit	No TD	Probably TD	Probably TD	TD
No one	No TD	Probably TD	TD	TD

Figure 6: A decision matrix for identifying technical debt, including various stakeholder perspectives

debt can be more clearly assigned to different hierarchical levels (at the team level vs. at the IT department level).

To illustrate the branches, example situations for each decision are listed here (see numbering in Figure 7):

1. Code is difficult to read, which means that further development in this part of the code takes longer. Every morning, a manual production step has to be carried out because there is no automation. Special case: If the additional work needs to be carried out by another IT team (e.g., administrators or support team), this can still be answered with "Yes" in order to do the other team a favor, for example.

2. Error messages repeatedly appear in the log, which do not cause any additional work for the developers. Nevertheless, the developers feel disturbed by this because they would like to have a clean log.

3. The system's performance is poor when booting. Users simply get their morning coffee during the boot process. However, developers must reboot the system several times an hour if they make changes to the boot process.

4. The error messages in the log refer to error messages in the interface that the user has also seen. Users may not have reported the error (yet) because they can simply click away the error message. The customer (PO) decides to expand this error despite the lack of complaints from users.

5. The database used by all IT teams would need to be updated to utilize the new functionalities.

6. To standardize the tool landscape, the company decides that all teams should switch from their issue tracking tool to Azure DevOps.

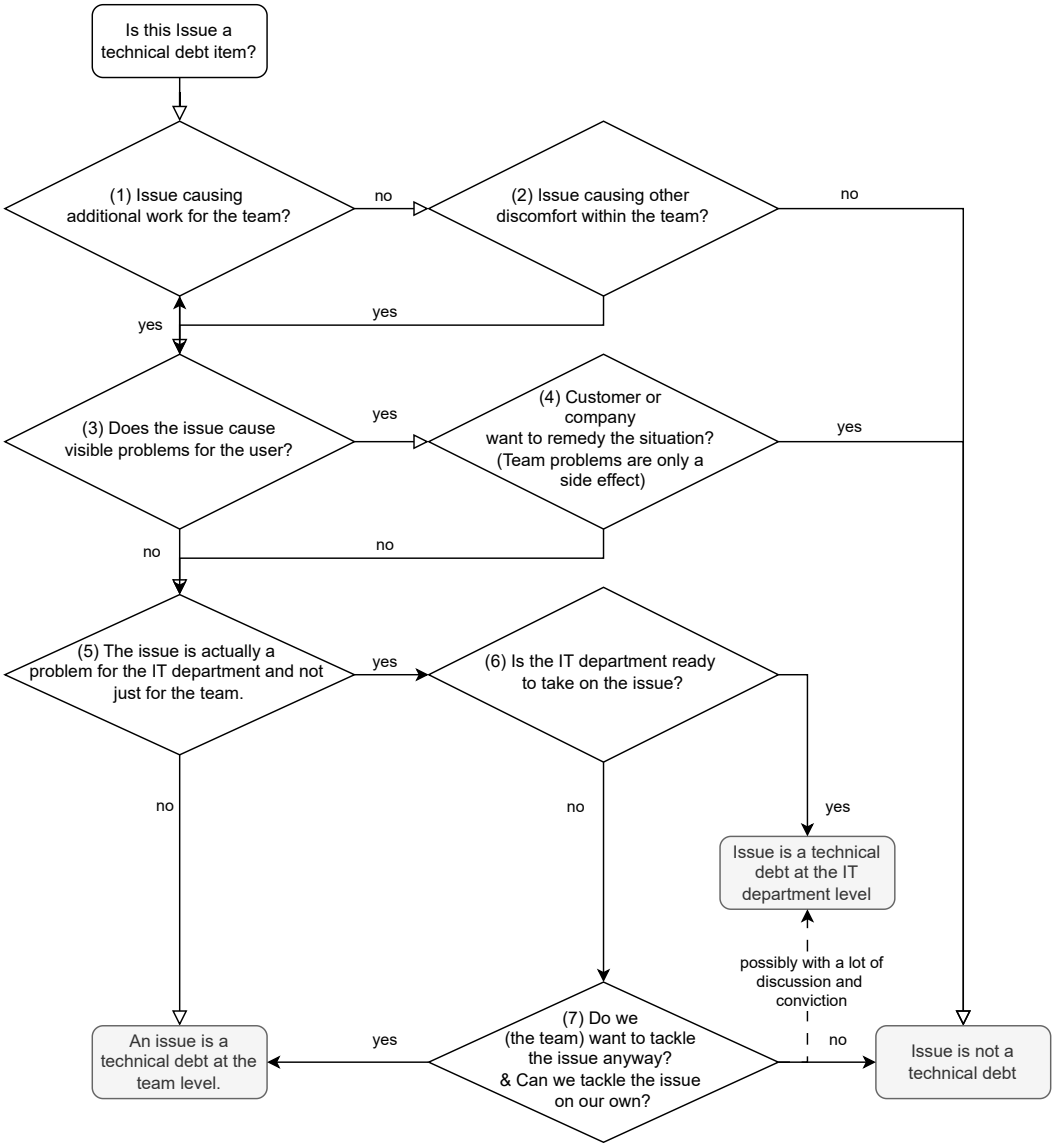


Figure 7: Identifying technical debt (TD), an orientation

7. The team needs a monitoring tool for secure maintenance. The IT department has no central guidelines for this and no licenses for any tools in stock. The team decides to use Grafana and purchase the licenses from its own budget.

3.3.1 Static Code Analysis

A static code analysis tool, such as SonarCloud or NDepend, can help identify technical debt. However, these tools currently have significant limitations:

- This method only finds technical debt in the source code, while other types of debt are not

taken into account.

- Many supposed technical debts are found that are not debts at all, so-called "false positives." These false positives cause an effort to identify the actual debts in the mass of reports.
- The relevance of the technical debts found is often low, and it is not clear whether they cause any damage at all. The debts identified directly by the developers, on the other hand, have always already caused additional effort for them.

Nevertheless, static code analysis can be a valuable addition to the process described here, as the technical debts identified can be recorded as issues and incorporated into the process.

3.3.2 Indicators for Technical Debt

In addition to the matrices presented above, there are various indicators for technical debt. Indicators suggest that technical debt may be present. However, the issues are not necessarily technical debt; the final decision rests with the team and is based on their experience and expertise. Below, we present the indicators in bullet point form:

- The use of certain terms in the title of the issue. The following terms (and their linguistic variations) are indicators:
  - *Optimizing, improving, revising, testing, documenting, performance, upgrading, updating, refactoring, cleaning up, converting*
- Discussing the issue:
  - "We did it quickly back then."
  - "We shouldn't have done it that way."
  - "Now we want to do it right."
  - "We implemented it hard-coded at first."
  - "Now we're doing it right."
  - "I forgot to expand it back then."
  - "At the time, we thought that would be good."
  - "It turned out that this is not optimal."
  - "Special cases would then be eliminated if we implement this."
  - "I don't understand what you're talking about, but it doesn't matter." (from the PO)
- The issue is not presented by the product owner, but by a developer or architect.
- Adapting cross-cutting concerns (e.g., testing procedures, logging mechanisms, monitoring)
- Use of new or changes to work tools (e.g., monitoring, documentation, issue tracking, error tracking)
- Fixing an error or simplifying a task for which a workaround already exists
- Moving a functionality/method to another area (class/package/...)
- Improvements to resilience (e.g., adjusting timeout)
- Product owner does not understand the issue.
- Communicating the solution and time frame to the user before consulting with the developers

3.3.3 If in doubt…

If the team is unsure whether an issue constitutes technical debt, it makes sense to reflect on the purpose of the classification. We classify an issue as technical debt so that we can then use a different template. This template serves the purpose of making the costs of technical debt visible and prioritizing technical debt.

> So if it makes sense to evaluate the issue in question (e.g., to justify implementation) or prioritize it (e.g., to rationalize discussions about its importance), then it makes sense to classify it as technical debt.

3.4 Managing Technical Debt

In this chapter, we provide an overview of the components of technical debt management: documenting (Chapter 3.4.1), prioritizing (Chapter 3.4.2), repaying (Chapter 3.4.3), and visualizing (Chapter 3.4.4).

3.4.1 Document Technical Debt

> As with all IT project planning, assessing technical debt can only be based on expert estimates.
>
> Various attributes have been established for the team to assess when evaluating technical debt. We distinguish here between "best practices," i.e., attributes that should always be assessed, and "nice-to-haves," i.e., attributes that can be additionally assessed depending on the team and/or company.

The Table 7–8 on the following pages list these attributes together with their explanations, field types, and, where applicable, contents, with the "nice-to-have" attributes highlighted in gray.

3.4.2 Prioritize and Resubmit Technical Debt

Determine Priority Priority is usually measured on a scale of five values, with the names for the levels varying. In some cases, a number between 1 and 5 is assigned. Alternatively, priorities can be labeled as "very low" to "very high." Some companies have established more specific labels, such as "guaranteed," "blocker," "critical," "medium," and "low."

> Prioritization is usually done on a 5-point scale. No new scale should be invented for technical debt; instead, the existing scale should be used. Since no best practices have been established yet, we present three variants for determining priority: "educated guess," "mean value," or "return on investment."

Field/attribute	Explanation	Example	Field/attribute Type
Risks of non-repayment (benefits of repayment)	What can happen if this technical debt is not repaid? There can be several consequences for technical debt at the same time. It is essential to list all consequences (including drawbacks and risks) here. You can use the 5 Whys method for this (see Chapter 4.1).	Additional work for developers, risk of errors in the system, performance losses, and security risks.	Free text; content: bullet point list
Repayment effort (SP)	For planning in the sprint, an effort estimate should be made in the unit that is also used for other issues.	Story points or T-shirt sizes	Depending on content: Integer / Text / Combo box
Contagiousness	Does the cost of repayment increase over time?	An API with technical debt initially has only one consumer that needs to be changed, but over time, additional consumers are added, all of which then also need to be changed.	Combo box with "will decrease," "will remain the same," and "will increase" as options
Interest probability or frequency	How likely is it that interest will actually be incurred? In principle, this assessment should be made for all "risks of non-repayment." A pragmatic approach that has become established is to assess the most serious risk of non-repayment.	How likely is it that someone will need the missing documentation? (1x/year) How likely is it that the technical debt will result in a production error that causes additional work? (1x/quarter) How often does a workaround have to be performed (manually) by the developers? (daily)	Combo box: 1x/day; 1x/week; 1x/month; 1x/quarter; >= 1x/year(half year)
Interest	For the most serious risk of non-repayment, an estimate is made here of how much effort will be required in development if the case occurs and interest has to be paid.	(Similar to the above): 1 day of additional effort to understand the code without documentation. 4 hours of effort to find and fix the bug and deploy the hotfix. The workaround requires 15 minutes of a developer's working time.	Combo box: 15 min.; 1 hr.; 4 hrs.; 1 day; >= 2 days
Priority	see Chapter 3.4.2	similar to other issue types	Combo box with 5-point scale

Table 7: Useful attributes for a "technical debt" issue – Part I (The "nice-to-have" attributes are highlighted in light gray)

Field/attribute	Explanation	Example	Field/attribute Type
Resubmission date	see Chapter 3.4.2		date field
Component	The (architectural) component (system part, program part) in which the technical debt is located or has an impact.	Technical debts in the checkout process of the online shop, in the tax calculation of the booking system, or even in the front-end, during database access.	Hierarchy (usually already included in the standard)
Pain Factor In addition to the hard factors, it sometimes makes sense to consider the developers' subjective assessment.	There is a workaround that is also economically viable, but this recurring task can still be frustrating for developers.	Integer or combo box (scale from 1 to 5)	
Quality attributes	List of influenced quality attributes. This information can be helpful if certain quality attributes are particularly important to the business or for planning quality initiatives.	In a bank: Technical debt that negatively impacts security. In the embedded domain: Technical debt that negatively impacts performance.	Keyword field, if possible, otherwise free text
Repayment effort (PD)	To determine a return on investment (ROI) value (see Chapter 3.4.2), it is necessary to estimate the effort in person-days (PD) as well.		Numbers / Integers
Risks of repayment	Technical debt increases the risk of side effects in further developments. Technical debt is often found in system components with a significant amount of other technical debt, resulting in a high risk of side effects. Such risks are listed here; if necessary, their priority can be lowered.	Risk that other parts of the system cannot be further modified during this time; impact on other teams (e.g., when changing an API)	Free text; content: bullet point list
Breaking Change	As an alternative to the "Repayment risks" field, you can simply use a check box to indicate that the issue may have potential side effects.	see above	Check box

Table 8: Useful attributes for a "technical debt" issue – Part II (The "nice-to-have" attributes are highlighted in light gray)

Educated Guess: In this case, the priority is determined jointly by the team. Compared to the previous approach, however, the information required to set the priority in an informed manner is available.

It is essential that team members agree on the relevant attributes that should be considered when prioritizing. For example, the expectation that a situation will occur (e.g., new corporate/IT strategy, new developments in the IT market) could be taken into account. However, it has been found that it makes more sense not to take this expectation into account, but instead to set the resubmission date accordingly (see the section on "Resubmission" below). Another example is the consideration of repayment costs. These cannot be taken into account in the "educated guess" either if there is a "low-hanging fruit" visualization (see Figure 8). With the help of this visualization, the combination of priority and effort can then lead to an order for repaying technical debt items.

This is the "simplest" and most pragmatic option. However, it is also a very subjective option, i.e., subject to the developers' subliminal preferences when making the assessment, and therefore possibly less reliable than the following options.

Mean value:

In this case, all attribute values are assigned a value between 1 and 5, and the mean value of these values is then used as the priority. An overview of the assignment is provided in Table 9. Not all of the listed attributes need to be used to calculate the mean value. The decision on this is up to the team.

Attribute	Value	assigned Integer	Remark
Interest	15 min.	1	
	1 hr.	2	
	4 hrs..	3	
	1 day	4	
	$\geq$ 1 day	5	
Interest probability	$\geq$ 1x/year	1	
	1x/year	2	
	1x/month	3	
	1x/week	4	
	1x/day	5	
Pain Factor	1-5	1-5	The severity of the stress should already be measured on a scale of 1-5
Quality attributes	n	1-5	Number n of quality attributes affected in addition to maintainability
Contagiousness	decreases	0	
	stagnates	3	
	increases	5	

Table 9: Integer assignment to attributes for a mean value calculation

Contagion must be considered separately, as it is not a linear change. If repayment costs decrease over time, the overall priority should also decrease, which is achieved by setting a value of 0. If repayment costs remain the same, the value should not affect the mean values. However, this is difficult to achieve mathematically because the calculated mean value can vary from issue to issue. The values given here can therefore only be used to create an approximation. The team is free to flexibly adjust the integers of the contagion to the situation when determining each mean value.

The repayment costs are not taken into account here (in contrast to the ROI variant). By means of a "low-hanging fruit" visualization (see Figure 8), the combination of these two values can lead to a sequence for repaying individual technical debt items.

The advantage of this method is that the calculation can be done in your head and is therefore more pragmatic and reliable than an "educated guess."

Return on Invest (ROI): The ROI is calculated using the interest rate, the probability of interest, and the effort in person-days. The result is the number of months it takes for the effort to repay the technical debt to break even. To calculate the ROI, we set the following values for the combo box fields:

- Interest in minutes: 15; 60; 240; 480; 960
- Interest probability in "per month": 30; 4.5; 1; 0.0027; 0.0013
- Expense in PD: as specified

From this, we calculate the monthly interest burden, as specified in Forumla 1, and the ROI in months, as specified in Forumla 2.

$$\textit{interest burden}\ \ \text{minutes/month} = \textit{interest}\ \ \text{minutes} * \textit{interest probability}\ \ \frac{1}{\text{month}} \tag{1}$$

$$\textit{ROI}\ \ \text{months} = \frac{\textit{effort}\ \ \text{minutes}}{\textit{interest burden}\ \ \text{minutes/month}} \tag{2}$$

The ROI can then be used to derive priorities in various ways:

- The ROI serves as additional information for an "educated guess" priority. This means that not only does the ROI determine the final priority, but other factors (e.g., additional quality attributes affected; see section below) are also taken into account in the determination.
- The ROI is directly transferred to a priority. Priorities are about ratios, i.e., the question of whether one technical debt is more important than another. Therefore, based on the determined ROI, it can be distributed as evenly as possible among the existing priorities. This distribution may vary from company to company. An example is shown in Table 10.

The advantage of ROI is that it presents a profit that is easily understood by management. ROI is based on the same principles as project planning as a whole, i.e., it is based on expert estimates. However, this also means that it is subject to the same inaccuracies. The team should decide together with management whether accuracy is desired and make it clear that the estimates (as in project planning) may be subject to errors.

ROI	Prio ID	Priority
$<$ 1 month	5	very high
$<$ 2 months	4	high
$<$ 1 year	3	medium
$<$ 3 years	2	low
$\geq$ 3 years	1	very low

Table 10: Example of determining priority based on return on investment (ROI). The actual distribution may vary depending on the team/company.

Resubmission In an agile environment, there are generally two types of prioritization. On the one hand, each issue is assigned a priority. On the other hand, issues in the backlog can be moved up and down, i.e., the order of the issues in the backlog specifies a further—usually more relevant—priority. Issues related to technical debt, as well as some other issues, are repeatedly prioritized downwards and eventually end up at the bottom of the backlog. If the number of such issues becomes excessive, they are no longer reviewed. The following procedure has therefore become established:

> Each technical debt issue is assigned a resubmission date. In each refinement, the list of all technical debts is then sorted by resubmission date, and all issues with an expired resubmission date are reevaluated.

To avoid starting from scratch every time a reassessment is carried out, it is essential that all relevant information and decisions are recorded in writing in the issue. In particular, it should be noted, for example, in a comment, if a specific situation is anticipated that will lead to a change in the assessment. Examples: "Reassessment necessary after business decision <xy>" or "Issue should be obsolete after system <xy> is shut down."

A resubmission date has the following advantages:

- Issues are not permanently overlooked at the end of the backlog (as explained above).
- If the priority depends on a business decision that has not yet been made, the resubmission date can be set to the estimated date of this decision in order to reevaluate the issue.
- If a change is planned that will render the issue obsolete in the near future (e.g., switching on or migrating a system), the date can be set to the date of the expected shutdown. On the resubmission date, a decision will be made regarding whether the issue can be closed.
- If the priority of the issue is low and it makes sense not to repay the technical debt but to pay the interest permanently, the resubmission date can be set. On the date, it will be checked whether repayment has now become sensible due to interim changes. The issue thus remains open and can also be used for risk assessment (see Chapter 3.4.3) without being repaid.

Quality attributes As already mentioned in Table 8, quality attributes can be used for prioritization. These can be used, for example, to overlay a calculated priority based on interest rates, interest probability, and effort, e.g., when security risks need to be minimized.

Depending on the context (system/industry/company), different quality attributes are most important. For example, security is most important in the context of banks, whereas performance (efficiency) and safety may play a greater role in the embedded sector.

> Each team should discuss which quality attributes are, in general, most relevant to their own system in their own context/company. To do this, the quality attributes should be put in order. The aim is not to establish a fixed and universally valid order, but to discuss the attributes and develop a common understanding of which attributes are relevant to their own system. The relevant quality attributes can then be used to influence decision-making/prioritization.

In Table 11, we briefly introduce all quality attributes according to ISO-25010 and explain them in simple terms using our own words and example questions. Those who are already familiar with ISO quality attributes are welcome to skip this section.

In addition to the ISO quality attributes, other quality attributes (e.g., "sustainability") or the sub-categories of the ISO standard (e.g., "installability" as part of "flexibility") may be useful and can be applied. It is also perfectly legitimate to use old terms, such as "performance" as an old term for (and component of) "performance efficiency", or "usability" instead of "interaction capability". The only important thing here is that there is clarity and agreement within the team about the meaning of the terms.

In practice, maintainability, security, performance, and reliability are often influenced by technical debt, whereas the other quality attributes are less frequently impacted.

Quality attributes	Sub-Categories	Explanation
Functional Suitability	F. completeness; F. appropriateness; F. correctness This concerns the appropriateness of the chosen solution, for example, the question: Does the solution truly meet the users' needs? From a technical debt perspective, this also includes, for example, systems/system components that are no longer in use and should be shut down to save on maintenance costs.	
Performance Efficiency	Time Behaviour; Ressource utilization; Capacity	This is about the efficiency of the solution. Does the chosen solution perform well? How many resources (e.g., memory) are required? The last point now also encompasses issues of sustainability in terms of environmental and climate concerns.
Compatibility	Co-Existence; Interoperability	This concerns the interaction of different systems. Can the chosen solution be integrated into the existing system landscape?
Interaction Capability	z.B.: Learnability, User error protection; Inclusivity; u.v.m.	Before 2023, this area was called "usability," and this term is usually easier to understand. It refers to the ease of use for the user. Is the system easy to learn? Is the user protected from user errors, e.g., through warning messages? Can the system also be used by people who are blind/deaf?
Reliability	Faultlessness; Availability; Fault Tolerance; Recoverability	This concerns errors that can occur in a system. How often/how many errors occur? Are there system crashes? How long must the system be available per year?
Security	z.B.: Confidentiality; Integrity; Authenticity u.v.m.	This is about protecting the system against misuse. Is the data encrypted? Is it possible to trace who changed which data?
Maintainability	Modularity; Reusability; Analyzability; Modifiability; Testability	This is the real core of technical debt. How easy is it to change the system? Is the system modular? Can the code be easily read by developers who are not familiar with it? (Note: Testability does not refer to the existence of tests, but rather whether the code is written in such a way that it can be easily tested.)
Flexibility	Adaptability; Scalability; Installability; Replaceability	The question here is whether the system can be easily adapted to other circumstances. How easily can peaks in demand be handled? How easily can the database be replaced?
Safety	z.B.: Risk Identification; Fail Safe; Hazard Warning u.v.m.	This concerns safety when using the system. Could misuse result in injury to a person (e.g., in mechanical engineering)? Are users warned before they endanger the environment, including the people in it (e.g., nuclear power plants, MRI)?

Table 11: Quality attributes according to ISO-25010 [10]

3.4.3 Repay Technical Debt

There are various approaches to repaying technical debt, each of which is appropriate in different situations. Situational repayment has therefore proven to be best practice. This means that there is no single perfect method for repaying technical debt; rather, different approaches are appropriate in different situations. Table 12 provides an overview of the various repayment methods and the situations in which each method is appropriate (use case). The repayment methods can be divided into three categories: ignore, rewrite, and refactor. For the sake of completeness, we include "Ignore" as a repayment method, even though no explicit repayment takes place and the term "repayment" may be misleading in this context.

Each team should decide for itself which methods it wants to use. In particular, a decision should be made regarding whether to use a quota method (e.g., 15

If a team is under a lot of pressure from customers and new customer requests and/or has a very confusing backlog with many "old" issues, it has proven useful to work with a quota solution. Teams that can work more flexibly and independently have generally not found the quota solution necessary. The decision on whether a quota solution is suitable should be made by the developers. The scope may be negotiated with the product owner or management.

Category	method	Explanation	Usage-Scenario
IGNORE	Impediment/ Road-Block	No one cares about technical debt. The debt is not reduced. Interest is paid (unconsciously) on an ongoing basis. Repayment only occurs if further development is not possible without repayment. The risk that further development will no longer be possible is (unconsciously) accepted.	A purely functional/customer-driven development; no awareness of and no management of technical debt.
	Pay interest	The debt is not being reduced. Interest is being paid on an ongoing basis (or until the next resubmission date).	A Repayment is not reasonable for economic or other reasons, such as affected quality attributes or developer frustration.
REWRITE	Magic	Technical debts "disappear" without explicit repayment because systems are rewritten, replaced by a purchased system, or simply shut down.	Rewriting when using very outdated technologies or as part of planned migrations (e.g., cloud migration); shutting down systems/parts of systems that are no longer used by users; replacing self-developed systems with market provider systems for economic reasons.
REFACTOR	Calculate/prove benefits	Every technical debt must "prove" its usefulness, e.g., because an ROI will be achieved in a few months. Repayment of "low-hanging fruit," i.e., issues with high priority and low effort.	The product owner should schedule these issues in the list of all issues for a sprint because their benefits are obvious. Schedule them when there are only a few SPs left to allocate in the planning, or when an issue in the sprint was completed faster than expected.
	Contingent	Setting a quota for repaying technical debt: x% of a sprint's SPs; every xth sprint solely for repayment; project for repaying technical debt ("quality initiatives")	For teams that are generally under a lot of pressure from customers and new customer requests, to ensure regular repayment and maintenance.
	Evolution-based	Repayment of technical debt in an (architectural) component that is being changed as part of an epic/project/feature. Goals: (1) Minimize the risk of technical debt causing disruption. (2) Leverage synergies because the relevant section is being changed and tested anyway.	When a major change to a specific component is planned.
	Polluter pays principle	Repayment of technical debt incurred as part of a major change (e.g., epic, project, feature development). Create an issue for each piece of technical debt incurred and repay it after the change has been deployed.	When a major change must be completed by a certain date and technical debt is incurred in order to meet that deadline.

Table 12: Repayment methods and the associated application scenarios

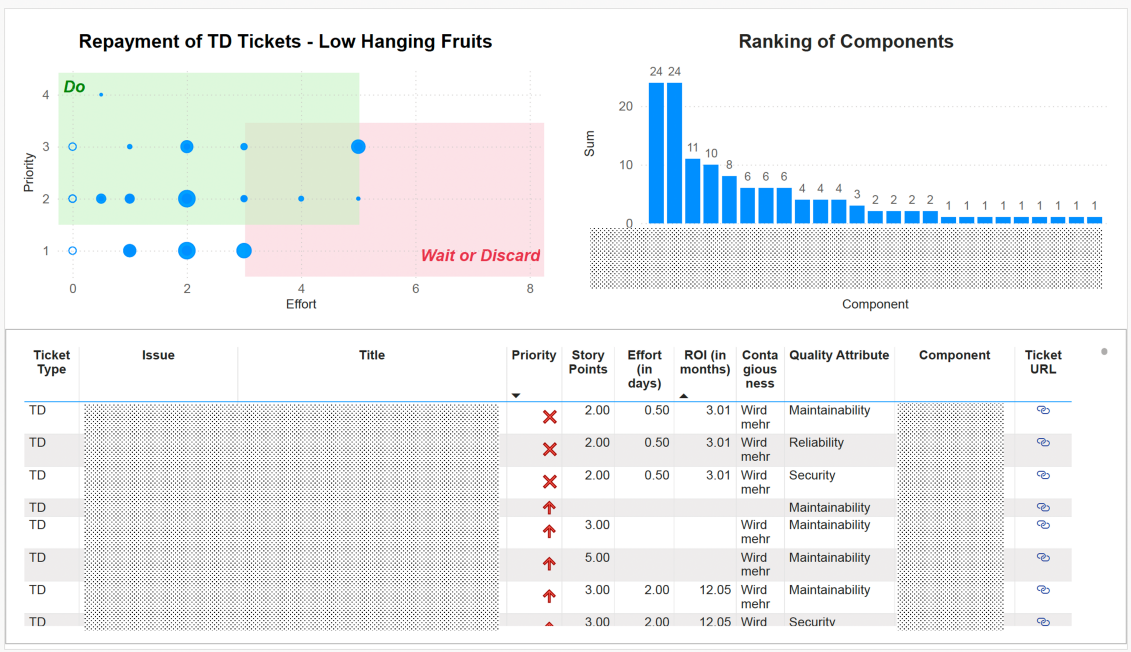
3.4.4 Visualize Technical Debt in the Backlog

Technical debt is visualized based on its issues and the estimated values for the attributes (see Table 7–8).

The visualization has two main objectives:

- Support for planning, i.e., prioritization and monitoring of technical debt
- Raising awareness of technical debt, especially among non-technical stakeholders such as customers, business management, project managers, etc.

> Interactive visualizations (low-hanging fruits, bar charts by component or quality attributes) have become particularly popular for planning purposes, as they filter an additional list of technical debt issues. This list should display the following attributes and be sortable by all attributes: ID, title, priority, contagion potential, interest, interest probability, interest burden, ROI, resubmission date, link to the original issue.



Ticket Type	Issue	Title	Priority	Story Points	Effort (in days)	ROI (in months)	Contagiousness	Quality Attribute	Component	Ticket URL
TD				2.00	0.50	3.01	Wird mehr	Maintainability		
TD				2.00	0.50	3.01	Wird mehr	Reliability		
TD				2.00	0.50	3.01	Wird mehr	Security		
TD								Maintainability		
TD				3.00			Wird mehr	Maintainability		
TD				5.00			Wird mehr	Maintainability		
TD				3.00	2.00	12.05	Wird mehr	Maintainability		
TD				3.00	2.00	12.05	Wird	Security		

Figure 8: Example of a visualization of technical debt issues for planning purposes (anonymized)

The list is controlled via a graphical visualization, whereby the following three visualizations (sorted by importance) have become established for this purpose:

1. Low-hanging fruits (Figure 8): In a coordinate system with “effort” on the x-axis and “priority” on the y-axis, points indicate technical debt issues. The size of the points symbolizes the number of issues at that point. The list displays the technical debt associated with the point clicked by the user.
2. Components (Figure 8): Issues are displayed per component in a bar chart or pie chart. The list displays the technical debt associated with the component clicked by the user.

3. Quality attributes: Issues are displayed per quality attribute in a bar chart or pie chart. The list displays the technical debt associated with the quality attribute clicked by the user.

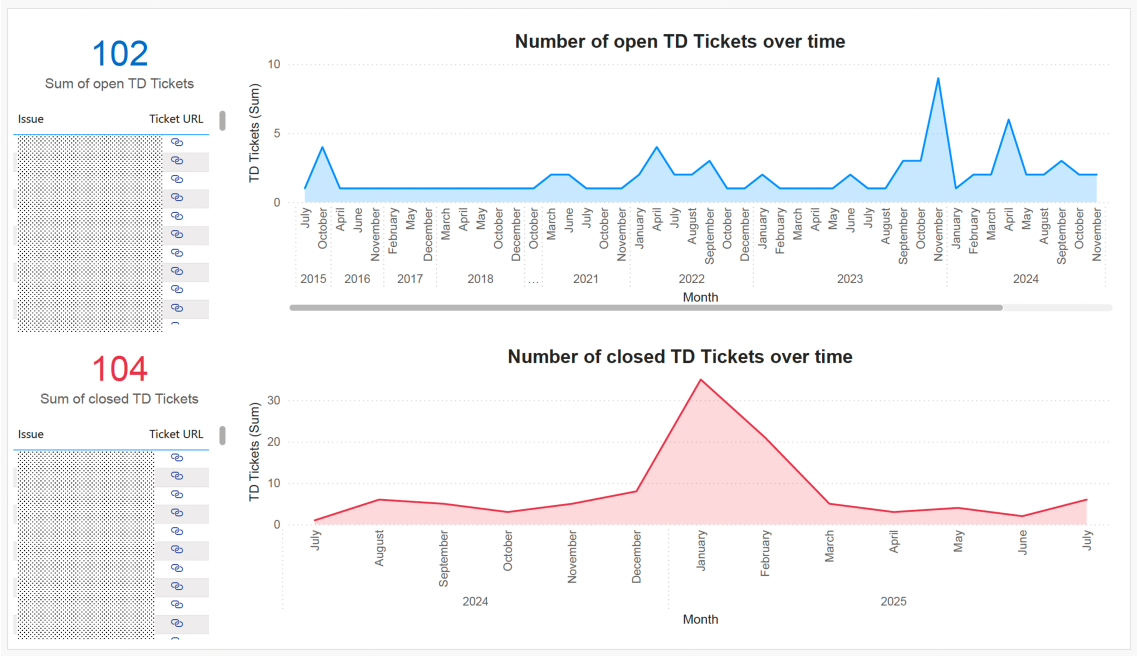


Figure 9: Example of a visualization of technical debt issues for management and monitoring purposes (anonymized)

The visualizations used to raise awareness of technical debt also serve to monitor the management process and are primarily based on time series:

- Number of open technical debt issues per month (Figure 9)
- Number of closed technical debt issues per month (Figure 9)
- Number of open technical debt issues per month
- Interest burden (see Forumla 1) of all open technical debt issues per month (Figure 10)

Technical obstacles Most issue tracking systems offer insufficiently flexible visualization options. It is therefore advisable, as already mentioned in Chapter 3.1.2, to connect the issue tracker to a reporting tool such as MS Power BI or Tableau. This creates development and maintenance problems. However, a bigger problem is that user acceptance of the visualization decreases when a separate and previously unknown tool has to be used.

In addition, it is particularly challenging to determine the number of open technical debt issues per month. Issue tracking systems store the "open" and "closed" dates. To determine the number of open issues for each month, a code snippet must be used to iterate through all months in a loop and then check for each month and each issue whether it was open. A counter is incremented for the number of open issues. For the interest burden of open issues, the interest burden must be determined and totaled for each open ticket. The code snippet to be developed makes development and maintenance even more difficult. A visualization of the interest burden

without this calculation is misleading, as the interest burden always appears to be increasing because issues that have already been closed are no longer factored into the calculation.

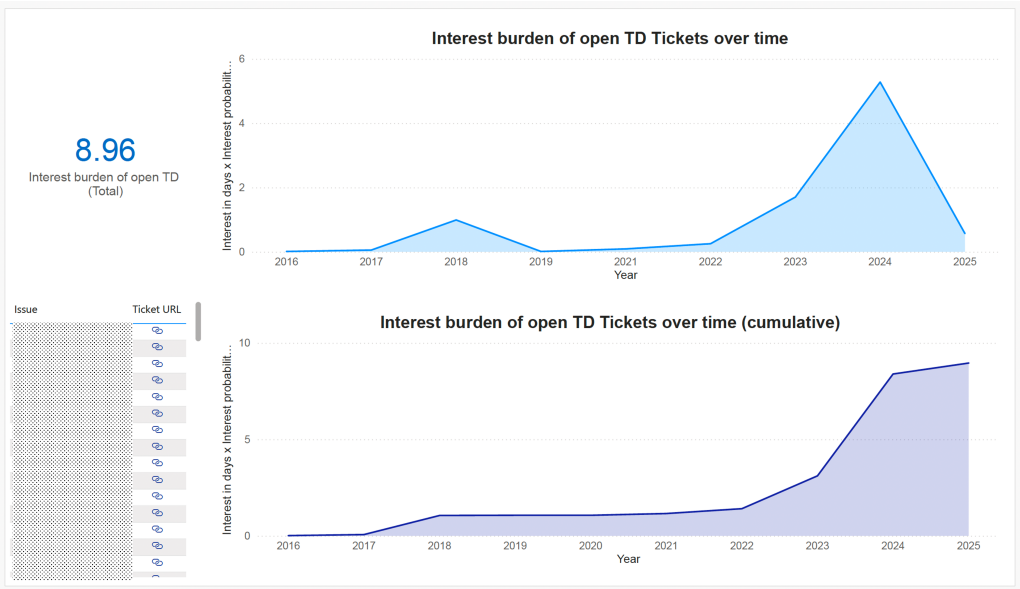


Figure 10: Example of a MISLEADING visualization of the interest burden, but only for currently open tickets and their “open” date, and not as a time series (anonymized)

3.5 Getting Started

To start the process, a few steps are necessary, which are listed here in chronological order:

1. The "technical debt manager" (TS manager; see Chapter 3.1) is appointed in a joint meeting. From then on, they control the further steps in accordance with these instructions.
2. A keyword name for technical debt (e.g., "TD") and issues that have already been searched but not identified as technical debt (e.g., "Non-TD") is defined.
3. All existing issues are reviewed, and it is determined whether or not they constitute technical debt. The issues are tagged accordingly ("TS" / "Non-TS"). See also Chapter 4.2. ("Non-TS" is useful for performing this analysis in several smaller blocks so that you know which issues have already been considered.)
4. All team members record all technical debt they have in mind but have not yet documented, and tag it as technical debt.
5. Discuss within the team which of the attributes from Table 7–8 are useful for this team/company.
6. Create a new issue type for technical debt with the selected attributes. Examples can be found in the appendix Figure 11–13
7. Transfer all issues tagged as technical debt to the new issue type
8. Evaluate all technical debt issues. There are two options for this:
    - Specific meetings are scheduled for all team members to refine technical debt issues. These issues are discussed and evaluated one by one, meaning the attributes are filled in. This approach is especially recommended at the beginning of the evaluations so that the team can develop a shared understanding of how to evaluate the attributes.
    - Each team member takes on 3-4 technical debt issues per week and evaluates them individually "in advance." In particular, the team member sets the resubmission date and can thus determine how urgently the issue needs to be considered by all team members. If the team member is unsure about their assessment, they can, for example, choose a short-term resubmission date. If the issue is obviously less relevant, the resubmission date can be set for a far future. During refinement, when all issues with expired resubmission dates are reviewed, the issue is then evaluated jointly by all team members. This distributes the burden of evaluation over time, eliminating the need for separate meetings.
9. The team decides (in a separate meeting, if necessary) which repayment procedure should be used (in which situations).
10. The TS manager ensures that the repayment agreements are adhered to.
11. The TS manager clarifies which visualization tools can be used, links the issue tracker to the visualization tool, and creates initial visualizations.
12. The visualizations are used by the team and adjusted by the TS manager as needed.

13. In all other issue types, the attributes for avoiding technical debt are added in accordance with Table 5–6. This can be done as an attribute field or as part of a DoR.

14. For all issues (including technical debt issues!) that are now being handled in refinement, the attributes are evaluated.

4 Typical Challenges in Managing Technical Debt

During the introduction of technical debt management, typical mistakes (Chapter 4.1) and questions (Chapter 4.2) arise time and again. The following two chapters provide an overview of these topics and suggest solutions. In particular, the typical mistakes from Chapter 4.1 should be taken into account in order to successfully introduce the process.

4.1 Typical Mistakes

There are mistakes that teams repeatedly make during implementation. These "mistakes" are usually caused by the fact that the issue of technical debt is overlooked in the hectic day-to-day business. The role of the TS manager helps to keep the topic in mind during day-to-day business. However, mistakes can still occur, despite having a TS manager, and the following list can be passed on to future TS managers as a learning experience.

4.1.1 We all remember it.

There are always situations in which a task in the process is overlooked, for example, discussing drawbacks and risks to avoid technical debt (see below). This is often because there is no clear responsibility for this task. The team decides to remember it ("we'll all do it"), but in the end, no one takes responsibility for the task, and it is forgotten. This phenomenon particularly affects the management of technical debt, as in this case, by definition, these are issues that are not (immediately visible) business-critical.

Solution: You need someone who will repeatedly remind you of the chosen process and the resulting tasks (e.g., discussing drawbacks and risks). In the initial phase of establishing the process, it has proven useful for this person not to be part of the team. This allows team members time to become familiar with the process and integrate it into their daily routines. In the long term, the TS manager should then take over the task. Other people can also take on tasks, but even then, there should be clear accountability.

4.1.2 The attributes all make sense – we use them all.

All attributes listed in the Table 5–8 initially appear useful and important, which is why they are listed here. However, this creates a significant risk of using all attributes in your own template, thereby making the process (unnecessarily) complicated.

Solution: Only the "best practices" should be used for the template initially. A maximum of one or two attributes from the "nice-to-haves" should be used at the beginning, but preferably none at all. During daily work, i.e., the application of the process, it will then become apparent whether additional attributes are helpful and should be added.

4.1.3 Forgetting the new attributes for non-technical debt issues

When introducing the new process, the focus is typically on managing existing technical debt. Avoiding new technical debt is often seen as secondary. As a result, people often forget to fill in the "attributes for all issue types" (see Chapter 3.2, Table 5–6).

Solution: It is advisable to appoint someone from the team to closely monitor this. This could also be the TS manager. And no, "all of us" will not pay attention to this (see above). Responsibility must be assigned to ensure that the issue is successfully addressed.

4.1.4 Alternatives? There aren't any.

In everyday working life, it's easy to overlook the attribute "alternative." The motto is: briefly mentioned and then forgotten again. "Alternatives? Nope, I can't think of any. So there aren't any—check." In many cases, this is the pragmatic approach; however, you must be careful that the approach doesn't take on a life of its own.

Solution: How to find alternatives is described in detail in Chapter 4.2.

4.1.5 Unconsciously accumulating technical debt: Planning Poker is just annoying

In many teams, planning poker is no longer used or is used only half-heartedly. The effort seems too small for the benefit, and "we all know what the right number is anyway." In some teams, it even goes so far that the product owner makes the estimates or at least gives the first estimate.

The problem with this is the anchoring effect, i.e., the tendency to prefer what is said or heard first. This is a cognitive bias, i.e., a mistake our brain makes for efficiency reasons, and one that is difficult for us to escape. Once a value has been put forward (e.g., 2 SP), it is difficult to name a completely different value (e.g., 8 SP). This leads to misestimations and thus to time pressure. However, time pressure is one of the main causes of conscious and unconscious technical debt. In particular, if the developer feels a moral responsibility because they accepted the estimate, they are more likely to want to stick to the estimate – at the expense of quality, if necessary.

Solution: The best thing to do is actually to resume Planning Poker. A lightweight variant is to at least provide a clear justification for each estimate. This allows another person to refer to the reasoning if they want to suggest a completely different value, e.g., "You only estimate 2 SP, but you forgot that there is also the dependency on the <xy> system, so I see the effort as significantly higher."

Under no circumstances should the product owner give an estimate. Estimates are always and exclusively made by the experts concerned.

4.1.6 Unconsciously taking on technical debt: But that has to be included in the sprint now.

Unfortunately, there are often situations where issues must be included in the sprint because they have a short deadline or have already been "promised" to the customer. However, if uncertainties still surround these issues, e.g., feedback from tool providers or other teams regarding the details of implementation, the issue may not be implemented optimally because important information is missing. Out of a misguided desire to accommodate the customer/requester, a poorer solution is implemented, thereby unknowingly incurring technical debt.

Solution: An issue may only be included in the sprint once all information is available. Here, the product owner has a duty to communicate openly with the customer/requester. Often, the customer/requester is not even aware that a less-than-optimal solution is being implemented because communication with the customer/requester is avoided. It should be made clear to the customer/requester that the deadline can only be met at the expense of quality and what specific consequences the lack of quality can have in the long term (see Chapter 2.3).

A similar communication problem sometimes arises between developers and product owners. Developers should also make it clear to the product owner that the deadline can only be met at the expense of quality and what specific consequences can be expected.

4.1.7 Consequence of technical debt: Poor maintainability

The attribute "risks of non-repayment," i.e., the consequences of technical debt, invites us to write something like "poor maintainability," which has indeed often happened in practice. However, this is already the definition of technical debt, or at least part of it. The consequences should be described in more detail and understood by non-technical stakeholders.

Solution: There is a method called the "5 Whys" [11], which involves identifying the root cause of a problem by asking multiple (five) questions about the cause ("Why") in succession. This method can be adapted for this purpose by repeatedly asking why the stated issue is a problem, e.g.: (1) Why is it a problem if maintainability is poor here? – Because changes take longer and are risky. (2) Why is it a problem if changes take longer and are risky? – Because we change this component frequently. (3) Why is it a problem if we change this component frequently? – Because we will no longer be able to meet deadlines in the future, and the risk of introducing bugs will increase. (4) Why is it a problem if we don't meet schedules and introduce bugs? – Because customers will be dissatisfied. (5) Why is it a problem if customers are dissatisfied? – The reputation of the system/company will decline. The point here is not to record all the details, but to extract the essential information and write that down. In the example, that would be: "Changes to this central component are frequent. These changes will take longer in the future and carry an increased risk of side effects, such as bugs or performance losses, which could upset our customers."

4.1.8 The components "Test," "Documentation," and "Logging"

Some teams find it difficult to list the (architectural) components. In particular, so-called "cross-cutting concerns" (i.e., tasks that span multiple components), such as testing, documentation, logging, monitoring, feature flag removal, and even refactoring, are often recorded as components. Component assignment is important in technical debt management, especially for evolution-based repayment. The goal is to identify the technical debt items contained in the component that is currently being modified to meet a functional requirement. However, if a cross-cutting concern, such as "documentation," has been selected as a component, this does not provide such information and is therefore meaningless. It is not clear which component is inadequately documented. It is unclear whether there might be faulty documentation of the component to be changed, and whether the documentation should be optimized along with the change in order to exploit synergies.

Solution: When recording the components, a counter-test should always be performed to determine whether it is a "cross-cutting concern." To do this, consider whether there could be a functional change associated with this component.

4.1.9 Nothing is 'very'

If a 5-point scale from "very low" to "very high" is chosen for the values in a combo box, there is a risk of "central tendency bias," i.e., the unconscious tendency of the human brain to choose a middle value. Extreme values, on the other hand, are avoided. As a result, a theoretical 5-point scale quickly becomes a practical 3-point scale.

Solution: In addition to the values from "very low" to "very high," there should be a clarification of what is meant by each value. We have already shown this using the example of interest rates and interest rate probabilities in Table 7. There, specific values such as "$<$ 15 min." or "$>$

1x/year" are also given, which gives developers a concrete idea of the values and allows them to use "very low" and "very high" normally.

4.1.10 Resubmission: As a precaution, next week

A common mistake is to set the resubmission date too soon.

Solution: When setting the resubmission date, it is not about when you want the issue to be resolved, as this is determined by the priority. It involves waiting for specific events (e.g., management decisions, developers returning from vacation) or observing whether general conditions have changed unexpectedly in other ways (e.g., changes in management, new customers, new technologies on the market). Therefore, it makes no sense to assign a resubmission date in a few months if no specific decision is to be awaited. In these cases, the date should be at least 6 months (preferably more) in the future, as an essential change in the general conditions is unlikely during this time.

4.2 Typical Questions

Some frequently asked questions in practice will be addressed here. It makes sense to read the specific answers to questions when they arise.

4.2.1 Too many issues in the backlog—how do I get started?

At the beginning of the process, all issues in the backlog must be searched, and technical debt must be marked. This can be a tedious task for large backlogs with long-standing issues.

Solution: An initial assessment is best made intuitively, based only on the title and the indicators mentioned in Chapter 3.3.2. The label for technical debt can be assigned generously, i.e., better once too often than once too little. Since all issues are discussed again in refinement, it is still possible to decide that an issue does not describe technical debt after all. In cases of very large backlogs with very old issues, you can also start by focusing on issues created within the last year (or the last two years). Older issues may still contain further technical debt, but these are usually either less problematic or they reappear and can then be recorded as new issues. In this case, it is more important to start the process than to have all issues fully recorded.

4.2.2 How can I find alternatives?

Some teams struggle to identify alternatives to the solution they initially proposed. Cognitive biases unconsciously lead them to consider only one solution.

Solution: Teams often start by gathering all the alternatives and then comparing their pros and cons. However, it has turned out that using the discussion order shown in Table 6 makes sense. First, the quality attributes affected by the initially proposed solution should be discussed. Based on this, the drawbacks and risks can then be derived (e.g., if performance is negatively affected). Starting from the drawbacks and risks, it is then easier to identify an alternative solution that may not entail these drawbacks/risks.

4.2.3 Can I use other terms?

The terms used here for technical debt are based on the metaphor of financial debt. This metaphor was originally developed to make the topic of "technical debt" more understandable to business

stakeholders [5]. Some of the terms are based on other scientific publications, such as the concept of contagion [12]. These terms are sometimes difficult for development teams to understand or comprehend. The terms have also not become as established among business stakeholders as originally hoped [13, 1].

Solution: It is therefore perfectly legitimate to use other terms as long as their meaning is clear to everyone involved. For example, some teams have adopted the term ”interest frequency” instead of ”interest probability” because it makes the assignment of combo box values easier to understand.

4.2.4 There are several consequences. How do I assess this?

As already mentioned in Table 7, there can be several consequences of technical debt, all of which should be recorded in the attribute “Risks of non-repayment.” However, there are also several different “interest charges,” but only one attribute ‘Interest’ and one attribute “Interest probability.” Theoretically, the interest rates for all consequences would have to be estimated with their respective interest rate probabilities and then summed, which is neither cognitively nor computationally feasible.

Solution: A pragmatic approach that has proven successful is to evaluate the most serious consequences. The team could also decide to set interest and interest probability higher than appropriate for the most serious consequence in order to account for further consequences. However, this approach also requires increased cognitive effort and should only be used if the team has already gained some experience with the process.

4.2.5 How do I handle ongoing or recurring maintenance tasks such as updates?

Some maintenance tasks need to be performed on an ongoing basis, such as updating 3rd-party libraries or tools used. In these cases, it would be too time-consuming to create an issue for every version change; instead, some teams maintain a simple list of tools with their versions and do not want to deviate from this approach.

Solution: In this case, you can create a single ’update’ issue and describe and evaluate the most urgent update within it. As soon as the priority of this most important update is higher than that of all other technical debt issues, the issue has to be implemented, i.e., the update has been carried out. After that, an issue is created for the next most important update.

Of course, the team only makes a rough estimate of which update is most important, rather than evaluating the priority as precisely as in the technical debt issues. This inaccuracy is a pragmatic compromise. If there are disagreements about which update is the most urgent, two or more update issues can be created and evaluated.

5 Outlook and Scaling of Technical Debt Management

The introduction of technical debt management described above serves individual teams. In a corporate environment, it makes sense to consider whether and how this approach can be scaled, i.e., applied to many or all teams within a company.

Using the process described here across all teams within a company offers many advantages. There are benefits for each individual team in their daily work:

- Overview of technical debt within the team
- Assessment of risks arising from technical debt
- Prioritization of various technical debt issues
- Clearly defined and consistent repayment of technical debt
- Ability to provide information and arguments to management

There are further advantages for management when the process is used by several or all teams and the resulting data is reported in summary form:

- Overview of technical debt per team and per component, enabling the identification of "hot spots."
- Assessment of the risks arising from technical debt in cross-team projects
- Prioritization of support for teams, e.g., relieving the team/component of customer requests in order to have time to repay technical debt or increase the team's staffing levels
- Clearly defined and therefore calculable procedure for repaying technical debt

For scaling, a team within the company should first be given the time to thoroughly address the topic and establish a process based on these guidelines. The result will be an issue type and a corresponding visualization. Both can then be used by other teams with significantly less initial effort. Nevertheless, these other teams must also be given time for the initial discussion and evaluation of technical debt issues.

According to initial estimates, the effort required for the first team is 5-6
In our view, there are two possible ways to encourage other teams to use this system:

1. Dissemination of the process "from the top down," e.g., via management or architecture committees, i.e., the team architects.
2. Dissemination of the process "from within the company" through a community of practice (CoP) on the topic of technical debt. This approach can have the advantage that the intrinsic motivation for change is higher and the process is taken more seriously and "lived" by the teams using it.

Which path is chosen depends heavily on the corporate culture. However, it is always possible to start with the CoP approach and, if this proves unsuccessful, to try the first approach via management or architects.

Thanks

We would like to thank our industry partners and their team members for their participation, trust, and valuable insights shared with us. In memory of Prof. André van Hoorn and Prof. Matthias Riebisch.

Funding and Declaration

The project on which this white paper is based was funded by the German Federal Ministry of Education and Research under grant number 01IS24031. The authors are responsible for the content of this publication. The opinions expressed herein are my own and do not necessarily represent those of my employer.

Appendix

At the end of the document, a collection of screenshots of issue types for technical debt is provided.

1. Azure DevOps in Figure 11
2. Jira in Figure 12
3. GitLab in Figure 13

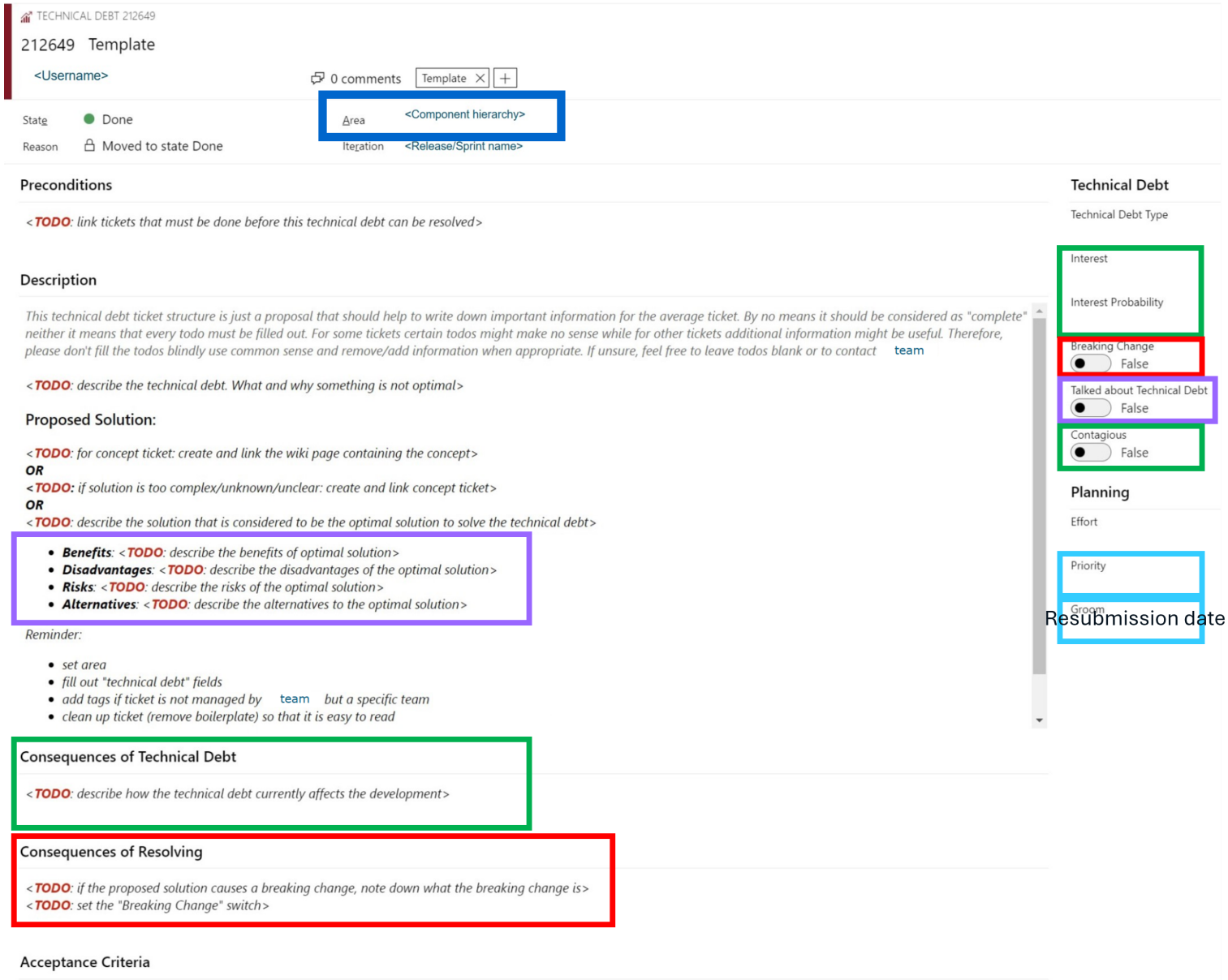


Figure 11: Example of an issue type for technical debt in Azure DevOps. (Purple: Attributes for avoiding technical debt. Green: Attributes for assessing the consequences of technical debt. Red: Attributes for assessing the risks of expanding technical debt. Light blue: Priority and resubmission date for managing technical debt. Dark blue: Component hierarchy for risk assessment of functional changes and dismantling with synergies)

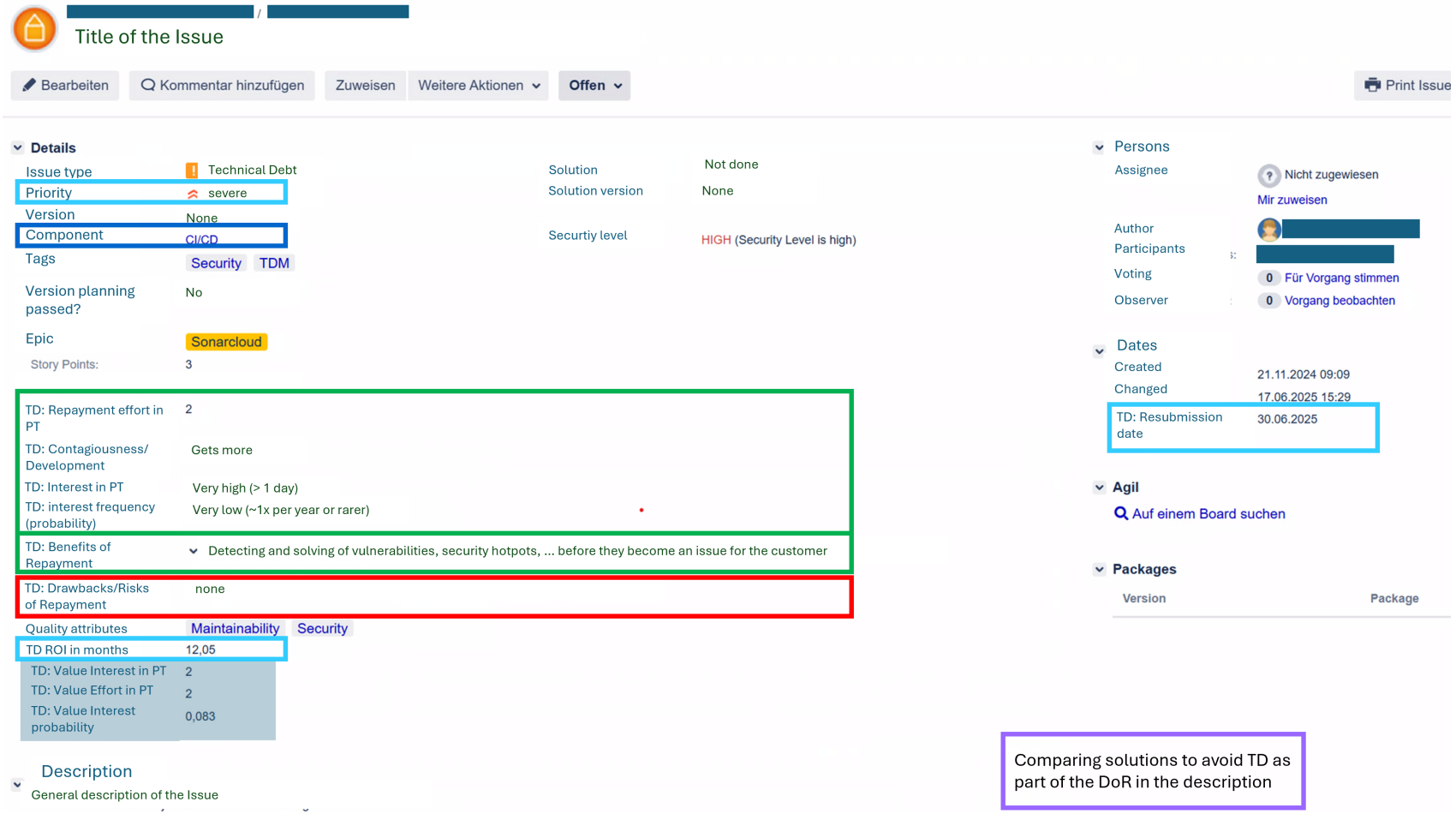


Figure 12: Example of an issue type for technical debt in Jira. (Purple: Attributes for avoiding technical debt. Green: Attributes for assessing the consequences of technical debt. Red: Attributes for assessing the risks of expanding technical debt. Light blue: Priority and resubmission date for managing technical debt. Dark blue: Component hierarchy for risk assessment of functional changes and dismantling with synergies)

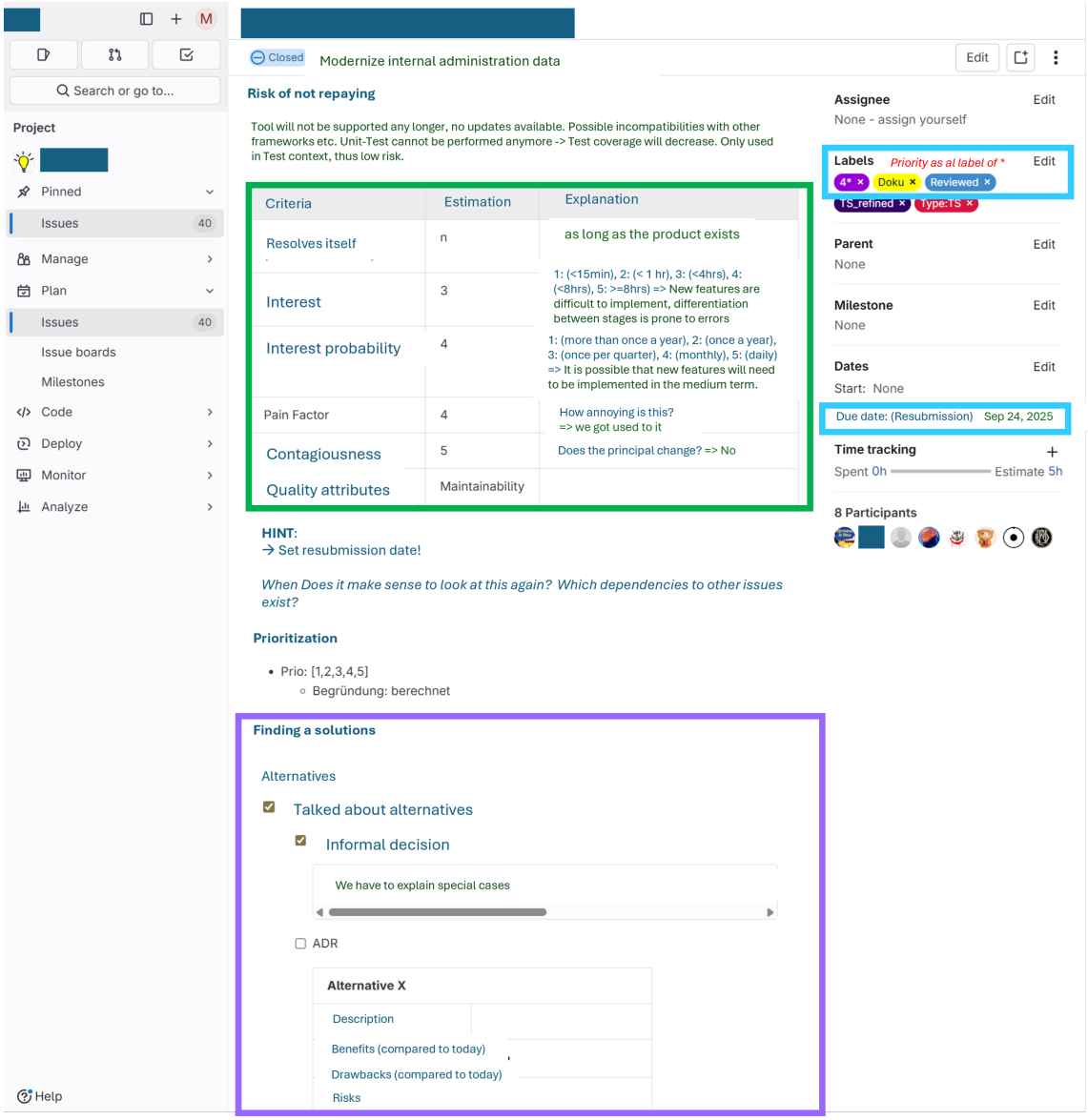


Figure 13: Example of an issue type for technical debt in GitHub (resolved as a template). (Purple: attributes for avoiding technical debt. Green: attributes for assessing the consequences of technical debt. Light blue: priority and resubmission date for managing technical debt.)

Acknowledgment

We thank the teams that participated in the study for their participation, trust, and insights. The project on which this report is based was sponsored by the Federal Ministry of Research, Technology, and Space under the funding code 01IS24031. Responsibility for the content of this publication lies with the authors.

References

[1] M. Wiese, K. Borowa, IT managers' perspective on Technical Debt Management, Journal of Systems and Software 202 (2023) 111700. doi:10.1016/j.jss.2023.111700.

[2] M. Wiese, A. Heinrichs, N. Rusieshvili, R. R. de Almeida, K. Borowa, The TechDebt Game - Enabling Discussions about Technical Debt (2025).
URL https://arxiv.org/abs/2502.02174

[3] M. Wiese, K. Serwa, B. Anastasia, E. Bittner, Establishing technical debt management – a five-step workshop approach and an action research study, Journal of Systems and Software 231 (2026) 111700. doi:/10.1016/j.jss.2025.112606.

[4] M. Wiese, K. Serwa, E. Bittner, A practical guide for establishing a technical debt management process, in: 2026 IEEE/ACM International Conference on Technical Debt (TechDebt), 2026. doi:10.1109/TechDebt.

[5] W. Cunningham, The WyCash portfolio management system, in: Proceedings of the Conference on Object-Oriented Programming Systems, Languages, and Applications, OOPSLA, Vol. Part F1296, 1992, pp. 29–30. doi:10.1145/157710.157715.

[6] P. Avgeriou, P. Kruchten, I. Ozkaya, C. Seaman, Managing Technical Debt in Software Engineering, Dagstuhl Reports 6 (4) (2016) 110–138. doi:10.4230/DagRep.6.4.110.

[7] P. Kruchten, R. Nord, I. Ozkaya, Managing Technical Debt: Reducing Friction in Software Development, Software Engineering Institute, Carnegie Mellon University, 2019.

[8] Z. Li, P. Avgeriou, P. Liang, A systematic mapping study on technical debt and its management, Journal of Systems and Software 101 (2015) 193–220. doi:10.1016/j.jss.2014.12.027.

[9] M. Fowler, Technical Debt Quadrant (2009).
URL https://martinfowler.com/bliki/TechnicalDebtQuadrant.html

[10] ISO/IEC, ISO 25010-2023 | Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models (2023).
URL https://www.iso.org/obp/ui/en/#iso:std:iso-iec:25010:ed-2:v1:en

[11] O. Serrat, The Five Whys Technique, in: O. Serrat (Ed.), Knowledge Solutions: Tools, Methods, and Approaches to Drive Organizational Performance, Springer, Singapore, 2017, pp.

307–310. doi:10.1007/978-981-10-0983-9_32.
URL https://doi.org/10.1007/978-981-10-0983-9_32

[12] A. Martini, J. Bosch, The Danger of Architectural Technical Debt: Contagious Debt and Vicious Circles, in: Proceedings - 12th Working IEEE/IFIP Conference on Software Architecture, WICSA 2015, 2015, pp. 1–10. doi:10.1109/WICSA.2015.31.

[13] K. Schmid, A formal approach to technical debt decision making, QoSA 2013 - Proceedings of the 9th International ACM Sigsoft Conference on the Quality of Software Architectures (2013) 153–162ISBN: 9781450321266. doi:10.1145/2465478.2465492.

Appendix B

Additional Material of Contributing Publications

1 *(TAP Framework)* **Preventing Technical Debt by Technical Debt Aware Project Management**	
Citation	M. Wiese, M. Riebisch and J. Schwarze, *Preventing Technical Debt by Technical Debt Aware Project Management*, 2021 IEEE/ACM International Conference on Technical Debt (TechDebt), 2021, Madrid, Spain, pp. 84-93, https://doi.org/10.1109/TechDebt52882.2021.00018
Link	https://doi.org/10.5281/zenodo.4616485
Content	This zip file contains eight files to replicate and verify the study evaluating the Framework for Technical Debt Aware Project Management described in the paper *Preventing Technical Debt by Technical Debt Aware Project Management*, Wiese et al., 2021, TechDebt21 • TDframework_ticket_rawdata_revised_anonymised.xlsx: Revised and anonymised raw data for ticket statistics, containing two sheets for TD ticket and maintenance ticket • TDframework_survey_questions_and_SPSSvariable_definition.xlsx: Excel sheet with all survey questions in German and English, SPSS variable names, and shortnames • TDframework_survey_results.sav: Survey results in SPSS format; variables with the suffix "D" are dichotomized versions of the variables as explained in the paper. • TDframework_survey_results.xslx: Excel-Export of TDframework_survey_results.sav • MannWhitneyUTest.spv: Results of the Mann-Whitney U-Test, which show the significance of the hypotheses mentioned in the paper (Section 5.2.2) • MannWhitneyUTest.xslx: Excel-Export of MannWhitneyUTest.sav • Correlations.spv Results of the Chi-square test for correlations and the significances mentioned in the paper (Section 5.2.3) • Correlations.xslx Excel-Export of Correlations.sav

TABLE B.1: Additional Material to Publication No. 1 *(TAP Framework)*

2 *(TAP Framework – extended)* **Preventing technical debt with the TAP framework for Technical Debt Aware Management**	
Citation	M. Wiese, P. Rachow, M. Riebisch, J. Schwarze, *Preventing technical debt with the TAP framework for Technical Debt Aware Management*, Information and Software Technology, Volume 148, 2022, 106926, ISSN 0950-5849, https://doi.org/10.1016/j.infsof.2022.106926
Link	https://doi.org/10.5281/zenodo.5788222
Content	This zip file contains twelve files and one folder to replicate and verify the study evaluating the TAP Framework (for Technical Debt Aware Project Management). • TDframework_ticket_rawdata_revised_anonymised.xlsx: Revised and anonymized raw data for ticket statistics containing two sheets for TD ticket and maintenance ticket • TDframework_survey_questions_and_SPSSvariable_definition.xlsx: Excel sheet with all survey questions in German and English, SPSS variable names, and short names • TDframework_survey_results.sav: Survey results in SPSS format; variables with the suffix "D" are dichotomized versions of the variables as explained in the paper. • TDframework_survey_results.xslx: Excel-Export of TDframework_survey_results.sav • MannWhitneyUTest.spv: Results of the Mann-Whitney U-Test, which show the significance of the hypothesis mentioned in the paper (Section 5.2.2) • MannWhitneyUTest.xslx: Excel-Export of MannWhitneyUTest.sav • Correlations.spv Results of the Chi-square test for correlations and the significances mentioned in the paper (Section 5.2.3) • Correlations.xslx: Excel-Export of Correlations.sav • Management-Survey_2021_en.pdf: Original survey with English translation of the questions • Management-Survey_2021_XXX.pdf: Results of the management survey from – team manager of the observed unit (TMO), participant A – team manager of the comparison unit (TMC), participant B – unit manager of the observed unit (UMO), participant C – unit manager of the comparison unit (UMC), participant C UMC => no survey, but email conversation between the author and UMC to get the answers for the comparison unit • Fig6_Cause-Effect-Diagram_arrow_details.xslx Details about the sources for the arrows in this figure • Folder "Case Study Protocol": Contains the case study protocol according to Runeson's guidelines.

TABLE B.2: Additional Material to Publication No. 2 *(TAP Framework – extended)*

3 *(IT Managers' Perspective)* **IT managers' perspective on Technical Debt Management**	
Citation	Marion Wiese, Klara Borowa, *IT managers' perspective on Technical Debt Management*, Journal of Systems and Software, Volume 202, 2023, 111700, ISSN 0164-1212, https://doi.org/10.1016/j.jss.2023.111700
Link	http://dx.doi.org/10.5281/zenodo.7803013
Content	This is the additional material for the Paper *IT Managers' Perspective on Technical Debt Management* published in the Journal of Systems and Software. The folder contains the following files: • Consent form - blinded.pdf / Consent form.pdf: The consent form was used to gather the participants' consent to participate in this study. • PreliminarySurvey.pdf: The survey was sent to all participants in advance to gather basic statistical data. • ParticipantsOverview.pdf / ParticipantsOverview.xslx: The overview of all participants was extracted from the preliminary survey. For anonymity reasons, we can not provide the country/language of the participants. • InterviewGuide.pdf: The interview guide was followed by the interviewers, including all questions and the rationale for questions added before the 2nd iteration. • CodeSystem.xlsx: All Codes were gathered from the interviews with a short explanation (in Excel to use the grouping function for a better overview). • AnalyticMemos_Themes.pdf: All free memos were created during the study, including all themes that led to the main concerns described in the paper. • Focusgroup-discussion_Themes_slides.pdf: The slides were created for the focus group discussion. The participants were introduced to the "main concerns" and asked to comment on these findings. • V4C_model.drawio: This is the original draw.io file of our V4C model, including all diagrams from the paper. Additionally, we provide the complete version of the visibility diagram, which had to be reduced in the paper due to the page limitation. Also, the file contains one more diagram with another example of the model's usage. • V4Cmodel.pdf, V4Cmodel_example.pdf, VisibilityOfCausesConsequences.pdf; VisibilityCycles.pdf: All diagrams of the paper are in a single PDF file. • V4CTDmodel+example.pptx: A short presentation of the model and an example of its usage in PowerPoint format to immediately use.

TABLE B.3: Additional Material to Publication No. 3 *(IT Managers' Perspective)*

4 *(Learning from each other)* **Learning from Each Other: How Are Architectural Mistakes Communicated in Industry?**	
Citation	M. Wiese, A.F. Brand, A. van Hoorn, *Learning from Each Other: How Are Architectural Mistakes Communicated in Industry?*, 17th European Conference on Software Architecture (ECSA), 2023, Istanbul, Turkey, Lecture Notes in Computer Science, vol 14212. Springer, Cham. https://doi.org/10.1007/978-3-031-42592-9_22
Link	https://doi.org/10.5281/zenodo.7924700
Content	The additional material for the paper *Learning from each other: How architectural mistakes are communicated* comprises the following documents: • interview_guide_V<1|2|3>.pdf The three versions of the questions that the interviewees were asked, and the remarks for the interviewer • study_information_and_consent_form.pdf General information about the study for the participants and a consent form that the participants had to sign • codesystem.xslx List of all codes and the frequency of their occurrences

TABLE B.4: Additional Material to Publication No. 4 *(Learning from each other)*

5 *(TD Game)*	
The TechDebt Game - Enabling Discussions about Technical Debt	
Citation	M. Wiese, A. Heinrichs, N. Rusieshvili, R. R. d. Almeida and K. Borowa, *The TechDebt Game - Enabling Discussions about Technical Debt*, 2025 IEEE/ACM International Conference on Technical Debt (TechDebt), 2025, Ottawa, ON, Canada, pp. 23-33, https://doi.org/10.1109/TechDebt66644.2025.00010
Link	https://doi.org/10.5281/zenodo.14205655
Content	This is an overview of the additional material for the paper *Technical Debt Game - Enabling Discussions about Technical Debt*, published at the TechDebt 2025 conference. • ENGameManualDigital.pdf Instructions on how to play the TechDebt Game, including details and explanations on how the game relates to the reality of dealing with technical debt. • ENShortInstruction.pdf Instructions on how to play the TechDebt Game, shortened to the minimum, without details/examples, and without explanation on how the game relates to the reality of dealing with technical debt. • ENPrintversion.pdf Gameboard, cards, scoreboard, etc., to print out and play the game offline. The Game must be augmented with two dice, two game stones, and multiple tiles in two different colors. Instead of tiles, one can also use cut-out paper tiles or taller game stones. • Introduction guide for game master.pdf When explaining the game to participants, the game masters, i.e., researchers, followed this guideline. • Survey Printversion - TD Game.pdf Print-version of the online survey each player filled out after the game. • Survey_analysis.xlsx Analysis of the survey results, including tables for: – AllResults: overview of all results – questions: list of all questions – AHA moments: explanations for each AHA-moment – sample 1 and 2: analysis of demographic results – comments game: analysis of the open question regarding feedback on the game – comments attitude + behavior: analysis of the open question regarding attitude and behavior changes – Data AHAmoments: Results of the AHA-moments per player – Data all Players (Vis.): summarized data and visualization for all players – Data per category: AHA-moment's results ordered by stakeholder category – Reality&Insights p. cat. (Vis.): sums for new real-life connections ("had this AHA moment, but is not new") and new insights ("had this AHA moment and is new") per player for each stakeholder category – Data $< B\|TJ\|TS >$-Player (Vis.): summarized AHA-moments and visualization for each stakeholder category
Online Game EN	https://tabletopia.com/workshop/games/techdebts/1-7players/test
Online Game DE	https://tabletopia.com/workshop/games/techdebts/1-6players-1/test
Bord Game	https://github.com/TechDebtGame/TechDebtGame

TABLE B.5: Additional Material to Publication No. 5 *(TD Game)*

	6 *(Establishing TDM)* **Establishing technical debt management — A five-step workshop approach and an action research study**
Citation	M. Wiese, K. Serwa, A. Besier, A S. Marion-Jetten, E. Bittner, *Establishing technical debt management — A five-step workshop approach and an action research study*, Journal of Systems and Software, Volume 231, 2026, 112606, ISSN 0164-1212, https://doi.org/10.1016/j.jss.2025.112606
Link	https://doi.org/10.5281/zenodo.14916678
Content	The additional material for the paper *Establishing Technical Debt Management - A Five-Step Workshop Approach and an Action Research Study* comprises the following documents: • Guideline_to_Use_TD-SAGAT.docx (.pdf): A short guideline on how to use the TD-SAGAT method. • SAGAT_for_TD_from_Goals_to_Queries.pdf: Determination of survey questions by following the SAGAT approach of following the hierarchy of goals, subgoals, decisions, requirements, and queries • WIKI page on technical debt.pdf: The team's wiki page on TD management, written by the TD manager • AzureTDTickets_Template.pdf Screenshot of the backlog's TD item template as presented in the paper for better readability. • OverviewOfActionCycles_ExtendedTable5.xlsx: Table 5 (Overview of action cycles with workshop topics, TD activities, actions taken, and learning) as an Excel sheet with additional information on actions and learning, which we did not put into the paper • Workshops/workshop_slides/TDMGuide_< x >_Workshop.pptx: Slides for each of the six workshops • Workshops/workshop_slides/TDMGuide_0_InitTalk_35Min_EN.pptx: The initial one-hour introduction into the TD topic (40 min. talk + 20 min. for a short discussion) • Workshops/workshop_flipcharts/WS<No.>_<explanation>.jpg: Six anonymized flipchart photos to give an impression of the workshops, particularly of the kick-off workshop (other workshops made less use of flipcharts). Translations are directly included in the photos, except for ... • Workshops/workshop_flipcharts/WS1_Topic Cache - Translation.txt: Translation of the German topic cache in an extra file, as there were too many topics to translate in the photo • Workshops/Workshop Agendas.pdf: Agendas for the kick-off workshop and a general agenda for further workshops, each with a detailed agenda for workshop moderators and a rough agenda for participants. • Workshops/ TD_Energy_Matrix.xlsx: Matrix where participants mark their own, the teams', and the companies' (assumed) level of willingness to invest in a TDM versus their (assumed) evaluation of TDM's importance. Used in the kickoff workshop • Evaluations/Backlog_Evaluations_and_Figures.xlsx: – Backlog data from the 5th workshop (for 6th workshop, we used the PowerBI Visualization instead) – Evaluation of this data comprising – priority calculations based on priority risk and – priority calculations based on priority risk based on ROI • Evaluations/Meetings_Overview_Observations_Evaluation_Figures.xlsx: – Overview of all meetings, plus information on length, participants, whether a video exists, and whether it was used for evaluation – Filled out observation protocols from different meetings – Summation, evaluations, and resulting figures • Evaluations/TD-SAGAT_Survey_and_SurveyResults_Figures.xlsx: – Survey questions for TD-SAGAT plus results – Evaluation plus figures • Evaluations/Workshop_Survey_and_SurveyResults_Figures.xlsx: – Survey questions for Workshops plus results – Evaluation plus figures • Evaluations/Kickoff_Survey_and_SurveyResults.xlsx: – Survey questions for Kickoff Workshops (incl. questions on current TDM process) plus results – Evaluation in Workshop_Survey_and_SurveyResults_Figures.xslx • Visualizations/V<No.>_<adopted | mockup>_<explanation>.png: – "adopted" visualizations are similarly presented in the paper in Fig. 2. – "mockup" visualizations were presented during the workshops, but not adopted in this form. If a respective "adopted" visualization for this No. exists, the visualization was replaced by a more suitable one. If no respective "adopted" visualization for this No. exists, the visualization was discarded. • Visualizations/Dashboard <Manager | Team>.png: shows the dashboard mockups for the manager and the team, respectively. Visualizations presented in the dashboards that were not adopted by the team were removed or replaced by the improved visualizations in the final dashboard versions.

TABLE B.6: Additional Material to Publication No. 6 *(Establishing TDM)*

	7 *(Practical Guide)* **A Practical Guide for Establishing a Technical Debt Management Process**
Citation	M. Wiese, K. Serwa, E. Bittner, *A Practical Guide for Establishing a Technical Debt Management Process*, 2026 IEEE/ACM International Conference on Technical Debt (TechDebt), 2026, Rio de Janeiro, Brazil, doi = 10.1145/3794915.3795789
Link	https://doi.org/10.5281/zenodo.17415605
Content	The additional material for the paper *A Practical Guide for Establishing a Technical Debt Management Process* comprises the following documents: • TDM_Guide.pdf: Whitepaper for practitioners: Guideline to establish a TDM process • Team_Comparison_adoptedApproaches_challenges.xls: Tab: Adopted_Approaches_Overview; Tab: Challenges_Overview • Workshop Agendas.pdf: Example Agendas for Kick-Off and other workshops • Survey_Overview.xlsx: List of all survey questions, type, "mandatory" status, and their relevance/relation to the RQs • TDMGuide_0_InitialTalk.pdf Slides for the Initial Talk given to participants at the beginning of the action research to introduce some basic information and terminology • Team A/ADOSTDTickets_Template.pdf: Screenshot of the backlog's TD item template of Team A. • Team B/JiraTDTickets_Template.pdf: Screenshot of the backlog's TD item template of Team B. • Team C/GitHubTDTickets_Template.pdf: Screenshot of the backlog's TD item template of Team C. • Team$< A\|B\|C >$workshop_flipcharts/WS$<$ *No.* $>_<$ *explanation* $>$.jpg: Anonymized flipchart photos to give an impression of the workshops and their results. Translations are directly included in the photos. • Team$< A\|B\|C >$/workshop_slides/TDMGuide_$<$ *No* $>$_Workshop.pptx: Anonymized slides of each workshop to give an impression of the workshops and their results. • Team$< A\|B\|C >$/evaluations/Meeting_Observations_Researcher$<$ $1\|2 >$.xslx: Individual observation protocols of each researcher to show the comments of both researchers • Team$< A\|B\|C >$/evaluations/Meetings_Overview_combinedObservations_Evaluation_-Figures.xlsx: – Overview of all meetings, plus information on length, participants, whether a video exists, and was used for evaluation – Observation protocols results combined for both researchers (except for comments -> see above for comments) – Summation, evaluations, and resulting figures • Team$< A\|B\|C >$/evaluations/TD-SAGAT_Survey_and_SurveyResults_Figures.xlsx: Survey questions for TD-SAGAT plus results Evaluation plus figures • Team$< A\|B\|C >$/evaluations/Workshop_Survey_and_SurveyResults_Figures.xlsx: Survey questions for Workshops plus results; Evaluation plus figures • Team$< A\|B\|C >$/evaluations/Kickoff_Survey_and_SurveyResults.xlsx: Survey questions for Kickoff Workshops plus results; Evaluation in Workshop_Survey_and_SurveyResults_Figures.xslx: • Team$< A\|B\|C >$/evaluations/Backlog_Evaluations_and_Figures.xlsx: Last manual backlog evaluation before the visualisations could be used for this evaluation • Team$< A\|B >$/visualizations/V$<$ *No.* $>_<$ *explanation* $>$.png: Anonymized visualizations of Team A and B. Team C was not able to visualize the results due to technical issues. • Team$< A\|B >$/visualizations/Team$< A\|B >$_readme.txt: Text file with short information for each visualization

TABLE B.7: Additional Material to Publication No. 7 *(Practical Guide)*

8 *(AK in Web Search Engines)* **Exploring Web Search Engines to Find Architectural Knowledge**	
Citation	M. Soliman, M. Wiese, Y. Li, M. Riebisch and P. Avgeriou, *Exploring Web Search Engines to Find Architectural Knowledge*, 2021 IEEE 18th International Conference on Software Architecture (ICSA), 2021, Stuttgart, Germany, pp. 162-172, https://doi.org/10.1109/ICSA51549.2021.00023
Link	https://github.com/m-a-m-s/ICSA2021
Content	• Statistics: Calculations and preparations of statistics • Tasks: Tasks the participants had to solve • URLs: URLs participants found and the evaluation of those URLs, e.g., categorizations • Participants_Background_And_Exit_Survey.xlsx: statistical information on the participants. • Task_Sequences.xlsx: List of tasks randomly assigned to participants • User_Guide: Information given to the participants • plugin.zip: Chrome Plugin to automatically gather the search data

TABLE B.8: Additional Material to Publication No. 8 *(AK in Web Search Engines)*

9 *(Debiasing)* **Debiasing Architectural Decision-Making: An Experiment With Students and Practitioners**	
Citation	K. Borowa, R. R. de Almeida and M. Wiese, *Debiasing Architectural Decision-Making: An Experiment With Students and Practitioners*, 2025 IEEE 22nd International Conference on Software Architecture (ICSA), 2025, Odense, Denmark, pp. 210-220, https://doi.org/10.1109/ICSA65012.2025.00029
Link	https://doi.org/10.5281/zenodo.14740330
Content	Additional Material for: Debiasing Architectural Decision-Making: An Experiment With Students and Practitioners submitted to ICSA 2025. The files contain: 1. README.md - A short file describing the contents of the replication package. 2. Coding guide.docx - a guide used for coding, with examples from the experiment's participants. 3. Coding_results.xlsx - data with all measurements for all participants. 4. Questionnaire.xlsx - questionnaire used to gather demographic participants' data. 5. Experiment-plan.docx - the detailed plan of all the experiment steps. 6. Debiasing-workshop-slides.pptx - slides used during the debiasing workshop. 7. Debiasing-workshop-plan.docx - a detailed plan of the debiasing workshop, explaining how exactly the debiasing presentation should be used. 8. Consent_form.docx - a consent form that was signed by the participants from one of the author's countries, where guidelines suggested the use of gathering additional written consent.

TABLE B.9: Additional Material to Publication No. 9 *(Debiasing)*

Declaration of Artificial Intelligence Use

During the preparation of this work, I used Grammarly to improve the readability and language of the manuscript. After using this tool, I reviewed and edited the content as needed and take full responsibility for the content of the dissertation.

Eidesstattliche Versicherung

Eidesstattliche Versicherung:
Hiermit erkläre ich an Eides statt, dass ich die vorliegende Dissertationsschrift selbst verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Sofern im Zuge der Erstellung der vorliegenden Dissertationsschrift generative Künstliche Intelligenz (gKI) basierte elektronische Hilfsmittel verwendet wurden, versichere ich, dass meine eigene Leistung im Vordergrund stand und dass eine vollständige Dokumentation aller verwendeten Hilfsmittel gemäß der Guten wissenschaftlichen Praxis vorliegt. Ich trage die Verantwortung für eventuell durch die gKI generierte fehlerhafte oder verzerrte Inhalte, fehlerhafte Referenzen, Verstöße gegen das Datenschutz- und Urheberrecht oder Plagiate.

Affidavit:
I hereby declare and affirm that this doctoral dissertation is my own work and that I have not used any aids and sources other than those indicated. If electronic resources based on generative artificial intelligence (gAI) were used in the course of writing this dissertation, I confirm that my own work was the main and value-adding contribution and that complete documentation of all resources used is available in accordance with good scientific practice. I am responsible for any erroneous or distorted content, incorrect references, violations of data protection and copyright law or plagiarism that may have been generated by the gAI.

Unterschrift / Signature: (Marion Wiese)

Ort, den / City, date:

Identität eingereichter Versionen

Versicherung zur Identität der elektronischen und gedruckten Dissertationsschrift
Ich versichere, dass dieses gebundene Exemplar der Dissertation und das in elektronischer Form eingereichte Dissertationsexemplar (über den Docata-Upload) und das bei der Fakultät (zuständiges Studienbüro) zur Archivierung eingereichte gedruckte, gebundene Exemplar der Dissertationsschrift identisch sind.

I, the undersigned, declare that this bound copy of the dissertation and the dissertation submitted in electronic form (via the Docata upload) and the printed bound copy of the dissertation submitted to the faculty (responsible Academic Office) for archiving are identical.

Unterschrift / Signature: (Marion Wiese)

Ort, den / City, date: