PGCOMP - Programa de Pós-Graduação em Ciência da Computação
Universidade Federal da Bahia (UFBA)
Av. Milton Santos, s/n - Ondina
Salvador, BA, Brasil, 40170-110

https://pgcomp.ufba.br
pgcomp@ufba.br

Context: Technical debt (TD) describes the effects of immature artifacts on software development that can bring benefits in the short term but may have to be paid with interest in the long term. TD management balances short-term and long-term goals, supporting development teams to decide on the need and the best time to eliminate the debt. TD management activities include prevention, monitoring, and payment. Through prevention, it is possible to prevent teams from incurring TD, while monitoring helps them follow the evolution of TD items concerning the cost-benefit of eliminating them or not, that is, paying the debt items. Knowing the practices used to prevent, monitor, and pay TD items can help development teams to choose the best practice to be used in their projects. Identifying the practice avoidance reasons (PARs) that lead to non-prevention, non-monitoring, and non-payment of TD can help teams understand which aspects need to be improved to enable TD management. Although the technical literature has investigated the prevention, monitoring, and payment of TD, current results only reflect the viewpoint of a small number of professionals and organizations. To achieve the benefits of TD management, it is necessary to investigate more deeply the practices and PARs associated with these TD activities. **Aims:** This Ph.D. dissertation aims to investigate, through the continuous and independent replication of a family of surveys conducted globally, the state of practice on the prevention, monitoring, and payment of TD items in software projects. **Method:** Initially, we conducted a literature review on the current state of research on TD and its prevention, monitoring, and payment. Then, we analyzed data collected by six replication teams from the InsighTD project, which is a family of globally distributed surveys on the causes, effects, and management of TD. From the body of knowledge resulted from the analysis of InsighTD data, we defined three artifacts: an updated version of the conceptual model for TD, a set of conceptual maps, and IDEA (Impediments, Decision factors, Enabling practices, and Actions) diagrams. Finally, we assessed these artifacts through empirical studies in academic and industrial settings. **Results:** This Ph.D. dissertation presents the leading practices used to prevent, monitor, and pay off TD items and the PARs that justify the non-application of these practices. Regarding the prevention of TD, well-defined requirements, adopting good programming practices, and better project management are among the five most cited practices related to prevention, while short deadlines, ineffective management, and lack of predictability in the software development are among the five most cited PARs to justify the non-prevention of debt. About TD monitoring, TD item backlog, use of tools, and team meetings are among the five most cited practices related to monitoring, while lack of interest, focus on short-term goals, and lack of time are among the five PARs used to explain the non-monitoring of TD items. Regarding TD payment, code refactoring, investing effort in TD payment activities, and design refactoring are among the top five payment-related practices, while focusing on short-term goals, lack of organizational interest, and lack of time are among the five most cited PARs to explain the non-payment of TD. We update the conceptual model for TD by including the knowledge we learn from the state of practice and organize all practices and PARs along with their types, natures, and categories into maps and IDEA diagrams. From the conceptual model and TD payment map assessment, we found that they are well organized and provide valuable information to define strategies for TD management. The IDEA diagrams assessment provided positive evidence that the diagrams are easy to read and follow and can influence decisions on how to manage TD items. **Conclusion:** Using the InsighTD data, this Ph.D. dissertation explores the state of practice on TD prevention, monitoring, and payment, revealing the primary practices used to perform these activities and the PARs that avoid their execution. All body of knowledge was organized into three artifacts that can drive new investigations on TD and support software practitioners in increasing their capabilities and reducing their issues in managing debt items.

DSC | 037 | 2023

Organizing the State of Practice on Technical Debt Prevention, Monitoring, and Payment in Software Projects

Emmanuel Sávio Silva Freire

UFBA

Organizing the State of Practice on Technical Debt Prevention, Monitoring, and Payment in Software Projects

Emmanuel Sávio Silva Freire

Tese de Doutorado

Universidade Federal da Bahia

Programa de Pós-Graduação em Ciência da Computação

Janeiro | 2023

Universidade Federal da Bahia
Instituto de Computação

Programa de Pós-Graduação em Ciência da Computação

ORGANIZING THE STATE OF PRACTICE ON TECHNICAL DEBT PREVENTION, MONITORING, AND PAYMENT IN SOFTWARE PROJECTS

Emmanuel Sávio Silva Freire

TESE DE DOUTORADO

Salvador
30 de janeiro de 2023

EMMANUEL SÁVIO SILVA FREIRE

ORGANIZING THE STATE OF PRACTICE ON TECHNICAL DEBT PREVENTION, MONITORING, AND PAYMENT IN SOFTWARE PROJECTS

Esta Tese de Doutorado foi apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal da Bahia, como requisito parcial para obtenção do grau de Doutor em Ciência da Computação.

Orientador: Prof. Dr. Manoel Gomes de Mendonça Neto
Co-orientador: Prof. Dr. Rodrigo Oliveira Spínola

Salvador
30 de janeiro de 2023

Ficha catalográfica elaborada pela Biblioteca Universitária de
Ciências e Tecnologias Prof. Omar Catunda, SIBI – UFBA.

F866 Freire, Emmanuel Sávio Silva

Organizing the state of practice on technical debt prevention, monitoring, and payment in software projects/ Emmanuel Sávio Silva Freire. – Salvador, 2023.

252 f.

Orientador: Prof. Dr. Manoel Gomes de Mendonça Neto
Coorientador: Prof. Dr. Rodrigo Oliveira Spínola

Tese (Doutorado) – Universidade Federal da Bahia. Instituto de Computação, 2023.

1. Technical debt. 2. Technical debt management. 3. Software. I. Mendonça, Manoel Gomes de. II. Spínola, Rodrigo Oliveira. III. Universidade Federal da Bahia. IV. Título.

CDU 004

EMMANUEL SÁVIO SILVA FREIRE

ORGANIZING THE STATE OF PRACTICE ON TECHNICAL DEBT PREVENTION, MONITORING, AND PAYMENT IN SOFTWARE PROJECTS

Esta tese foi julgada adequada à obtenção do título de Doutor em Ciência da Computação e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação da UFBA.

Salvador, 30 de janeiro de 2023

[illegible]

Prof. Dr. Manoel Gomes de Mendonça Neto
(Orientador-UFBA)

[illegible]

Prof. Dr. Daniela Soares Cruzes
Norwegian University of Science and Technology/Norway

[illegible]

Prof. Dr. Uirá Kulesza
Federal University of Rio Grande do Norte/Brazil

Thiago Souto Mendes

Prof. Dr. Thiago Souto Mendes
Federal Institute of Bahia/Brazil

Renato Lima Novais

Prof. Dr. Renato Lima Novais
Federal Institute of Bahia/Brazil

To my son (Eduardo), my wife (Shirley), and my family.

ACKNOWLEDGEMENTS

First and foremost, I would like to express my gratitude to God for always guiding me, especially in times of difficulty, and empowering me in the face of challenges.

I want to express deep and sincere gratitude to my advisors, Manoel Mendonça and Rodrigo Spínola. Manoel Mendonça provided an excellent place to develop my doctorate project at UFBA and welcomed me into his research group. Through our conversations, I learned much more than just research in the software engineering field. Thanks for his valuable support. Rodrigo Spínola believed in me since we met at the Brazilian Symposium on Software Quality (SBQS'18), inviting me to do my doctorate under his and Mendonça's guidance. Thanks for sharing his research knowledge with me and for allowing me opportunities to work with different researchers in the technical debt field. His motivation and guidance were essential for the execution of this work.

I want to thank my wife, Shirley, for her patience, especially during the first year of the doctorate, when I traveled back and forth between Salvador and Fortaleza, and for listening to a lot about my research topic. And thanks for her support and for giving me the best gift of my life: our son (Eduardo). He came to light up our lives and bring us more happiness.

I also thank my parents and my sister for their support. First, to my father, Aderbal, who closely followed the beginning of my doctorate, always overly concerned and attentive, but unfortunately, he passed away in 2021. I will always treasure his teachings and the moments we lived together with me. Then, I thank my mother, Francisca, for being an example of strength, lightness, and love. Thank her so much for always motivating me. And lastly, to my sister, Cibely, for all her dedication to our family. Her advice always makes a difference.

I would like to extend my sincere thanks to other members of my family for celebrating each achievement with me: Fátima (my mother-in-law), Murilo (my brother-in-law), João Lucas (my nephew), Rennan (my nephew), Sheila (my sister-in-law), and Júnior (my brother-in-law). Cibely and Murilo, thanks for giving the new member of our family, Bernardo.

I want to thank all colleagues and friends from LES, INÊS, and LABES^2 laboratories, especially Dênia Campos, Tiago Motta, Renata Sousa, Gláucya Boechat, Iuri Sousa, Larissa Soares, Larissa Barbosa, Verusca Rocha, Felipe Gomes, Éder Pereira, Lucinéia Batista, Luis Paulo, Antônio Carlos, Lidianny Cerqueira, and Beatriz Santana. Thanks for working together or for the suggestions offered during the completion of the dissertation. Dênia, Tiago, and Gláucya, thanks for having me so well in Salvador!

I want to thank Amanda Ferraz, Monck Charles, Júlio Oliveira, Verônica Reis, Ícaro Ramires, Dênia, Renata Santos, and João Victor for sharing their knowledge in our studies in the Algorithms and Graphs subject. Their support was essential. I also thank

Eliakim Gama and Diego Costa for the opportunity to collaborate in their master's thesis projects at the State University of Ceará (UECE).

A special thanks to all members of the InsighTD project and the technical debt research team, especially Rodrigo Spínola, Manoel Mendonça, Nicolli Rios, Carolyn Seaman, Clemente Izurieta, Bóris Pérez, Alexia Pacheco, Gustavo López, Vladimir Mandi, Camilo Castellanos, Darío Correal, Robert Rama, Neboja Tauan, and Davide Falessi. Thanks for designing the InsighTD survey and collecting the data in their countries. The collected data and our collaboration were essential to this dissertation and the high-level publications we obtained together.

Many thanks to the colleagues and friends who shared the apartment 203 with me in Salvador. Tiago Motta, Tássio Virgínio, Iuri Sousa, Uoston Sousa, and Andre were essential during the first year of the doctorate. Thanks for their support, friendship, and long conversations.

I thank my friends Roseanny Nascimento, Paulo Parente, Raquel Sena, Nádia Esmeraldo, Stephanie Menezes, Daianne Diógenes, Marina Araújo, Ivan Mota, and Wladimir Diniz for make my life enjoyable outside work. My godson Miguel also contributed to this.

I am also grateful to the Federal Institute of Ceará / Morada Nova campus for allowing me to take a leave of absence during my doctorate course. Special thanks to Beatriz Claudino, Leonardo Galvão, Julliano Cruz, Adrileide Almeida and all members of the collegiate of the informatics course. Also, a special thanks to Joselice Siebra and Antônio José for taking over my subjects in technical collaboration at the suggestion of Raquel Silveira, whom I also thank.

A special thanks for professors Mariela Cortés and Antônio Serra for their support and friendship.

Thanks should also go to my examination committee composed of the professors Daniela Soares Cruzes, Uirá Kulesza, Renato Lima Novais, and Thiago Souto Mendes for their willingness to evaluate my work and for their valuable contributions.

Many thanks to all professors and collaborates who are part of the Programa de Pós-graduação em Ciência da Computação (PGCOMP) at UFBA.

I thank the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) for partial funding through the Programa de Formação Doutoral Docente (prodoutoral) scholarship.

Lastly, thanks for all participants who answered the InsighTD survey or participated in the other empirical studies performed in this dissertation.

“Ah, que bom você chegou
Bem-vindo a Salvador
Coração do Brasil”
—NIZAN GUANÃES

RESUMO

Contexto: Dívida Técnica (DT) descreve os efeitos de artefatos imaturos no desenvolvimento de software que podem trazer benefícios a curto prazo, mas que poderão ter que ser pagos com juros a longo prazo. O gerenciamento da DT equilibra os objetivos de curto e longo prazo, auxiliando as equipes de desenvolvimento a decidir sobre a necessidade e o melhor momento para eliminar a dívida. Dentre as atividades de gerenciamento da DT, têm-se a prevenção, o monitoramento e o pagamento. Por meio da prevenção, é possível evitar que equipes incorram em DT, enquanto o monitoramento as auxilia a acompanhar a evolução dos itens da DT em relação ao custo-benefício de eliminá-los ou não, ou seja, pagar os itens da dívida. Conhecer as práticas utilizadas para prevenir, monitorar e pagar os itens da DT pode auxiliar equipes de desenvolvimento na escolha da melhor prática a ser utilizada nos seus projetos. Identificar as razões que levam à não prevenção, ao não monitoramento e ao não pagamento da DT pode ajudar equipes a entenderem quais aspectos precisam ser melhorados para possibilitar o gerenciamento da DT. Embora a literatura técnica tenha investigado a prevenção, o monitoramento e o pagamento da DT, os resultados atuais refletem apenas o ponto de vista de um pequeno número de profissionais e organizações. Para alcançar os benefícios da gestão da DT, é necessário investigar mais profundamente as práticas e as razões associados a essas atividades da DT.

Objetivo: Esta tese tem como objetivo investigar, por meio da replicação contínua e independente de uma família de *surveys* conduzidos globalmente, o estado da prática sobre a prevenção, o monitoramento e o pagamento de itens da DT em projetos de software.

Método: Inicialmente, foi realizada uma revisão da literatura sobre o estado atual da pesquisa sobre DT e sua prevenção, monitoramento e pagamento. Em seguida, foram analisados os dados coletados por seis equipes de replicação do projeto InsighTD, uma família de *surveys* globalmente distribuída sobre causas, efeitos e gerenciamento da DT. A partir do corpo de conhecimento resultante das análises dos dados de InsighTD, foram definidos três artefatos: uma versão atualizada do modelo conceitual de DT, um conjunto de mapas conceituais e os diagramas IDEA (*Impediments, Decision factors, Enabling practices, and Actions*). Por fim, os três artefatos foram avaliados por meio de estudos experimentais na academia e na indústria.

Resultados: Esta tese apresenta as principais práticas utilizadas para prevenir, monitorar e pagar itens da DT e as razões que justificam a não aplicação dessas práticas. Em relação à prevenção da DT, *requisitos bem definidos*, *adoção de boas práticas de programação* e *melhor gerenciamento do projeto* estão entre as cinco práticas relacionadas à prevenção mais citadas, enquanto *prazo curto*, *gerenciamento ineficiente* e *falta de previsibilidade no desenvolvimento de software* estão entre as cinco razões mais citadas para justificar a não prevenção da dívida. Sobre o monitoramento da DT, *backlog composto por itens da DT*, *utilização de ferramentas* e *reuniões da equipe* estão entre as cinco práticas relacionadas ao monitoramento mais citadas, enquanto *falta de interesse*, *foco em metas de curto prazo* e *falta de tempo* estão entre as cinco razões utilizadas para explicar o não monitoramento de itens da DT. Em relação ao pagamento da DT, *refatoração de código*, *investindo esforço nas atividades de pagamento da DT* e *refatoração de design* estão entre as cinco práticas relacionadas ao pagamento mais citadas, enquanto *foco em metas de curto prazo*, *falta de interesse organizacional* e *falta de tempo* estão entre as cinco razões mais citadas para explicar o não pagamento da DT. O modelo conceitual da DT foi atualizado por meio da inclusão do conhecimento oriundo do estado da prática e todas as práticas e razões, juntamente com seus tipos, natureza e categorias, foram organizadas em mapas conceituais e diagramas IDEA. Por meio da avaliação do modelo conceitual e do mapa de pagamento da DT, foi encontrado que eles são bem-organizados e fornecem informações valiosas para definir estratégias de gerenciamento da DT. A avaliação dos diagramas IDEA forneceu evidências positivas de que eles são fáceis de ler e seguir e podem influenciar as decisões sobre como gerenciar itens de TD.

Conclusão: Utilizando dados de InsighTD, esta tese explora o estado da prática da prevenção, monitoramento e pagamento da DT, revelando as principais práticas utilizadas para realizar essas atividades e as razões que evitam sua execução. Todo o corpo de conhecimento foi organizado em três artefatos que podem guiar novas investigações sobre a DT e apoiar os profissionais de software a aumentar suas capacidades e reduzir seus problemas no gerenciamento de itens de dívida.

Palavras-chave: dívida técnica, gerenciamento da dívida técnica, prevenção da dívida técnica, monitoramento da dívida técnica, pagamento da dívida técnica, família de *surveys*, diagramas IDEA.

ABSTRACT

Context: Technical debt (TD) describes the effects of immature artifacts on software development that can bring benefits in the short term but may have to be paid with interest in the long term. TD management balances short-term and long-term goals, supporting development teams to decide on the need and the best time to eliminate the debt. TD management activities include prevention, monitoring, and payment. Through prevention, it is possible to prevent teams from incurring TD, while monitoring helps them follow the evolution of TD items concerning the cost-benefit of eliminating them or not, that is, paying the debt items. Knowing the practices used to prevent, monitor, and pay TD items can help development teams to choose the best practice to be used in their projects. Identifying the practice avoidance reasons (PARs) that lead to non-prevention, non-monitoring, and non-payment of TD can help teams understand which aspects need to be improved to enable TD management. Although the technical literature has investigated the prevention, monitoring, and payment of TD, current results only reflect the viewpoint of a small number of professionals and organizations. To achieve the benefits of TD management, it is necessary to investigate more deeply the practices and PARs associated with these TD activities.

Aims: This Ph.D. dissertation aims to investigate, through the continuous and independent replication of a family of surveys conducted globally, the state of practice on the prevention, monitoring, and payment of TD items in software projects.

Method: Initially, we conducted a literature review on the current state of research on TD and its prevention, monitoring, and payment. Then, we analyzed data collected by six replication teams from the InsighTD project, which is a family of globally distributed surveys on the causes, effects, and management of TD. From the body of knowledge resulted from the analysis of InsighTD data, we defined three artifacts: an updated version of the conceptual model for TD, a set of conceptual maps, and IDEA (Impediments, Decision factors, Enabling practices, and Actions) diagrams. Finally, we assessed these artifacts through empirical studies in academic and industrial settings.

Results: This Ph.D. dissertation presents the leading practices used to prevent, monitor, and pay off TD items and the PARs that justify the non-application of these practices. Regarding the prevention of TD, *well-defined requirements*, *adopting good programming practices*, and *better project management* are among the five most cited practices related to prevention, while *short deadlines*, *ineffective management*, and *lack of predictability in the software development* are among the five most cited PARs to justify the non-prevention of debt. About TD monitoring, *TD item backlog*, *use of tools*, and *team meetings* are among the five most cited practices related to monitoring, while *lack of interest*, *focus on short-term goals*, and *lack of time* are among the five PARs used to explain the non-monitoring of TD items. Regarding TD payment, *code refactoring*, *investing effort in TD payment activities*, and *design refactoring* are among the top five payment-related practices, while *focusing on short-term goals*, *lack of organizational interest*, and *lack of time* are among the five most cited PARs to explain the non-payment of TD. We update the conceptual model for TD by including the knowledge we learn from the state of practice and organize all practices and PARs along with their types, natures, and categories into maps and IDEA diagrams. From the conceptual model and TD payment map assessment, we found that they are well organized and provide valuable information to define strategies for TD management. The IDEA diagrams assessment provided positive evidence that the diagrams are easy to read and follow and can influence decisions on how to manage TD items.

Conclusion: Using the InsighTD data, this Ph.D. dissertation explores the state of practice on TD prevention, monitoring, and payment, revealing the primary practices used to perform these activities and the PARs that avoid their execution. All body of knowledge was organized into three artifacts that can drive new investigations on TD and support software practitioners in increasing their capabilities and reducing their issues in managing debt items.

Keywords: technical debt, technical debt management, technical debt prevention, technical debt monitoring, technical debt payment, family of surveys, IDEA diagrams.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

IDE	Integrated development environment
IDEA	Impediments, decision factors, enabling practices, and actions
iTDp-RQ	InsighTD project research question
MVC	Model-View-Controller
NTS	National Transplantation System
PAR	Practice avoidance reason
PGCOMP	Programa de Pós-Graduação em Ciência da Computação
Q	Question
RQ	Research question
SG	Specific goal
SQALE	Software quality assessment based on lifecycle expectations
SWOT	Strengths, weaknesses, opportunities, and threats
TAM	Technology acceptance model
TD	Technical debt
UML	Unified modeling language

Chapter
1

This chapter presents the context of this dissertation, its motivation, goals, and research questions. It also discusses the methodology, the main contributions, and the out-of-scope items of this work.

INTRODUCTION

Software development teams usually perform their activities under time and resource constraints, affecting the quality of systems and impacting software maintenance and evolution activities (LIENTZ; SWANSON; TOMPKINS, 1978). In these cases, long-term productivity impacts can be observed, as modifying low-quality software often requires more effort from a development team than maintaining high-quality software (LEHMAN; BELADY, 1985).

In the software industry, the term *Technical debt (TD)* contextualizes these technical compromises that can bring short-term benefits, such as higher productivity and lower costs, but may negatively impact the long-term health of software projects (IZURIETA *et al.*, 2012; SPINOLA *et al.*, 2019). The negative impacts yield risks associated with unexpected delays in system evolution and difficulty in achieving quality criteria defined for the project (IZURIETA *et al.*, 2012; SPINOLA *et al.*, 2019).

The term *Technical debt (TD)* was first used by Cunningham (1992) when he introduced the metaphor "going into debt," indicating that a small debt could speed up software development in the short term. However, every extra minute spent on poorly built code would count as interest on that debt (CUNNINGHAM, 1992). Over the last decade, the term has aroused increasing interest in the scientific community that seeks to understand how this phenomenon occurs and how to use TD management in practice during software development, maintenance, and evolution processes (LI; AVGERIOU; LIANG, 2015; AMPATZOGLOU *et al.*, 2015; ALVES *et al.*, 2016; RIBEIRO *et al.*, 2016; BEHUTIYE *et al.*, 2017; RIOS; MENDONÇA; SPINOLA, 2018).

1.1 PROBLEM STATEMENT

As mentioned before, TD describes the effect of immature artifacts on the software development process, bringing benefits to projects in the short term but which can be adjusted with interest later. The benefits can be seen as higher productivity and lower cost. At the same time, the “interest” is associated with unexpected delays in system evolution activities and the difficulty in achieving the quality criteria defined for the project (SPINOLA *et al.*, 2013; ZAZWORKA *et al.*, 2013).

Several studies (LI; AVGERIOU; LIANG, 2015; AMPATZOGLOU *et al.*, 2015; ALVES *et al.*, 2016; RIBEIRO *et al.*, 2016; BEHUTIYE *et al.*, 2017; RIOS; MENDONÇA; SPINOLA, 2018) have addressed the identification of TD items and strategies for their management. However, not all TD items identified in the project necessarily need to be eliminated, as, in some cases, these items may not influence the evolution of the software (BROWN *et al.*, 2010). Through TD management activities, a project team can balance short-term and long-term goals, supporting decision-making on the need and the best time to eliminate a debt item (GUO; SPINOLA; SEAMAN, 2016).

TD management comprises a set of activities that include the identification, measurement, prioritization, prevention, monitoring, documentation, communication, visualization, time-to-market analysis, scenario analysis, and payment of debt items (LI; AVGERIOU; LIANG, 2015; RIOS; MENDONÇA; SPINOLA, 2018). Among those, TD prevention, TD Monitoring, and TD payment are critical. The **TD prevention** activity enables software development teams to prevent TD items from occurring in the project (LI; AVGERIOU; LIANG, 2015; RIOS; MENDONÇA; SPINOLA, 2018). The **TD monitoring** activity seeks to support software teams in observing changes in the cost and benefit of debt items that have not yet been eliminated during the project evolution (LI; AVGERIOU; LIANG, 2015; RIOS; MENDONÇA; SPINOLA, 2018). Lastly, the **TD payment** activity supports decision-making about the most appropriate time to pay debt items and the choice and application of practices that should be used to pay TD items (LI; AVGERIOU; LIANG, 2015; RIOS; MENDONÇA; SPINOLA, 2018).

Many authors have holistically researched the practices used by software professionals to prevent, monitor, and pay TD items (ERNST *et al.*, 2015; LI; AVGERIOU; LIANG, 2015; YLI-HUUMO; MAGLYAS; SMOLANDER, 2016; BEHUTIYE *et al.*, 2017; MARTINI; BESKER; BOSCH, 2018; TOLEDO *et al.*, 2019; APA *et al.*, 2020b; RIOS *et al.*, 2020; ARAGÃO *et al.*, 2022; ALBUQUERQUE *et al.*, 2022; BONFIM; BENITTI, 2022). However, the results are limited as most studies address only one type of debt and focus on specific case studies. Only Ernst *et al.* (2015), Martini, Besker and Bosch (2018), and Apa *et al.* (2020b) considered a sizable number of study participants but they have a limited variety of software development contexts.

Our work stems from the fact that a broader investigation of the practices used to prevent, monitor, and pay off TD items is required. Moreover, it also proposes to investigate the reasons used to justify the non-prevention, non-monitoring, and non-payment of TD items. Herein, these reasons are called *Practice avoidance reasons (PARs).*

Identifying the practices used by software practitioners to manage the debt and the PARs that lead them not to manage debt is fundamental to guiding new research related to TD. Knowing the management practices can help software practitioners identify new practices they have not yet used. Knowing the PARs can aid development teams in understating which aspects need to be improved to prevent, monitor, and pay off TD items.

Our work proposes to holistically address TD management practices and PARs by considering the perspectives of software engineering practitioners on TD prevention, monitoring, and payment activities.

The following sections present the goals and research questions defined for this investigation.

1.2 MAIN GOAL AND RESEARCH QUESTIONS

This work aims **to investigate, through continuous and independent replication of a globally distributed family of surveys, the state of practice on TD prevention, monitoring, and payment in software projects**. To this end, we split our main goal into a set of Research questions (RQs), as shown in Table 1.1.

Table 1.1 High-level research questions

ID	Research question (RQ)
RQ1	**How could software development teams avoid Technical debt items on their projects?** This question aims to identify the preventive practices that could prevent potential TD items and the PARs that could explain TD non-prevention.
RQ2	**How have software development teams monitored Technical debt items on their projects?** Through this question, we seek to identify the practices used for monitoring the debt and the PARs to explain the non-monitoring of TD items.
RQ3	**How have software development teams paid off Technical debt items on their projects?** This question aims to identify the practices used by teams to eliminate TD items from their projects and the PARs that explain the non-elimination of these items.
RQ4	**How to organize the body of knowledge composed of prevention, monitoring, and payment practices - and PARs for TD non-prevention, non-monitoring, and non-payment - to support TD management?** This question investigates how the set of practices and PARs can be organized to support the management of TD items.

To achieve our goal and answer the research questions, we have defined the following Specific goals (SGs):

SG1. **To investigate the current state of TD research by identifying studies that addressed TD prevention, monitoring, and payment**. We conducted a literature review on TD management, considering TD prevention, monitoring, and payment activities.

SG2. **To investigate the state of the practice on TD prevention, monitoring, and payment**. We analyzed data collected by the replication teams of the InsighTD project, a globally distributed family of surveys on the causes, effects, and management practices of TD (RIOS *et al.*, 2018; RIOS *et al.*, 2020). Our study compiled responses from more than six hundred software practitioners from six different countries around the world.

SG3. **To organize the body of knowledge composed of practices and PARs identified in SG2**. We extended the conceptual model for TD initially proposed by Avgeriou *et al.* (2016) and Izurieta *et al.* (2016) and evolved by Rios, Mendonça and Spinola (2018), including the concepts found from the state of practice. We also build conceptual maps that organize the body of knowledge on practices and PARs associated with TD prevention, monitoring, and payment. We then structured the set of practices and PARs into Impediments, decision factors, enabling practices, and actions (IDEA) Diagrams. Lastly, we provided guidelines on how to use the maps and the IDEA Diagrams in practice.

SG4. **To assess the body of knowledge organized in SG3**. We planned and executed three empirical studies to assess the body of knowledge derived from SG3. Firstly, we conducted a follow-up survey to partially assess the extended conceptual model and the TD payment map with software practitioners. Lastly, we applied the Technology acceptance model (TAM) (DAVIS, 1989) with undergraduate students in academic contexts and interviewed experienced software practitioners to collect their perceptions of the IDEA Diagrams.

1.3 METHODOLOGY

The activities defined and performed in this dissertation are based on the Experimental Software Engineering paradigm (BASILI, 1993), which aims to assist in better evaluating, predicting, understanding, controlling, and improving software engineering practices (BASILI; SELBY; HUTCHENS, 1986).

Figure 1.1 presents the methodological steps we followed to achieve the work's goal. Each step comprises four elements: goal, activity performed, the result obtained, and result dissemination. We used different shapes and colors to identify each of these elements. The goals are represented by white rectangles and their activities by blue pentagons. The obtained results are represented by rounded green rectangles. Finally, we used an icon to describe the disseminated results, that is, articles that have been published (red icon) or that are in the review process (orange icon) in the literature. We also identified if an article is a conference (C) or a journal (J) paper. For example, the paper "C1" is a published conference paper, while "J1" is a journal paper in the review process. The last chapter presents more detail on disseminated, including some secondary contributions of our research.

We detail our work methodology activities below:

1. **Initial literature review**: we conducted a literature review on TD, aiming to understand its concept and state of the art on the subject. Chapter 2 presents the results of this activity.

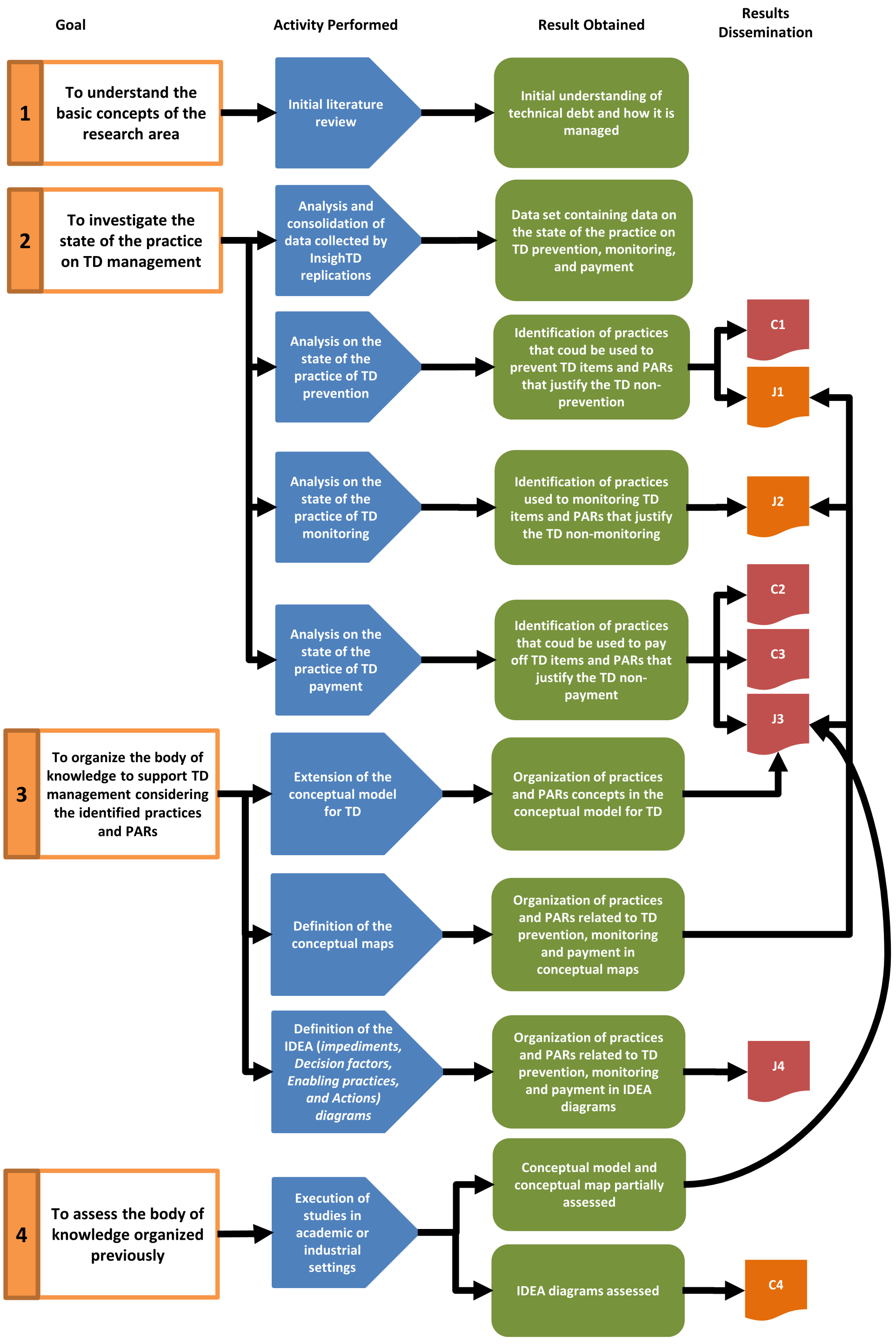

Goal
Activity Performed
Result Obtained
Results Dissemination
1
To understand the basic concepts of the research area
Initial literature review
Initial understanding of technical debt and how it is managed
2
To investigate the state of the practice on TD management
Analysis and consolidation of data collected by InsighTD replications
Data set containing data on the state of the practice on TD prevention, monitoring, and payment
Analysis on the state of the practice of TD prevention
Identification of practices that coud be used to prevent TD items and PARs that justify the TD non-prevention
C1
J1
Analysis on the state of the practice of TD monitoring
Identification of practices used to monitoring TD items and PARs that justify the TD non-monitoring
J2
Analysis on the state of the practice of TD payment
Identification of practices that coud be used to pay off TD items and PARs that justify the TD non-payment
C2
C3
J3
3
To organize the body of knowledge to support TD management considering the identified practices and PARs
Extension of the conceptual model for TD
Organization of practices and PARs concepts in the conceptual model for TD
Definition of the conceptual maps
Organization of practices and PARs related to TD prevention, monitoring and payment in conceptual maps
Definition of the IDEA (impediments, Decision factors, Enabling practices, and Actions) diagrams
Organization of practices and PARs related to TD prevention, monitoring and payment in IDEA diagrams
J4
4
To assess the body of knowledge organized previously
Execution of studies in academic or industrial settings
Conceptual model and conceptual map partially assessed
IDEA diagrams assessed
C4

Figure 1.1 Work methodology

2. **Analysis and consolidation of data collected by InsighTD replications**: we contributed to the InsighTD Project by conducting data analysis on the survey's questions about TD management. Furthermore, we consolidated the data collected in Brazil with others collected by InsighTD replications conducted in Chile, Colombia, Costa Rica, Serbia, and the United States. In total, our data set comprises 653 responses from software professionals from these six countries.

3. **Analysis of the state of the practice of TD prevention**: we analyzed the data from the InsighTD survey's questions on TD prevention to identify practices and PARs. Initially, we analyzed a data set composed of answers collected by the Brazilian and North American InsighTD replication teams. Afterward, we consolidated the analysis with the results of Chilean, Colombian, Costa Rican, and Serbian InsighTD replication teams.

4. **Analysis of the state of the practice of TD monitoring**: To recognize practices and PARs associated with TD monitoring, we analyzed the InsighTD survey's questions on TD monitoring, considering only the data collected in Brazil and the United States. Afterward, we consolidated the analysis by including the results of Chilean, Colombian, Costa Rican, and Serbian InsighTD replication teams.

5. **Analysis of the state of the practice of TD payment**: in this activity, we analyzed the answers collected by the Brazilian and North American InsighTD replication teams, considering the InsighTD survey's questions on TD payment. Then, we consolidated the analysis by adding the findings from Chilean, Colombian, Costa Rican, and Serbian InsighTD replication teams.

6. **Extension of the TD conceptual model**: we extended the conceptual model initially proposed by Avgeriou *et al.* (2016) and Izurieta *et al.* (2016) and evolved by Rios, Mendonça and Spinola (2018), including the knowledge of TD prevention, monitoring, and payment we learned from the state of practice.

7. **Definition of conceptual maps**: we defined three conceptual maps that summarizes the practices and PARs we found in the state of practice. Each map corresponds to one TD management activity (prevention, monitoring, and payment).

8. **Definition of the IDEA Diagrams**: we defined the IDEA diagrams, explaining their structure and how to use them for supporting TD management activities. We also specialized the diagrams by a process model and types of debt.

9. **Execution of studies in academia and industry**: we planned and conducted three studies to assess the conceptual model, conceptual maps, and IDEA diagrams. By conducting a follow-up survey with InsighTD participants, we partially evaluated the conceptual model and the TD payment map. By completing a TAM study with undergraduate students, we characterized the IDEA diagrams concerning ease of use, usefulness, and potential future use. By conducting an interview-based study with experienced software practitioners, we characterized the experts' perceptions of the diagrams concerning their support for TD management activities.

1.4 MAIN CONTRIBUTIONS

This dissertation investigates the state of practice on TD management using the data collected by the InsighTD replications conducted in Brazil, Chile, Colombia, Costa Rica, the United States, and Serbia. The InsighTD project is the first large-scale study of TD, its causes, effects, and management[1]. As previously mentioned, we focus on using InsighTD data to learn from software practitioners about TD prevention, monitoring, and payment activities in the Software Engineering field. Our main contributions are:

- A review of the current state of research on TD management, allowing for identifying key research findings that addressed TD prevention, monitoring, and payment.
- The organization of an open and generalizable set of empirical evidence on TD prevention, monitoring, and payment collected from the InsighTD project.
- The analysis and synthesis of results from six InsighTD replications about TD prevention, monitoring, and payment.
- A comprehensive list of TD prevention, monitoring, and payment practices used by software practitioners to manage TD in their software projects.
- A comprehensive list of PARs that justify the non-application of prevention, monitoring, and payment practices in software development projects.
- An analysis of prevention, monitoring, and payment practices and PARs that reveal their types, nature, categories, and relationships with types of debt.
- An updated version of the TD conceptual model, including the concepts from the state of practice. The model can guide new research efforts aligned with the demands and current context of TD management as experienced by practitioners.
- The empirical evidence about the accuracy and completeness of the updated version of TD conceptual model in relation to the software practitioners perception about its representation of TD payment concepts.
- A set of conceptual maps that organize the body of knowledge on practices and PARs associated with TD prevention, monitoring, and payment. These maps aim to guide software practitioners on what to employ (practices) or curb (PARs) based on experience from other development teams.
- The empirical evidence about the accuracy and completeness of the TD payment map in software practitioners point of view.
- A set of IDEA diagrams that organize the body of knowledge on practices and PARs associated with TD prevention, monitoring, and payment. These diagrams also support software practitioners in recognizing their capability and impediments related

[1]<https://www.td-survey.com/>

to TD prevention, monitoring, and payment and help define strategies for boosting the capabilities and reducing the impediments.

- The empirical evidence about the diagrams' support to TD management activities derived from two complementary empirical studies, one performed in an academic setting and another in the software industry.

1.5 WHAT IS OUTSIDE THE SCOPE OF THIS WORK

The out-of-scope items of this dissertation are:

- Although the practices and PARs can be investigated considering any TD management activity, this dissertation only investigates TD prevention, monitoring, and payment activities.

- The InsighTD project allows for continuous and independent replication of the survey in different countries. Up to now, twelve replication teams have joined the project. However, we only considered data from six replications of InsighTD (Brazil, Colombia, Chile, Costa Rica, Serbia, and the United States), which are the replications performed so far.

- We use the knowledge obtained from the analysis of InsighTD data to build the IDEA diagrams and then evaluate them through studies in academic and industry settings. However, it is out of the scope of this dissertation to define new TD management strategies using the produced diagrams.

1.6 OUTLINE OF THIS DISSERTATION

We divide this document as follows:

- **Chapter 2. Technical Debt and its Management** provides an overview of TD and its management, focusing on TD prevention, monitoring, and payment activities.

- **Chapter 3. Research Method** introduces the InsighTD Project and how we collected and analyzed its data set. This chapter also discusses the threats to validity that affect the results obtained from the project.

- **Chapter 4. Technical Debt Prevention's State of Practice** presents our results on TD prevention from the InsighTD project.

- **Chapter 5. Technical Debt Monitoring's State of Practice** presents our results on TD monitoring from the InsighTD project.

- **Chapter 6. Technical Debt Payment's State of Practice** presents our results on TD payment from the InsighTD project.

- **Chapter 7. TD Conceptual Model and TD Management Conceptual Maps** presents the updated TD conceptual model and TD prevention, monitoring, and payment conceptual maps. Also, this chapter presents the follow-up survey we performed to assess these artifacts.

- **Chapter 8. IDEA Diagrams** presents the IDEA (Impediments, Decision factors, Enabling practices, and Actions) diagrams - their structure and ways to use - and presents the studies we performed to assess them.

- **Chapter 9. Concluding Remarks** presents the work history and its limitations, summarizes its main findings, and points out perspectives on future research directions.

This dissertation also has the following appendices:

- APPENDIX A - InsighTD Questionnaire presents the complete questionnaire used for all InsighTD replications teams.

- APPENDIX B - Technical Debt Prevention - Complementary Material presents the detailed analyses we performed on the InsighTD data described in Chapter 4.

- APPENDIX C - Technical Debt Monitoring - Complementary Material presents the detailed analyses we performed on the InsighTD data described in Chapter 5.

- APPENDIX D - Technical Debt Payment - Complementary Material presents the detailed analyses we performed on the InsighTD data described in Chapter 6.

- APPENDIX E - Follow-up Survey presents the questionnaire we used to assess the updated model for TD and TD payment map as described in Chapter 7.

- APPENDIX F - Specializations of IDEA Diagrams presents the IDEA diagrams specialized per process models and types of debt described in Chapter 8.

- APPENDIX G - Evaluation Form presents the set of questions used in the assessment of IDEA diagrams described in Chapter 8.

Chapter
2

This chapter introduces technical debt and its management, focusing on technical debt prevention, monitoring, and payment activities. It also presents related work on these activities.

TECHNICAL DEBT AND ITS MANAGEMENT

Technical debt (TD) emerges from shortcuts or even mistakes software practitioners make during software projects' development (AVGERIOU *et al.*, 2016; FALESSI; KAZMAN, 2021). TD can bring short-term benefits, such as increased development speed, but may have to be paid for with extra effort when the debt needs to be eliminated (IZURIETA *et al.*, 2012). Although TD can be a good investment, it can become a critical problem that brings, for example, unexpected quality loss and significant cost overruns (SEAMAN *et al.*, 2012). Software teams need to be aware of the presence of TD items in their projects and apply practices and strategies to manage them (KRUCHTEN; NORD; OZKAYA, 2012).

Current technical literature has proposed strategies for TD management to support software development teams in deciding whether, when and how to eliminate an existing TD item in the project (AMPATZOGLOU *et al.*, 2015; LI; AVGERIOU; LIANG, 2015; BEHUTIYE *et al.*, 2017; RIOS; MENDONÇA; SPINOLA, 2018). This chapter presents an overview of TD and its management. Section 2.1 explains the TD concept. Section 2.2 discusses TD management, providing details about TD prevention, monitoring, and payment activities that are within the scope of this dissertation. Section 2.3 also presents some related work on these activities. Lastly, Section 2.4 presents the concluding remarks of this chapter.

2.1 TECHNICAL DEBT

TD is related to the effects of immature artifacts in software development. These effects can be positive, as they benefit projects in the short term, but they may have to be adjusted with interest later (SPINOLA *et al.*, 2013; ZAZWORKA *et al.*, 2013). On the one hand, these benefits enable greater productivity and lower costs. On the other hand, interest can be seen as unexpected delays in system evolution activities, making it difficult to reach the defined quality criteria.

Cunningham (1992) coined the TD term when he introduced the metaphor "going into debt," indicating that a small debt could speed up software development in the short term. Still, every extra minute spent on poorly built code counts as interest on that debt. Although the term is recent, its concept is related to the notion of systems decay (LEHMAN; BELADY, 1985) and software aging (PARNAS, 1994). Furthermore, TD is associated with software quality, software effort estimation, and risk management (SEAMAN; GUO, 2011).

The technical literature has reported the following types of debt (ALVES *et al.*, 2016):

- Architectural debt: refers to problems found in the software architecture, for example, modularity violation, which can affect the architectural requirements (e.g., performance and robustness, among others) of the project. Typically, this type of debt cannot be repaid with simple code interventions, implying broader development activities.

- Build debt: refers to issues that make the build task more difficult and unnecessarily time-consuming. The build process may contain code that has no added value for the customer. Also, if the build process needs to run with poorly defined dependencies, the process becomes unnecessarily slow. When this occurs, it is possible to identify build debt items.

- Code debt: refers to problems found in the source code that can negatively affect the readability of the code, making it challenging to maintain. This debt can usually be identified by examining the source code for issues related to bad coding practices.

- Defect debt: refers to defects that are known but, due to competing priorities and limited resources, have their adjustments postponed. Decisions to defer defect handling can accumulate a significant amount of TD in a product, making it more difficult to fix later.

- Design debt: refers to debt that can be discovered by analyzing the source code and identifying violations of sound programming principles (e.g., very large or tightly coupled classes in object-oriented systems).

- Documentation debt: refers to problems found in the documentation of software projects and can be identified by looking for non-existent, inadequate, or incomplete documentation.

- Infrastructure debt: refers to infrastructure issues that may delay or impede some development activities. Examples of this type of debt are delays in adjusting or upgrading infrastructure.

- People debt: refers to issues related to people who may delay or prevent the performance of some development activities. An example of this type of debt is knowledge concentrated in a few people because of training and/or late hiring.

- Process debt: refers to inefficient processes. For example, the defined process may no longer be the most appropriate for the organization's current activities.
- Requirements debt: refers to decisions made regarding which requirements the development team needs to implement or how to implement them. Some examples of this type of debt are requirements that are only partially implemented, requirements that are implemented but only some cases, and requirements that are implemented but in a way that only partially satisfies all non-functional requirements.
- Service debt: refers to the inappropriate selection and replacement of web services that lead to incompatibility between service characteristics and application requirements. This type of debt is relevant for systems based on service-oriented architectures.
- Test automation debt: refers to the work that has yet to be performed in automating tests of previously developed features to support continuous integration and faster development cycles. This debt can be considered a subtype of the test debt.
- Test debt: Refers to issues found in testing activities that can affect the quality of those activities. Examples of this type of debt are anticipated tests that were not performed or known deficiencies in the test suite (e.g., low test coverage).
- Usability debt: refers to inappropriate usability decisions that must be adjusted later. Examples of this debt are the lack of usability standards and inconsistency between the navigation aspects of the software.
- Versioning debt: refers to source code versioning problems, such as the unnecessary use of forks.

2.2 TECHNICAL DEBT MANAGEMENT

TD management balances short-term and long-term goals, supporting software teams in deciding the need to eliminate a debt item and the most appropriate time to do it (GUO; SPINOLA; SEAMAN, 2016). If well-managed, the cost of TD is kept visible and under control, helping the project achieve its goals sooner or more cheaply. If unmanaged, TD items can cause financial and technical problems, compromising the projects future (RIOS *et al.*, 2020).

A set of activities composes TD management (LI; AVGERIOU; LIANG, 2015; RIOS; MENDONÇA; SPINOLA, 2018). These activities are described below:

- Communication: it makes the TD items visible to stakeholders, supporting the discussion of these items and their further management.
- Documentation: it provides a way to represent the identified TD items.
- Identification: it detects TD items in a software project by performing a specific technique, such as static code analysis.

- Measurement: it quantifies the cost and benefit of TD items through estimation techniques or estimates the level of all TD items existing in a system.
- Monitoring: through this activity, software teams can observe changes in the cost and benefit of debt items that still need to be eliminated during the evolution of the project.
- Payment: it refers to activities carried out to support decision-making on the most appropriate time to pay debt items.
- Prevention: it enables software development teams to prevent TD items from occurring in the project.
- Prioritization: it ranks the TD items according to an ordering criterion to define which items should be paid off first.
- Scenario analysis: it refers to a scenario analysis with TD results to clarify the different possible decisions to be made.
- Time-to-market analysis: it considers the time needed to implement the decisions when managing technical debt.
- Visualization: it provides mechanisms to present and visualize how TD items impact the system.

Figure 2.1 summarizes the tools and strategies reported in the literature for TD management activities (RIOS; MENDONÇA; SPINOLA, 2018). By analyzing the figure, we notice that prevention, payment, visualization, time-to-market analysis, and scenario analysis activities have few or no strategies or tools to support these activities. On the other hand, identification, monitoring, and measurement activities have several initiatives to help software teams to execute these activities.

Rios, Mendonça and Spinola (2018) also reported that tools and strategies existing in the technical literature have been proposed independently of each other and focused on dealing with just one TD management activity. Then, using these tools and practices can make their practical application difficult.

Our work will focus on TD prevention, monitoring and payment, because they are key, complementary, activities.

TD prevention allows software teams to avoid TD items. It is fair to expect that TD prevention can many times be “cheaper” than its payment. Moreover, prevention may also help other TD management activities. For example, setting up prevention practices helps catch inexperienced developers’ ’not-so-good’ solutions (YLI-HUUMO; MAGLYAS; SMOLANDER, 2016). A preventive action is an intentional activity, aligned with the project management plan that ensures the future performance of the project work (PMI, 2017). When applied to TD, preventive actions can support the development team in using good practices that minimize the occurrence of debt. The following practices are instances of TD preventive actions: use of *guidelines*, use of *coding standards*, *code revisions*, *retrospective meetings*, and use of *definition of done* lists (APA *et al.*, 2020b).

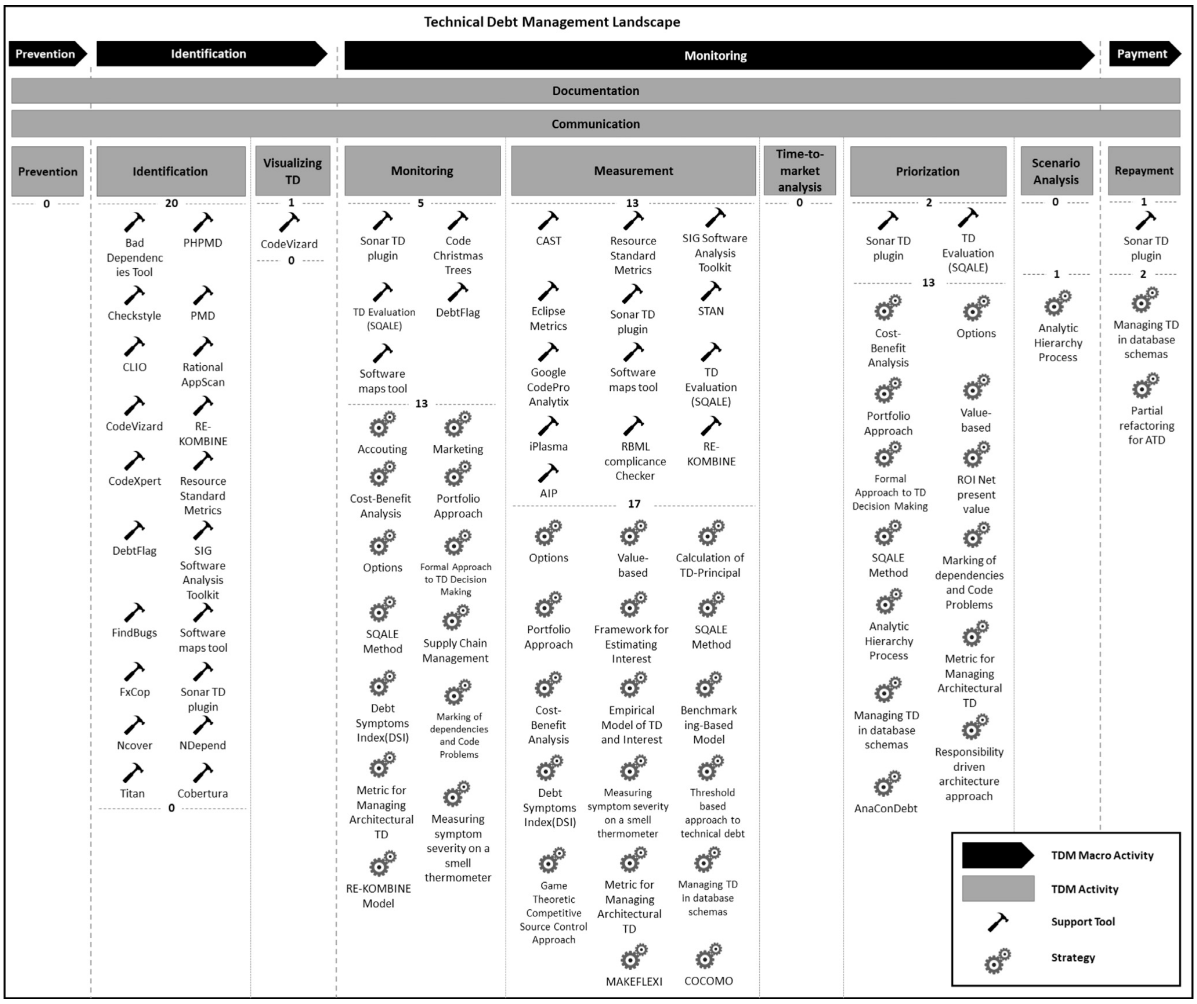


Figure 2.1 Activities, strategies, and tools for TD management (RIOS; MENDONÇA; SPINOLA, 2018)

TD monitoring is a central activity that allows software teams to track unresolved debt items, identifying changes in their cost and benefit during the project life cycle (RIOS; MENDONÇA; SPINOLA, 2018). *Cost/benefit analysis*, *real options analysis*, and *portfolio management* are the most popular approaches for monitoring the debt (AMPATZOGLOU *et al.*, 2015). Through *cost/benefit analysis*, software teams can analyze their TD items evolution, comparing cost and benefit to decide on their repayment (AMPATZOGLOU *et al.*, 2015). Performing *real options analysis*, practitioners can quantify the long-term value associated with a TD item, track it during the project, and decide how to deal with it (ALZAGHOUL; BAHSOON, 2014). Lastly, *portfolio management* allows, in an iterative way, TD items to be assessed and tracked to decide if one of them should be repaid (AMPATZOGLOU *et al.*, 2015).

TD payment refers to the activity of expending maintenance effort and resources to make up for the effects of previous decisions to incur technical debt (RIOS; MEN-

DONÇA; SPINOLA, 2018). For example, a TD item that was incurred because shortcuts were taken during testing can be paid by carrying out the testing that was previously skipped. TD payment also refers to the strategies for supporting decision-making about the most appropriate time to pay debt items off and the practices to pay off debt items (RIOS; MENDONÇA; SPINOLA, 2018). For example, by using the *portfolio management strategy* (GUO; SEAMAN, 2011; SEAMAN *et al.*, 2012), TD items are organized as assets and populate the portfolio of the software development organization. Each item is analyzed in an iterative way to decide whether this item will or will not be paid. The decision is based on risk assessment. Once a decision is taken, a payment practice (e.g., code refactoring, design refactoring, testing) can be employed to pay off the debt item.

The following section revises the technical literature on TD prevention, monitoring, and payment in relation to their practices and Practice avoidance reasons (PARs).

2.3 RELATED WORK ON TD PREVENTION, MONITORING, AND PAYMENT

Codabux, Williams and Niu (2014) proposed a set of practices for avoiding TD items in agile software development, such as *education and training*, *pair programming*, *refactoring*, *continuous integration*, *conformance to process and standards*, *tools*, and *customer feedback*. The authors indicated the following practices that contribute to TD prevention in agile development: *bugs bashes*, *having dedicated teams whose primary focus is on reducing TD*, *having each development team dedicating one iteration during a set release period towards debt reduction*, *allowing slack*, and *reflective improvement*. The authors also proposed a set of practices for eliminating TD items in agile software development, such as *code reviews*, *automating unit tests*, and *customer feedback*.

Yli-Huumo, Maglyas and Smolander (2014) conducted a case study in one middle-size Finnish software company to identify the sources of TD and the practices and strategies for TD management. By answering the research question "What management and reduction strategies/practices are being used for technical debt?" the authors found that *refactoring*, *bug fixing days*, *code reviews*, *coding standards and guides*, and *communication structure between business management and development team* were used to prevent and repay the debt.

Ampatzoglou *et al.* (2015) conducted a systematic literature review to identify existing financial aspects in the TD context. In answering the research question: "Which financial approaches have been applied for identifying, prioritizing, repaying, and monitoring technical debt?" the authors found TD monitoring (*accounting*, *cost/benefit*, *real options*, *marketing*, and *portfolio management*) and payment (*real options*, *cost/benefit*, *portfolio management*, *value-based*, and *ROI* or *net present value*) approaches.

Ernst *et al.* (2015) performed a two-part study comprising a case study in three large companies and semi-structured interviews to learn the use of tools and techniques to manage the debt. By answering the research question: "Are there practices and tools for managing TD?" the authors identified that practitioners had applied TD monitoring *as part of risk process* or *backlog grooming*.

Li, Avgeriou and Liang (2015) performed a systematic mapping study to answer, among others, the research question: "What approaches are used in each technical debt management activity?”. The authors identified the following categories of TD prevention practices, but the practices composing each category were not discussed:

- *Development process improvement* category: it improves the development process to prevent TD.
- *Architecture decision making support* category: it evaluates the architecture designs to each one with less potential TD.
- *Lifecycle cost planning* category: it develops cost-effective plans to analyze the system to reduce potential TD items.
- *Human factors analysis* category: it cultivates a culture to minimize TD causes by human factors.

Also, the authors found the following TD monitoring practice categories:

- *Threshold-based approach* category: it defines thresholds for TD-related quality metrics and issue warnings if the thresholds are not met.
- *TD propagation tracking* category: it tracks the influences of TD through dependencies between other parts of a system and the parts of the system that contains TD.
- *Planned check* category: it regularly measures identified TD and tracks the change of the TD.
- *TD monitoring with quality attribute focus* category: it monitors the change of quality attributes detrimental to TD, such as stability.
- *TD plot* category: it plots various aggregated measures of TD over time and looks at the shape of the curve to observe the trends.

Lastly, they identified the following TD payment categories:

- *Refactoring*: it changes the code, design, or architecture of a software system without changing the external behaviors of the software system to improve its internal quality.
- *Rewriting*: it rewrites the code affected by the debt.
- *Automation*: it automates work performed manually, such as manual tests, builds and deployment.
- *Reengineering*: it evolves existing software to provide new behaviors, features, and operational quality.
- *Repackaging*: it simplifies the code by grouping cohesive modules with manageable dependencies.
- *Bug fixing*: it resolves bugs identified in the software.

- *Fault tolerance*: it includes strategically placing runtime exceptions where the TD is.

Oliveira, Goldman and Santos (2015) performed action research on two companies using Scrum to evaluate the application of the TD management framework proposed by Seaman and Guo (2011). This framework encompasses the following steps: (i) to identify the list of TD items, (ii) to measure the effort for eliminating the items, and (iii) to monitor the items to decide the most appropriate time to deal with them. The authors identified *defining a responsible person for monitoring each identified and measured TD item* were TD monitoring practices used by both companies.

In another work, Abad *et al.* (2016) conducted in-depth interviews with stakeholders about development, testing, and product management teams and a quantitative survey on effort estimation, quality management processes, and understanding of the risks and benefits of TD. By answering the research question: "What suggestions do testers and developers have for reducing TD in their projects?" the authors found the following TD payment practices: *allocating more resources (time, budget, and infrastructure)*, *technical solutions (e.g., refactoring, using design patterns, and improving design and architecture)*, *prioritizing TD*, *having more flexible schedules*, and *quantifying TD*.

Gupta *et al.* (2016) conducted a case study on managing TD in a legacy system. The authors evidenced the following prevention practices: *following boy scout rule*, *continuous improvement*, *technical debt awareness*, *improved definition of done*, and *continuous improvement in functional test automation*; and the following practices used to pay off the debt: *continuous identification, prioritization, and resolving identified technical debt*, *visualizing debt with information radiators*, *continuous collaboration with product owner*, *internal debt stories*, and *common product backlog*.

Yli-Huumo, Maglyas and Smolander (2016) identified the main TD management activities performed in a large software company. When answering the research question: "What methods, models, practices, or tools do the studied development teams use for each TD management activity?" the authors identified that TD prevention practices were associated with *coding standards*, *code reviews*, *reviews of the used tools*, *definition of done to ensure code quality*, and *definition of the done standard*. The authors found the following TD prevention practices: *coding reviews*, *education and training*, *pair programming*, *test-driven development*, *refactoring*, *continuous integration*, *conformance to process and standards*, *tools*, and *customer feedback*. The authors also found that TD monitoring was conducted rarely and *used data collected from (management or TD measuring) tools*. Lastly, the authors identified that software practitioners used *refactoring*, *rewriting*, or *redesigning* to pay off TD items. Also, the authors recognized that TD payment was performed in normal development by including TD items in the actual development backlog.

In another work, Behutiye *et al.* (2017) ran a systematic literature review to investigate the state of the art of TD, its causes, consequences, and management strategies in the context of agile software development. By answering the research question "What are the strategies proposed in the literature to manage TD in agile software development?" the authors identified the following TD monitoring practices: *collective dashboards*, *vi-*

sualization techniques, *continuous integration tools*, *setting a commonly agreed definition of done*, *improving estimation techniques of sprints*, *planning (in advance) for TD*, and *implementing pair programming or test-driven development.* Moreover, the authors identified the following TD payment categories: *refactoring*, *code analysis*, and *test automation.*

By interviewing six agile teams from four software companies, Bomfim and Santos (2017) sought to identify strategies and practices for eliminating the debt. The authors identified that *using coding standards* was used to prevent TD items. The authors also identified that *including TD tasks in product backlog* was the strategy for monitoring TD items by interviewing six agile teams from four software companies. They also reported the following TD payment practices: *splitting methods to make them more reusable*, *using design patterns*, *refactoring older code*, and *using palliative solutions.* Regarding the PARs for TD non-payment, the authors investigated the factors that influence TD payment in agile software projects, reporting that *concern of impacting some module because the team does not know all parts of the code to carry out a deeper impact analysis*, *lack of test coverage or excessive manual testing*, *low impact for business*, and *high effort* discourage the application of payment practices.

In the same year, Samarthyam, Muralidharan and Anna (2017) provided an overview of test debt, discussing the causes of incurring test debt items, strategies for managing them, and two case studies on managing them in real-world projects. The authors indicated the following prevention practices: *increasing awareness in the development and test teams on test debt* and *introducing relevant processes can also help stop accumulation of debt.* Furthermore, the authors indicated the payment practices: *pair programming*, *clean coding*, and *refactoring.*

Charalampidou *et al.* (2018) defined a tool to support the prevention of requirements documentation debt items. The tool makes it possible to integrate requirements specifications into a development Integrated development environment (IDE). Through a case study, the authors evaluated the tool, finding that its main benefit was to motivate developers to define, maintain, and use requirements specifications and track them as part of their daily routine.

Martini (2018) proposed a management tool, *AnaConDebt*, to support tracking and evaluating TD items. The tool allows *the creation of TD items in a backlog* for monitoring them. Among its features, *AnaConDebt* makes it possible to track TD items in a dedicated repository.

Martini, Besker and Bosch (2018) performed a multi-method research comprising a survey and case study of three companies to understand practitioners' efforts to manage TD. By answering the research questions: "What tools are used to track TD?," "How do software organizations introduce a TD tracking process?," and "What are the initial benefits and challenges when large organizations start tracking TD?," the authors recognized that surveyed practitioners had used tools for supporting the following TD monitoring practices: *using comments in the code or other artifacts*, *documenting issues in text or spreadsheets*, *using a system for bug fixing*, *reporting TD items in the backlog*, *statically analyzing the code for finding TD items or potential bugs, or security issues, and measuring test coverage.*

Rios, Mendonça and Spinola (2018) conducted a tertiary study to investigate the current state of the art on TD and its management. Through the research question "What are the activities, strategies and tools that have been proposed to support the management of TD?" the authors identified the following practices for TD monitoring: *accounting*, *cost-benefit analysis*, *options*, *Software quality assessment based on lifecycle expectations (SQALE) method*, *debt symptoms index*, *metrics for managing architectural TD*, *RE-KOMBINE model*, *measuring symptom severity on a smell thermometer*, *making of dependencies and code problems*, *supply chain management*, *formal approach to TD decision making*, *portfolio approach*, and *marketing.* Moreover, the authors identified the following strategies for TD payment: *managing TD in database schemas* and *partial refactoring for architectural TD.*

From existing literature, Lenarduzzi and Fucci (2019) presented a holistic definition of requirements debt. The authors then proposed the following TD payment strategies for requirements debt items: *once the neglected users' need is identified, it is formalized and included in the software requirements specification document, as for code smells, refactoring is needs to be applied to pay back requirements smells*, and *the implementation of the best new solution matching the updated software requirements specification document.*

In another related work, Lenarduzzi *et al.* (2019) performed a focus group with five practitioners to identify the most common types of TD faced in the past, their causes, and the ways to prevent them. By answering the research question "How to mitigate TD?" the authors identified the following prevention practices: *learning from customers*, *careful estimation*, and *continuous improvement.*

Mendes *et al.* (2019) developed and assessed a tool, *VisminerTD*, to support TD identification and monitoring activities. The tool presents the set of TD items distributed in panels and uses a *Kanban-based* approach to monitor them.

Toledo *et al.* (2019) conducted an exploratory case study in a real-life project to investigate the relationship between architecture TD and microservices. The authors identified the following TD payment practices by answering the research question "What is a solution for the identified architecture TD in microservices and its associated refactoring cost (principal)?": *rewrite the communication layer*, *migrate the services to use the new architecture*, *remove the business logic inside the communication layer*, *move the business logic to the services*, *define a canonical model per domain*, *update the services to use the newly defined canonical models*, *centralize the source code and documentation for all services in a common management system*, *provide a common middleware that can be used by all services*, *rewrite services to use the same middleware*, *define and execute a governance plan to handle the migration*, *maintain the system working with different solutions during the migration period*, *new requirements should be down prioritized*, and *service developers must learn new technologies.*

Silva, Junior and Travassos (2019) ran a survey to understand the perception of TD and its management in the Brazilian software industry. Based on 40 complete answers, the authors characterized TD awareness and perception and the strategies used in TD management. By answering the research question: "Which technologies and strategies are adopted for each TD managament activity?" the authors identified that *guidelines*, *coding standards*, *code revisions*, *retrospective meetings*, and *Definition of Done* were the TD prevention practices used. The authors did not find any practice used to monitor the debt.

And, the practices *refactoring*, *redesign*, *code rewriting*, and *meetings/workshops/training* were used to pay off the debt.

Two replications of the Silva, Junior and Travassos (2019) survey were performed in the Uruguayan industry, obtaining 259 answers on TD in general (APA *et al.*, 2020b) and 33 answers from practitioners working in software startups (APA *et al.*, 2020a). These replications identified the same preventive practices as Silva, Junior and Travassos (2019). Regarding TD monitoring, Apa *et al.* (2020b) identified that *using tools (SonarQube)* supported TD monitoring, and Apa *et al.* (2020a) found that *manual monitoring* and *tools (Jira and Wiki)* were also used to monitor TD items. Regarding TD payment, Apa *et al.* (2020a) and Apa *et al.* (2020b) recognized the same TD payment practices: *refactoring*, *rewriting code*, and *redesign*.

Rios *et al.* (2020) conducted two complementary studies on the causes, effects, and preventive and payment actions associated with documentation debt. Through the research question "How can development teams react to the presence of documentation debt?" the authors identified the following prevention practices: *comment the code*, *create tutorials on how to fill in the documentation*, *define process and good practices for documentation*, *define roles concerning the documentation process*, *document the project since its begin*, *have a documentation repository*, *improve commitment of the team concerning documentation*, *involve several roles in documenting the project*, *penalties if not follow the documentation process*, *training on the problems by do not document*, *use of peer review*, and *use of Unified modeling language (UML) to document and share information.* Moreover, the authors identified that to *adopt TD payment prioritization criteria*, *keep the documentation updated*, and *review outdated documentation* were the TD payment practices used by the participants studies.

Aragão *et al.* (2022) defined a catalog to support the management of test debt items. As preventive practices, the authors included *present already identified debts* and *review elaborated test cases.* As monitoring practices, the catalog has *monitor changes in the cost/benefit ratio of the identified debt*, *monitor triggers* and *changes in the test process (if the team has an existing test process).* Lastly, *elaborate and perform tests for releases that were not tested* and *change test cases by analyzing defects* were the practices included in the catalog to pay off test debt items.

Bogner, Verdecchia and Gerostathopoulos (2021) performed a systematic mapping study to investigate TD in the context of artificial intelligence-based systems. By answering the research question: "Which solutions have been reported to address technical debt and antipatterns in AI-based systems?" the authors identified 46 payment practices: *to manage model configuration*, *to use clear component and code APIs*, *to remove unnecessary features*, *refactor the code*, and *to monitor deployed models.*

In a systematic mapping study on architecture and code debt, Das *et al.* (2022) defined the following research question "What are the common ways to manage technical debt." By answering it, the authors identified the following payment practices: *strategic management*, *getting a better understanding of technical debt*, and *using automated tools.*

Ernst, Delange and Kazman (2021) discussed technical debt in practice per types of debt and reported insights collected from case studies. The authors suggested a set of prevention practices for each type of debt. For requirements debt, *do a good job of re-*

quirements elicitation, and *leverage crowd-sourcing techniques for gathering requirements.* For design debt, *employ a design method such as attribute-driven design.* For code debt: *choose your language and libraries wisely*, *efficient code reviews*, and *gather and analyze metrics on the code base.* For test debt, *adopt test-driven development*, *maintain and analyze test runs*, *automate testing activities*, and *avoid manual testing.* For deployment debt [1], *automated gradual deployments*, *integrate deployment with your continuous integration pipeline*, *define a deployment process*, and *implement kill switches on new features.* For documentation debt, *traceability*, *check quality of the documentation as part of the release process*, and *documenting design and technical debt.* For machine learning debt [2], *focus on how to properly design machine learning systems* and *evaluate the differences in performance and accuracy between the model being used and state-of-the-art models for which benchmarks are available.* The authors also identified payment practices for architecture (*refactoring*), design (*fixing the antipatterns*), and social [3] debt, such as: *to improve coordination of decisions*, *living documents*, and *coaching.* In total, they reported 42 practices.

de Toledo, Martini and Sjøberg (2021) performed an exploratory multiple-case study by interviewing 22 employees from seven companies. The authors identified 31 payment practices for eliminating architecture debt items, for example: *to add services ownership metadata to the messages*, *allowing identification of their source*, *implementation of a Canonical Data Model that ensure compliance*, *removal of the dead letter queue and to move the responsibility of the message deliveries to the endpoints*, *add metadata to identify the source of the messages*, *splitting the dead letter queue into smaller queues, managed by different teams*, *use some time to design generic and independent services*, *internal training about API development*, *use of an API-first approach while designing services*, *considering slot for continuous API improvement during development*, and *ensure standardization with a Canonical Data Model.*

Wiese, Riebisch and Schwarze (2021) evaluated a framework that enables the awareness of the team that is incurring TD and the integration of TD management with project management. The framework allows debt-related activities to be recorded as tickets and included in the project backlog. Through the research question "Can TD be prevented by the adoption of the framework?" the authors find that the use of the framework allowed the project team to have more rational discussions and decisions about the TD items, thus enabling the prevention of these items.

Albuquerque *et al.* (2022) performed a survey to analyze the broader concept of TD and its management. By considering 120 answers collected from the Brazilian, Canadian, Indian, North American, and Swiss software industries, the authors responded to the research question, "What solutions are adopted for each technical debt management

[1]Deployment debt refers to "all the shortcuts, errors, or mistakes that happen when deploying and operating a system" (ERNST; DELANGE; KAZMAN, 2021).

[2]Machine learning debt is "an issue in these systems as well, and there are some forms of debt that are specific to machine learning systems" (ERNST; DELANGE; KAZMAN, 2021).

[3]Social debt is "the mismatch between system structure and organizational structure. Furthermore, it stands to reason that some mappings of organizational designs onto system designs will also be suboptimal" (ERNST; DELANGE; KAZMAN, 2021).

activity?". As a result, the authors found the following prevention practices: *code reviews*, *coding standards*, *automated tests*, *guidelines*, and *meetings*. The authors also identified the following monitoring practices: *manually*, *static analysis*, *issue tracker*, *test tool*, and *Wiki*. The payment practices reported by the survey's participants were *refactoring*, *code rewriting*, *architectural redesign*, *meetings*, and *artifact change*.

Bonfim and Benitti (2022) performed an interviewed-case study with 19 agile software practitioners from 13 organizations in Brazil, France, Portugal, United Arab Emirates, and Belgium. By answering the research question "What practices do agile organizations employ that can minimize requirements debt?" the authors identified the following TD prevention categories: *meeting requirements elicitation*, *helping requirements analysis*, *supporting the implementation of requirements specification*, *implementing requirements validation*, and *assisting in requirements management*. The authors also identified 15 TD prevention practices. For example, *recording all requirements, demands, and stories in the backlog*, *defining and validating what will be prioritized*, and *managing the requirements, deliveries, backlog delays in progress*. Lastly, the authors identified the TD payment category called *reducing requirements debts* with the following TD payment practices: *assessing impact of TD* and *usually pays debt in the next sprint*.

Lastly, Fu *et al.* (2022) conducted an exploratory study considering 2,030 review comments from the Nova project of OpenStack and the Qt Base project of Qt to investigate which potential technical debt is identified in code reviews. By answering the research question "What practices are used by developers to resolve potential technical debt that has been identified in code reviews?" the authors identified that *code refactoring*, *documentation improvement*, *testing improvement*, *bug fixing*, and *code change abandonment* as TD payment practices.

2.3.1 Discussion

Our work aims to holistically investigate the practices for TD prevention, monitoring, and payment and PARs for TD non-prevention, non-monitoring, and non-payment from the perspective of software practitioners.

Through the analysis of the technical literature, we found evidence of the practices used to prevent, monitor, and pay off debt items. Table 2.1 summarizes relevant information about related work, reporting whether practices or categories of these practices were found, the types of debts considered, the representativeness of each study (sample size and number of organizations), and the research method used. We can see that most studies have a quite small sample size, except the studies conducted by Ernst *et al.* (2015), Martini, Besker and Bosch (2018), and Apa *et al.* (2020b). However, these studies have a limited variety of software development contexts. Having a larger and broader data set helps to unfold decisions made by practitioners in different roles, following different development models, and in different countries. Our approach is more likely to generalize the results to broader real-world scenarios. This increases ecological validity (ANDRADE, 2018), as well as confidence in the validity of results (WOHLIN *et al.*, 2012).

A third of the related works focused on only one type of debt (architecture, design, defect, documentation, requirements, and test debt) (OLIVEIRA; GOLDMAN; SAN-

TOS, 2015; SAMARTHYAM; MURALIDHARAN; ANNA, 2017; CHARALAMPIDOU *et al.*, 2018; LENARDUZZI; FUCCI, 2019; TOLEDO *et al.*, 2019; RIOS *et al.*, 2020; ARAGÃO *et al.*, 2022; DAS *et al.*, 2022; de Toledo; MARTINI; SJøBERG, 2021; BONFIM; BENITTI, 2022), while the technical literature (RIOS; MENDONÇA; SPINOLA, 2018) reported 15 different types of TD. Although having practices for specific types of debt can aid software practitioners in defining specific support to certain TD management challenges, analyzing only one type of debt can reduce the findings on TD management. Moreover, knowing the practices from the perspective of a wide variety of software practitioners is critical to guide research directions and, also, to serve as a benchmark to better position the TD management actions in software organizations.

Only the studies conducted by Yli-Huumo, Maglyas and Smolander (2014), Bomfim and Santos (2017), Silva, Junior and Travassos (2019), Apa *et al.* (2020a), Apa *et al.* (2020b), Aragão *et al.* (2022), Albuquerque *et al.* (2022) focused on TD prevention, monitoring, and payment activities simultaneously. And, only Aragão *et al.* (2022) organized the practices into a catalog to make their findings more feasible to use in practice. However, their catalog only addresses test debt. Moreover, by using the results from the other studies, development teams rely on textual information spread through several tables, thus hindering the use of current knowledge on TD management.

Regarding the PARs for TD non-prevention, non-monitoring, and non-payment, to the best of our knowledge, only PARs for TD non-payment have been investigated in the technical literature (BOMFIM; SANTOS, 2017). These PARs for TD non-prevention, non-monitoring, and non-payment merit further study, as the potential impediments to prevention, monitoring, and payment strategies (or to prevention, monitoring, and payment in general) are essential factors for a team to understand when designing their own TD prevention, monitoring, and payment strategy. Choosing prevention, monitoring, and payment practices without understanding and planning for the inherent risks to TD prevention, monitoring, and payment is likely to result in failure to execute the plan.

Overall, the findings presented in the related work reveal the point of view of a small number of practitioners and organizations. Chapters 4, 5, and 6 will triangulate their findings with our results and discuss how they complement each other.

Table 2.1: Overview of the limitations of related work on TD prevention, monitoring, and payment

Related work	**TD prevention**		**TD monitoring**		**TD payment**		**Type of debt**	**Representativeness**		**Research method**
	Practice	Category	Practice	Category	Practice	Category		Sample size	# organizations	
(CODABUX; WILLIAMS; NIU, 2014)	Yes	No	No	No	Yes	No	General	-	-	Overview
(YLI-HUUMO; MAGLYAS; SMOLANDER, 2014)	Yes	No	No	No	Yes	No	General	11	1	Case study
(AMPATZOGLOU *et al.*, 2015)	No	No	Yes	No	Yes	No	General	-	-	Systematic review
(ERNST *et al.*, 2015)	No	No	Yes	No	No	No	General	536	3	Case study
(LI; AVGERIOU; LIANG, 2015)	No	Yes	No	Yes	No	Yes	General	-	-	Systematic review
(OLIVEIRA; GOLDMAN; SANTOS, 2015)	No	No	Yes	No	No	No	Design or defect	16	2	Action research
(ABAD *et al.*, 2016)	No	No	No	No	Yes	No	General	48	1	Case study
(GUPTA *et al.*, 2016)	Yes	No	No	No	Yes	No	General	-	1	Case study
(YLI-HUUMO; MAGLYAS; SMOLANDER, 2016)	Yes	No	Yes	No	Yes	No	General	25	1	Case study
(BEHUTIYE *et al.*, 2017)	No	No	Yes	No	No	Yes	General	-	-	Systematic review
(BOMFIM; SANTOS, 2017)	Yes	No	Yes	No	Yes	No	General	6	4	Case study
(SAMARTHYAM; MURALIDHARAN; ANNA, 2017)	Yes	No	No	No	Yes	No	Test	-	-	Overview
(CHARALAMPIDOU *et al.*, 2018)	Yes	No	No	No	No	No	Requirement documentation	-	-	Tool development
(MARTINI, 2018)	No	No	Yes	No	No	No	General	-	-	Tool development
(MARTINI; BESKER; BOSCH, 2018)	No	No	Yes	No	No	No	General	226	3	Case study
(RIOS; MENDONÇA; SPINOLA, 2018)	No	No	Yes	No	Yes	No	General	-	-	Systematic review
(LENARDUZZI; FUCCI, 2019)	No	No	No	No	Yes	No	Requirement	-	-	Literature review
(LENARDUZZI *et al.*, 2019)	Yes	No	No	No	No	No	General	5	-	Focus group
(MENDES *et al.*, 2019)	No	No	Yes	No	No	No	General	28	-	Case study
(TOLEDO *et al.*, 2019)	No	No	No	No	Yes	No	Architecture	1	1	Case study
(SILVA; JUNIOR; TRAVASSOS, 2019)	Yes	No	No	No	Yes	No	General	40	-	Survey
(APA *et al.*, 2020b)	Yes	No	Yes	No	Yes	No	General	259	-	Survey
(APA *et al.*, 2020a)	Yes	No	Yes	No	Yes	No	General	33	-	Survey
(RIOS *et al.*, 2020)	Yes	No	No	No	Yes	No	Documentation	4	1	Case study

(*table continues*)

Table 2.1: Overview of the limitations of related work on TD prevention, monitoring, and payment (continued)

Related work	**TD prevention**		**TD monitoring**		**TD payment**		**Type of debt**	**Representativeness**		**Research method**
	Practice	Category	Practice	Category	Practice	Category		Sample size	# organizations	
(ARAGÃO *et al.*, 2022)	Yes	No	Yes	No	Yes	No	Test	5	1	Case study
(BOGNER; VERDECCHIA; GEROSTATHOPOULOS, 2021)	No	No	No	No	Yes	No	General	-	-	Systematic review
(DAS *et al.*, 2022)	No	No	No	No	Yes	No	Architecture	-	-	Systematic review
(ERNST; DELANGE; KAZMAN, 2021)	Yes	No	No	No	Yes	No	Requirements, design, architecture, code, test, deployment, documentation, machine learning, social	5	4	Case study
(de Toledo; MARTINI; SJøBERG, 2021)	No	No	No	No	Yes	No	Architecture	22	7	Case study
(WIESE; RIEBISCH; SCHWARZE, 2021)	Yes	No	No	No	No	No	General	22	1	Case study
(ALBUQUERQUE *et al.*, 2022)	Yes	No	Yes	No	Yes	No	General	120	-	Survey
(BONFIM; BENITTI, 2022)	Yes	Yes	No	No	Yes	Yes	Requirements	19	13	Case study
(FU *et al.*, 2022)	No	No	No	No	Yes	No	General	2,030	-	Exploratory study

2.4 FINAL REMARKS

This chapter introduces the definitions and activities of TD that are part of the dissertation background. These concepts are the basis for understanding the other chapters of this dissertation.

Although many studies have addressed the practices used to prevent, monitor, and pay off TD items in software projects, the discussion about these practices is limited, as they usually represent the point of view of a small number of software practitioners and organizations. Therefore, a deeper investigation into practices related to TD prevention, monitoring, and payment is necessary to understand how software practitioners have dealt with TD items in a more comprehensive way. In addition, only one study has investigated the PARs associated with TD non-payment, while PARs for TD non-prevention and non-monitoring have yet to be investigated. This indicates the need for a more in-depth investigation to identify these PARs, and to understand better how to help software teams to implement TD management practices in their projects.

The next chapter presents the InsighTD project (RIOS *et al.*, 2018; RIOS *et al.*, 2020) and the research method we applied to analyze the InsighTD data on TD prevention, monitoring, and payment, focusing on the practices used in these activities and the PARs that justify the non-application of these practices.

Chapter

3

This chapter presents the research method we followed to achieve the dissertations goal. Initially, it shows the research questions and the InsighTD Project, giving details on the project and the data collection and analysis. Next, the chapter discusses the threats to project validity and presents the InsighTD data set. Lastly, it presents the concluding remarks.

RESEARCH METHOD

Surveys are conducted to take a snapshot of a determinate situation using sample questioning to understand a population (WOHLIN *et al.*, 2012). A survey can be either descriptive, explanatory, or explorative. A descriptive survey can be carried out to make claims about some characteristics or traits of a population. An explanatory survey seeks to make claims about phenomena or behaviors associated with a population. And an explorative survey is a pre-study to an in-depth investigation of a specific issue. Surveys shall also be accessible for replication, allowing the study to be repeated under similar conditions to increase confidence in the results (WOHLIN *et al.*, 2012).

A survey is a convenient research instrument for collecting responses from large samples and can help to provide broad overviews of large populations. Surveys have, however, some drawbacks. In a survey, data about a phenomenon is always collected after the fact and based on the respondents recall of it. This recollection may not be as accurate as direct measurement or observation. Also, broad surveys make it tedious for respondents to fill out their questions. Moreover, survey researchers have to frequently deal with the challenges of low response rates, resulting in lower representativeness of the population.

Families of surveys have been used to deal with some of those problems. In software engineering, families of surveys have been used to investigate phenomena such as software requirement issues (FERNÁNDEZ *et al.*, 2017) and technical debt (RIOS *et al.*, 2018; RIOS *et al.*, 2020).

Our work is based on a family of surveys named the InsighTD Project. This family of surveys aims to examine the state of software engineering Technical debt (TD) practice, revealing its causes, effects, and management practices.The InsighTD survey was designed to allow its replication in different countries. It was defined in the context of a Ph.D. dissertation (ALVES, 2020) aiming at investigating the causes and effects of TD. The project also addresses TD management (prevention, monitoring, and payment) by asking the participants whether these activities have been performed and giving details about how these activities were performed or why they were not. Our dissertation uses the data collected from the InsighTD Project to investigate the state of the practice of TD prevention, monitoring, and payment.

The rest of the chapter is organized as follows. Section 3.1 presents the research questions posed in this dissertation. Section 3.2 gives an overview of the InsighTD project. Section 3.3 discusses the data collection strategy conducted by the project. Section 3.4 presents the data analyses we followed to identify the TD management practices (and reasons to avoid them), supporting us in answering the research questions. Section 3.5 discusses the threats to the validity of our work. Section 3.6 presents the InsighTD data set. Finally, Section 3.7 offers the concluding remarks of this chapter.

3.1 RESEARCH QUESTIONS

This dissertation seeks **to investigate, through continuous and independent replication of a globally distributed family of surveys, the state of practice on TD prevention, monitoring, and payment in software projects**. For this, we defined four research questions and their sub-questions, as shown in Table 3.1.

Table 3.1 Research questions

ID	Research question (RQ)
RQ1	**How could software development teams avoid technical debt items on their projects?**
RQ1.1	What are the leading prevention practices that can be used to avoid TD?
RQ1.2	What are the practice avoidance reasons (PARs) considered by software practitioners for TD non-prevention?
RQ2	**How have software development teams monitored technical debt items on their projects?**
RQ2.1	What are the leading practices for monitoring TD items in software projects?
RQ2.2	What are the leading practice avoidance reasons (PARs) to explain the non-monitoring of TD items?
RQ3	**How have software development teams paid off technical debt items on their projects?**
RQ3.1	What are the practices used by software practitioners to deal with technical debt items in their projects?
RQ3.2	What are the practice avoidance reasons (PARs) considered by software practitioners for not paying off TD?
RQ4	**How to organize the body of knowledge composed of prevention, monitoring, and payment practices - and Practice avoidance reasons (PARs) for TD non-prevention, non-monitoring, and non-payment - to support TD management?**
RQ4.1	How are the proposed artifacts characterized in terms of its support for TD management activities?

In RQ1, RQ2, and RQ3, we intend to investigate the state of the practice of TD prevention, monitoring, and payment, respectively. By exploring it, we aim to identify the practices used by software practitioners to prevent (RQ1.1), monitor (RQ2.1), and pay off (RQ3.1) the debt, and the PARs used by software practitioners to justify the non-prevention (RQ1.2), non-monitoring (RQ2.2), and non-payment (RQ3.2) of debt items.

For answering RQ1 to RQ3 and their sub-questions, we used data collected from the InsighTD project. We address RQ1 to RQ3 and their sub-questions in Chapters 4, 5, and 6, respectively.

Lastly, RQ4 seeks to define a structure of the set of practices and PARs in an artifact to support software practitioners in their TD prevention, monitoring, and payment initiatives. This structure also allows for the organization of the body of knowledge learn from the state of practice. To answer this RQ, we updated the conceptual for TD (AVGERIOU *et al.*, 2016; IZURIETA *et al.*, 2016; RIOS; MENDONÇA; SPINOLA, 2018), proposed conceptual maps, and defined the Impediments, decision factors, enabling practices, and actions (IDEA) diagrams. We build these artifacts using the set of practices and PARs identified from RQ1-RQ3 (and their sub-questions). To evidence the artifacts support for TD management (RQ4.1), we conducted a follow-up survey considering the conceptual model and maps and performed two complementary empirical studies in academic and industrial settings considering the IDEA diagrams. We address RQ4 and its sub-question in Chapters 7 and 8.

The following subsection introduces the InsighTD project and the data collection and analyses we performed to answer RQ1, RQ2, and RQ3 and their sub-questions.

3.2 THE INSIGHTD PROJECT

The InsighTD project is a family of globally distributed surveys on the causes, effects, and management of TD (RIOS *et al.*, 2018; RIOS *et al.*, 2020). InsighTD was designed to allow the survey to be continuously replicated in different countries, seeking generalizable results on the state of practice in the TD area. In addition, the project intends to organize data on the practical problems of TD considering the different contexts of software development, that is, different development cultures, organization sizes, development methodologies, and among others.

At each replication, discoveries can be made for the area, and through the combination of data collected by different replications, possible differences or similarities can be elucidated concerning perceptions about the concept of TD, its causes and effects, and the way to manage TD items. Thus, the project provides a shared infrastructure to support replications and dissemination of results. This infrastructure consists of the same version of the questionnaire, a set of instruments to guide data analysis, and a communication plan that indicates how the results should be communicated among the project team. So far, researchers from twelve countries have participated in the project: Argentina, Brazil, Chile, Colombia, Costa Rica, Finland, India, Italy, Norway, Saudi Arabia, Serbia, and the United States. More information about the project, its replications, and the achieved results can be found at <www.td-survey.com>.

3.2.1 Aims and Planning

The purpose of the InsighTD project is to investigate the state of practice and industry trends in the field of TD, including the status quo, the causes that lead to the occurrence of TD, the effects of its existence, how these problems manifest themselves in the development process, and how software development teams react when they are aware of the presence of debt items in their projects (RIOS *et al.*, 2018; RIOS *et al.*, 2020). The following RQs were defined for the InsighTD Project (herein called InsighTD project research question (iTDp-RQ)[1]) to achieve these goals:

- **iTDp-RQ1**: Are software professionals familiar with the concept of TD?
- **iTDp-RQ2**: What causes lead software development teams to incur TD?
- **iTDp-RQ3**: What effects does TD have on software projects?
- **iTDp-RQ4**: How do software development teams react when they are aware of the presence of debt items in their projects?

Through **iTDp-RQ1**, the project seeks to identify how the concept of TD is disseminated among software development professionals and whether the concept varies according to practitioners role in the software development process. In **iTDp-RQ2** and **iTDp-RQ3**, the project aims to recognize the causes and effects of TD identified by software development professionals. Finally, in **iTDp-RQ4**, the project aims to verify whether software professionals have prevented, monitored, or paid off the existing TD items in their projects. It also seeks to identify how professionals have prevented, monitored, and paid off the debt and why they have not carried out these activities.

InsighTD was planned in cooperation with several researchers in the TD field, composing the core or replication teams. The first is responsible for leading the project activities and conducting the survey in Brazil. The latter is responsible for using the same infrastructure shared by the core team to conduct the survey replications. InsighTDs design comprises the following steps (RIOS *et al.*, 2020):

- **Conception**: at this stage, the InsighTD core team defined the experimental package, containing the RQs, the planning of the survey family, the research instruments, the target audience, and the initial discussion on data analysis. Also, the team defined the set of researchers composing each replication team.
- **Validation**: this step consisted of validating the experimental package generated in the conception stage. Each validation was performed individually and incrementally, ensuring that the next researcher would only start their review of the experimental package when the reviewing researcher had finished their task. Initially, the experimental package underwent four internal reviews and one external review, to ensure that the survey questions were easy to interpret and complete to respond to the RQs defined for the project. The internal reviewers were members of the InsighTD project who did not participate in the design stage, while the external reviewer was not a project member. Finally, a pilot study was conducted to observe whether the survey was well understood by a small number of professionals who represented the target population of the study, that is, professionals who

[1]The InsighTD project RQs are not to be confused with the dissertation RQs discussed in Section 3.1. For this reason, we use the acronym iTDp-RQ to refer to them.

played different roles in software development. The core team analyzed and applied adjustments to the project's artifacts at each validation.

- **Initiation**: at this stage, the survey was carried out in Brazil and replicated in the United States. Initially, the channels for inviting participants were defined. These channels were LinkedIn, industry-affiliated member groups, mailing lists, and industry partners. Then, participants from the Brazilian industry were invited. Upon receipt of the responses, they were analyzed, resulting in a baseline report containing the initial findings associated with the project's RQs. Then, the survey was replicated in the North American software industry, making it possible to synthesize the results obtained in the application of the survey in the Brazilian and American software industries.
- **International replication**: comprises the survey replication in other countries independently and using the same experimental package. With this, the project seeks more generalizable data on the state of TD practice, synthesizing data from different replications.

The following section presents the InsighTD survey, explaining its questions and their relationship to the project's RQs.

3.2.2 Questionnaire

The InsighTD questionnaire comprises 28 questions that aim to help answer the project's RQs and characterize the participants and their work environments (RIOS *et al.*, 2020). Table 3.2 presents a simplified version of the questionnaire, with the Question (Q) description, its related iTDp-RQs, its identifier, its description, and its type (closed or open). Appendix A presents the questionnaire in more detail.

Questions Q1-Q8 capture the characterization of participants and their working environment concerning the companys size they work, the country they work, the systems size (in lines of code - LOC), the systems age, the teams size, the role they play, the experience in that role, and the development process used in the project (agile, hybrid, or traditional).

Questions Q9-Q15 seek evidence to answer RQ1. Initially, participants indicate how familiar they are with the concept of TD (Q9), choosing one of the following options: "Never heard of it," "I have read about it in books/articles," "I have been on projects where I recognized TD, but the project did not explicitly manage it," and "I have been on projects where we attempted to actively manage TD." Then, the participants give their definition of TD in Q10. Then, the definition[2] of TD adapted from McConnell (2007) is presented, and the participant indicates how close it is to their understanding

[2]The TD definition used in the InsighTD survey was adapted from McConnell (2007): "Technical debt contextualizes the problem of outstanding software development tasks (for example, tests planned but not executed, pending code refactoring, pending documentation update, use of bad design practices, code that does not exhibit good coding practices) as a kind of debt that brings a short-term benefit to the project (normally in terms of higher productivity or shorter release time of software versions), that may have to be paid later in the development process with interest (for example, a poorly designed class

Table 3.2 The InsighTD survey questions (adapted from Rios *et al.* (2020)). The questions in bold are used in this dissertation.

RQ	ID	Question (Q)	Type
	Q1	**What is the size of your company?**	**Closed**
	Q2	**In which country you are currently working?**	**Closed**
	Q3	**What is the size of the system being developed in that project? (LOC)**	**Closed**
	Q4	**What is the total number of people of this project?**	**Closed**
	Q5	**What is the age of this system up to now or to when your involvement ended?**	**Closed**
	Q6	**To which project role are you assigned in this project?**	**Closed**
	Q7	**How do you rate your experience in this role (at the time)?**	**Closed**
	Q8	**Which of the following most closely describes the development process model you follow on this project?**	**Closed**
iTDp-RQ1	Q9	How familiar you are with the concept of TD?	Closed
	Q10	**In your words, how would you define TD?**	**Open**
	Q11	How close to the above TD definition is your understanding about TD?	Closed
	Q12	Are there any parts of the definition above from McConnell that you disagree with?	Open
	Q13	**Please give an example of TD that had a significant impact on the project that you have chosen to tell us about:**	**Open**
	Q14	Why did you select this example?	Open
	Q15	**About this example, how representative it is?**	**Closed**
iTDp-RQ2	Q16	What was the immediate, or precipitating, cause of the example of TD you just described?	Open
	Q17	What other cause or factor contributed to the immediate cause you described above?	Open
	Q18	What other motives or reasons or causes contributed either directly or indirectly to the occurrence of the TD example?	Open
	Q19	Considering all the cases of TD youve encountered in different projects, and the causes of those TD cases, which causes would you say are the most likely to lead to TD (ordered by likelihood of causing TD)? Please list up to 5 causes.	Open
iTDp-RQ3	Q20	Considering the TD item you described in question 13, what were the impacts felt in the project?	Open
	Q21	Considering all the cases of TD youve encountered in different projects and the effects of that TD that you have personally experienced, which 5 effects would you classify as the effects that have a bigger impact (ordered by their level of impact)	Open
iTDp-RQ4	**Q22**	**Do you think it would be possible to prevent the type of debt you described in question 13?**	**Closed**
	Q23	**If yes, how? If not, why?**	**Open**
	Q24	**Once identified, was the debt item monitored?**	**Closed**
	Q25	**If yes, how? If not, why?**	**Open**
	Q26	**Has the debt item been paid off (eliminated) from the project?**	**Closed**
	Q27	**If yes, how? If not, why?**	**Open**
	Q28	Considering your personal experience with TD management, what actions have you performed to prevent its occurrence?	Open

of TD (Q11), choosing one of the following options: “Very close,” “Close,” “Far,” “Very Far,” and “Had no prior knowledge of TD.” In Q12, the participant indicates whether they agree with McConnell’s concept and specifies the points they do not agree with, if applicable. In Q13, the participant provides an example of a TD item that occurred in its project. About the example, the participant justifies why they were chosen (Q14) and indicates how representative it is (Q15), choosing one of the following options “It was a unique instance,” “It is the type of thing that happens from time to time in the project,” and “It is the type of thing that happens very often in the project.”

Questions Q16-Q19 seek to identify the causes that lead software development teams to incur TD items in their projects (RQ2). Using the TD example provided by the participant in Q13 as context, they indicate the causes in Q16-Q18. Then, in Q19, the participant lists up to five causes that can cause TD items, considering all TD cases they have already experienced.

Questions Q20-Q21 intend to identify the effects of the presence of TD items in software projects (RQ3). Using the TD example provided by the participant in Q13 as context, they indicate the effects in Q19. Then, in Q20, the participant lists up to five effects that have the greatest impact on the project, considering all the TD cases they have experienced.

Lastly, questions Q22-Q28 aim to identify how TD has been managed in practice concerning its prevention, monitoring, and payment. In questions Q22-Q23 and Q28, participants discuss TD prevention. Using the TD example provided by the participant in Q13, they indicate whether this TD item could have been prevented (Q22). If so, it describes how (Q23). Otherwise, explain the reason for not preventing (Q23). In Q28, the participant indicates which actions to avoid TD have already been taken by the participant to prevent TD, but the participant uses all their professional experience as context (that is, it is not related to the TD example provided by the participant in Q13). To answer about monitoring and payment, participants use the example provided in Q13. They indicate whether this TD item was monitored or not (Q24) and whether it was paid off or not (Q26). If monitored or paid, they describe how this happened in questions Q25 and Q27, respectively. Otherwise, they justify the non-monitoring (Q25) or non-payment (Q27).

In this dissertation, we only use a subset of questions (marked in bold in Table 3.2) related to participants characterization (Q1 thru Q8), participants point of view on TD (Q10, Q13, and Q15), and TD management (Q22 thru Q27).

The following sections (3.3 and 3.4) will explain the data collection and analyses performed in the InsighTD context.

3.3 DATA COLLECTION

At the time of this writing, the project has concluded data collection for the InsighTD replications in Brazil, Chile, Colombia, Costa Rica, Serbia, and the United States. The data collection strategy was to send e-mail invitations for the survey to software practi-

tends to be more difficult and costly to maintain than if it had been implemented good object-oriented practices).”

tioners from those countries. The project followed the same strategy in all cases, using LinkedIn, industry-affiliated member groups, mailing lists, and industry partners as invitation channels. The data-gathering stage was done in 2018 in Brazil and the United States and 2019-2020 in Chile, Colombia, Costa Rica, and Serbia.

At least three researchers validated all collected answers in each of the six replications. Two researchers separately validated the answers to Q10, Q13, Q23, Q25, and Q27, and a third researcher resolved possible disagreements. The determination of a "valid answer" was based on the following acceptance criteria:

- *The participants should consider a TD perspective to answer the questions.* InsighTD replication teams analyzed the definition of TD described by the participants in Q10. If the definition was in conformance with the definition of TD used in the InsighTD project (RIOS *et al.*, 2020), then we moved to the second acceptance criterion.

- *The participants should provide a valid example of a TD item.* InsighTD replication teams analyzed the example of TD described by the participants in Q13. If the example was in conformance with the definition of TD used in the InsighTD project (RIOS *et al.*, 2020), then InsighTD replication teams concluded that the participant answered the other surveys questions considering the TD perspective. For example, answers like "I dont know what TD means" and "Errors that arise while fixing other issues" were discarded because they do not represent valid TD examples. On the contrary, answers from participants who provided examples of TD items like "Tests planned but not executed" and "Hard maintenance and future change due to poor documentation from the development team" passed to the next validation step.

- *The participants should provide valid answers to TD prevention, monitoring, and payment questions.* We analyzed all answers given for Q23, Q25, and Q27 to verify whether we can identify prevention, monitoring, or payment practices, respectively, or PARs for TD non-prevention, non-monitoring, or non-payment, respectively. As we did not find any invalid answers, we concluded that the participants did not misunderstand this question. For example, we considered the answers given by a participant who reported that "conduct ongoing code reviews to ensure project growth is consistent with established quality," "keep a list of all TD and regularly determine how to eliminate it," and "the identified tech debt has been resolved through updated designs and refactors..." are TD prevention, monitoring, and payment practices, respectively.

3.4 DATA ANALYSES

As shown in Table 3.2, the questionnaire is composed of open and closed questions, requiring the application of different data analysis strategies.

3.4.1 Analyzing Answers from Closed Questions

For closed questions, we used descriptive statistics and calculated the share of participants choosing each option. These procedures are used in Q1 thru Q8 (characterization questions), Q15 (the frequency that the TD item occurs in the project), Q22 (whether the TD item could be avoided or not), Q24 (whether the TD item was monitored or not), and Q26 (whether the TD item was paid off or not).

3.4.2 Analyzing Answers from Open-ended Questions

To analyze the open-ended questions, we applied qualitative data analysis techniques (STRAUSS; CORBIN, 1998; SEAMAN, 1999). More specifically, we performed open coding because we did not provide respondents with a predetermined list of practices and PARs. As Strauss and Corbin (1998) explained, open coding is the "analytic process through which concepts are identified, and their properties and dimensions are discovered in data." Thus, we identify the concepts, i.e., the central ideas in the data and their categories and characteristics that are present in the data. The categories are composed of properties that define the categorys meaning and dimensions, which are variations in the category (STRAUSS; CORBIN, 1998).

Figure 3.1 summarizes how our analysis process evolved to reveal the dimensions described above and other complexities of our central concepts. It shows that for TD prevention, monitoring, and payment practices, and PARs for TD non-prevention, non-monitoring, and non-payment, we used open coding (STRAUSS; CORBIN, 1998) to identify the types, natures, and categories of practices and PARs by grouping them into different properties and dimensions. This allowed us to understand the central concepts more deeply. Afterward, we discovered the relationships between TD prevention, monitoring, and payment practices and types of debt and between PARs for TD non-prevention, non-monitoring, and non-payment and types of debt. The following sections describe how we analyzed the data, resulting in this set of dimensions and relationships.

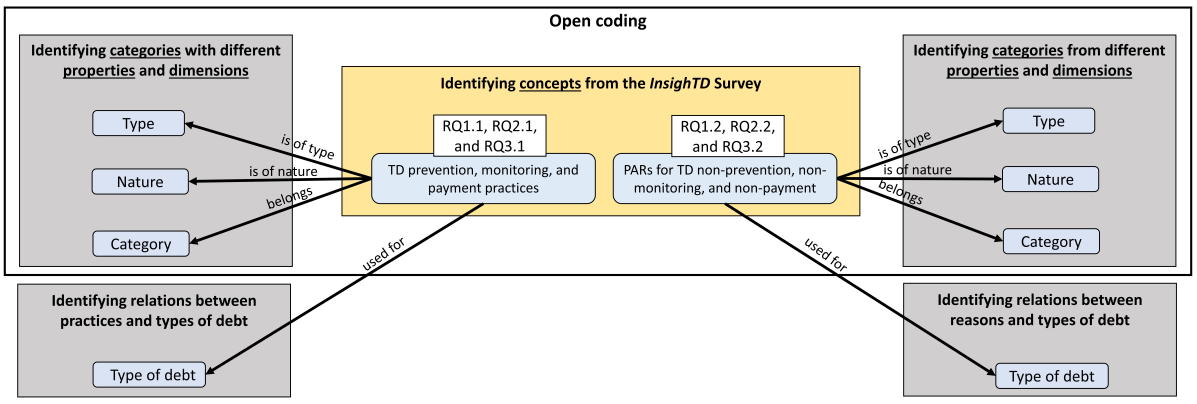

Open coding
Identifying categories with different properties and dimensions
Type
Nature
Category
is of type
is of nature
belongs
Identifying concepts from the InsighTD Survey
RQ1.1, RQ2.1, and RQ3.1
TD prevention, monitoring, and payment practices
RQ1.2, RQ2.2, and RQ3.2
PARs for TD non-prevention, non-monitoring, and non-payment
Identifying categories from different properties and dimensions
Type
Nature
Category
used for
used for
Identifying relations between practices and types of debt
Type of debt
Identifying relations between reasons and types of debt
Type of debt

Figure 3.1 Schema of research questions and their corresponding findings

3.4.2.1 Identifying Practices and PARs. In answers given to Q23, Q25, and Q27, we initially identified the concepts related to TD prevention, monitoring, and payment, i.e., the practices and PARs for these TD activities, resulting in a set of codes. We divided those codes into two subsets for each activity based on the answers to Q22, Q24, and Q26 (a yes/no question). If the answer was positive, the code was related to TD prevention, monitoring, or payment practices, respectively. This set of codes supports responses to RQ1, RQ1.1, RQ2, RQ2.1, RQ3, and RQ3.1. If the answer was negative, the code was related to PARs for TD non-prevention, non-monitoring, and non-payment, supporting responses to RQ1, RQ1.2, RQ2, RQ2.2, RQ3, and RQ3.2. The process was performed iteratively, revising and unifying codes at each analysis cycle until reaching a state of saturation, i.e., no new codes were identified. At the end of the analysis, we obtained a stable list of codes and their citation frequency.

At least three researchers conducted the coding in each of the six InsighTD replications. Each researcher could assume one of the following roles: (i) **code identifier** - the person responsible for extracting the codes from the answers; (ii) **code reviewer** - responsible for reviewing all extracted codes; and (iii) **referee** - responsible for resolving disagreements in codes identified by the code identifier and code reviewer. We had six code identifiers, six code reviewers, and six referees. The Brazilian replication team created the first codified list of TD prevention, monitoring, and payment practices and PARs for TD non-prevention, non-monitoring, and non-payment, which was distributed to the other replication teams to standardize the used nomenclature. The Brazilian replication team verified the consistency.

Code unification required some effort. To exemplify the process, let us consider the following answers given by three practitioners in Q23 when Q22 had a positive answer: "more structured requirements..." and "better product functional needs forecasting..." As these answers are associated with improvements in requirements, they were unified under the preventive practice *well-defined requirements*. Other participants cited the following explanation for TD non-monitoring in Q25 (when Q24 received a negative answer): "due to tight deadlines," "lack of time," and "the project timeline didn't allow it." The initially extracted PARs were *tight deadlines*, *lack of time*, and *insufficient timeline*, respectively. Then, as these PARs had different nomenclature but shared a common meaning, we unified them as *lack of time*. Regarding TD payment, participants cited the following TD practices in Q27 (when Q26 had a negative answer): "to make it solid and defect-free system" and "(...) corrective maintenance of artifacts." The initially extracted codes were *to make defect-free system* and *corrective maintenance of artifacts*, respectively. Then, as these codes had different nomenclature but shared a common meaning, we unified them as *bug fixing*.

3.4.2.2 Delving into the Practice Concepts. After identifying the TD prevention, monitoring, and payment practices, we analyzed them to better understand the concepts complexities. The data revealed several dimensions related to the concept of TD prevention, monitoring, and payment practices, resulting in the following groupings:

- **Types of practices**. The data showed that several fundamentally different groups of responses differed in their role in the software development process. For instance, some practices can manage TD items directly (*action*) while others (*enabling practices*) can support these activities by creating a favorable scenario for future TD management or being practices from other TD management activities, such as TD identification and prioritization. For example, *well-defined requirements*, *tracking the cost*, and *code refactoring* are prevention, monitoring, and payment *actions*, respectively, while *training*, *team restructuring*, and *negotiating a deadline extension* are *enabling practices* for TD prevention, monitoring, and payment, respectively. Thus, we followed open coding, grouping the practices into types. The dissertations author assigned a type for each practice, and an experienced researcher reviewed this assignment.

- **Nature of practices**. All the practices we discovered are related to either technical or managerial activities of software development. The difference between these groups is related to the nature of the practice. The dissertations author identified each practices nature (technical or managerial), and an experienced author reviewed this identification.

- **Categories of practices**. We found that many of the codes for practices were related to each other, so we followed a grouping process to organize them into categories reflecting the primary concern of each subset. This process also followed open coding. The dissertations author analyzed each code, comparing its meaning with that of the other codes. When he identified a relation between them, he grouped them into a category. To name the categories, we used the list proposed by Rios *et al.* (2019), which is presented in Table 3.3. For consistency, once again, the entire process was reviewed by an experienced researcher.

Table 3.3 Categories adapted from Rios *et al.* (2019)

Name	**Definition**
Development issues	Refers to TD elements issues that occur during project development.
External factors	Encompasses TD elements that are external to the development team and organization.
Infrastructure	Groups TD elements related to tools, technologies, and development environments.
Internal quality issues	Encompasses TD items related to internal quality issues.
Lack of knowledge	Refers to TD elements related to the teams lack of knowledge to develop the project.
Methodology	Refers to TD elements related to processes and methodologies used in the development of the project.
Organizational	Groups TD elements associated with organizational level.
People	Encompasses TD elements directly related to members of software development teams.
Planning and management	Groups TD elements related to project planning and management.

3.4.2.3 Delving into PAR Concepts. We also analyzed the dimensions of the PAR concept, resulting in the following groupings:

- **Types of PARs**. The data showed that a PAR can be a *decision factor* or an *impediment*. The former refers to the teams decision not to manage the debt, and the latter represents situations in which the practitioners could have the intention of managing the debt, but they would not be able to due to issues that are out of their control. For example, *focusing on short-term goals* is a decision factor for TD non-payment, while *lack of time* is an impediment. Thus, we followed open coding, grouping the PARs into types. The dissertations author assigned a type for each PAR, and an experienced researcher reviewed this assignment.

- **Nature of PARs**. All the PARs we discovered are related to either technical or managerial activities of software development. The dissertations author identified the nature (technical or managerial) of each PAR, and an experienced author reviewed this identification.

- **Categories of PARs**. We found that many of the codes for PARs were related to each other, so we followed a grouping process to organize them into categories reflecting the primary concern of each subset. This process also followed open coding. The dissertations author analyzed each code, comparing its meaning with that of the other codes. When he identified a relation between them, he grouped them into a category. To name the categories, we used the list proposed by Rios *et al.* (2019), which is presented in Table 3.3. For consistency, once again, the entire process was reviewed by an experienced researcher.

3.4.2.4 Relationships with Types of Debt. We followed the same process performed by Rios *et al.* (2020) to identify a TD type presented in the TD example given to Q13. This process uses the definitions of TD types reported by Rios, Mendonça and Spinola (2018) and the list of TD indicators given by Alves *et al.* (2016). To illustrate this process, let us consider the following answer given for Q13: "using an external framework that took more time and resources where a native framework would have worked better." By analyzing this text, we see that the answer describes issues in the architecture, representing a scenario of architecture debt. This determination was performed by at least three researchers in each InsighTD replication. As the respondents used the TD example provided by them in Q13 as context to answer the questions on TD prevention, monitoring, and payment, we used the TD type identified in Q13 to associate this type with TD prevention, monitoring, and payment practices and PARs for TD non-prevention, non-monitoring, and non-payment.

3.5 THREATS TO VALIDITY

There are threats to validity that could affect our work. We used the following categorization of Wohlin *et al.* (2012) to identify and analyze the threats: (i) construct validity (threats from this category are related to the design of the experiment or social factors),

(ii) conclusion validity (this is associated with threats that affect the ability to draw correct conclusions from the results), (iii) internal validity (threats from this category are associated with other factors that could affect the results without the researchers knowledge), and (iv) external validity (this is related to the possibility of generalizing the results). We tried to remove them when possible or mitigate their effects when removal was not possible.

- **Construct validity**. We identified a threat arising from the validity of participants responses. Participants could answer the survey questions without considering the context of TD or TD prevention, monitoring, or payment. And as the survey was performed remotely, it can maximize the effect of this threat. To mitigate it, we included two acceptance criteria: (i) the example of TD provided by participants in Q13 must describe an actual TD item, and (ii) the answer given to Q23, Q25, and Q27 must be associated with practices or PARs for TD prevention, monitoring, and payment, respectively. Then, a participants answer was considered in our analysis if the answer fit into these acceptance criteria.

- **Conclusion validity**. We identified two threats affecting the conclusion validity. First, a threat is related to the identification of TD type, the coding of practices and PARs, and the grouping processes used in this study. As the identification of TD types and the coding process are subjective and subject to inconsistencies, both processes were performed by three researchers of each replication team playing different roles: identifier, reviewer, and referee. For the grouping process, a researcher grouped the practices or reasons into types, natures, and categories, and an experienced researcher reviewed the grouping. Lastly, some of the data may seem old, but practices and PARs are related to the what and not about how. For example, the TD payment practice *code refactoring* (the *what*) can be done by several means (the *how*), such as using external tools to automating refactoring, or by doing some small changes in the code. Therefore, the practice could remain updated no matter how it was performed. Similarly, a PAR for TD non-payment, such as *customer decision* (the *what*) can come from multiple sources (the *how*).

- **Internal validity**. As the survey questions were answered remotely, the participants could misunderstand these questions, arising an internal threat which affects our study. To minimize it, the survey passed through three internal reviews conducted by experienced researchers from the InsighTD project and one external review conducted by a senior researcher. Afterward, a pilot study was run to assess the survey questions, structure, and duration. More details on this process are described in Rios *et al.* (2020).

- **External validity**. We reduced this threat by targeting industry practitioners and seeking to achieve respondent diversity from several countries. Although this diversity brings a good view of TD prevention, monitoring, and payment and their practices and PARs, we cannot say how generalizable the results are because we are not able to estimate the representativeness of our sample given the lack of empirical data characterizing the population. However, an argument can be made that the ecological validity (ANDRADE, 2018) of the work, i.e., the extent to which these findings approximate other real-world scenarios, is likely to hold in other settings.

3.6 THE INSIGHTD DATA SET

Although researchers from 12 countries participated in the InsighTD project, to date, the data collection and analysis process has been completed by replication teams from Brazil, Chile, Colombia, Costa Rica, the United States, and Serbia. In total, we obtained 653 valid (according to the criteria presented in Subsection 3.3) answers, including data from Brazil (107 answers), Chile (89), Colombia (134), Costa Rica (145), Serbia (79), and the United States (99). This data helps us understand the state of the practice of TD prevention (Chapter 4), monitoring (Chapter 5), and payment (Chapter 6).

Figure 3.2 summarizes the participants characterization by country, company and team size, system size and age, role, level of experience, and process model. Overall, the collected data includes a wide variety of projects from the software development industry in Brazil, Chile, Colombia, Costa Rica, Serbia, and the United States, including projects of different ages, sizes, team sizes, process models, and composed of several participant roles and levels of experience from organizations of different sizes.

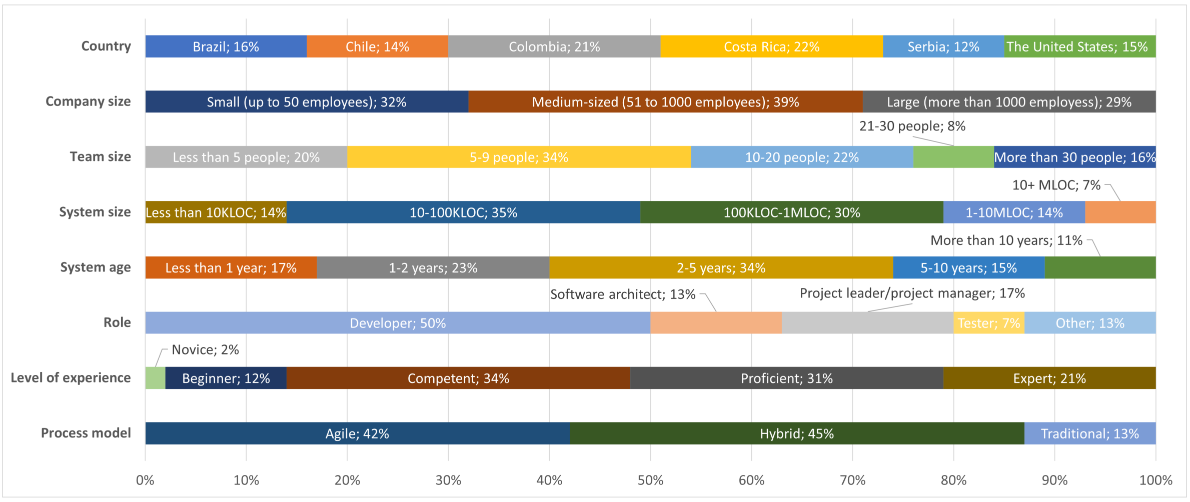


Figure 3.2 Summary of participants characterization

Figure 3.3 shows how participant characteristics are spread among countries. In each diagram, the possible values for each participant characteristic are shown in the left, while the countries are presented in the right side. For instance, by considering the “company size,” we can see that most of the participants worked in medium-sized companies (40%) and that most of them were from the Costa Rican software industry (23%). The distribution of each characterization variable is similar from country to country.

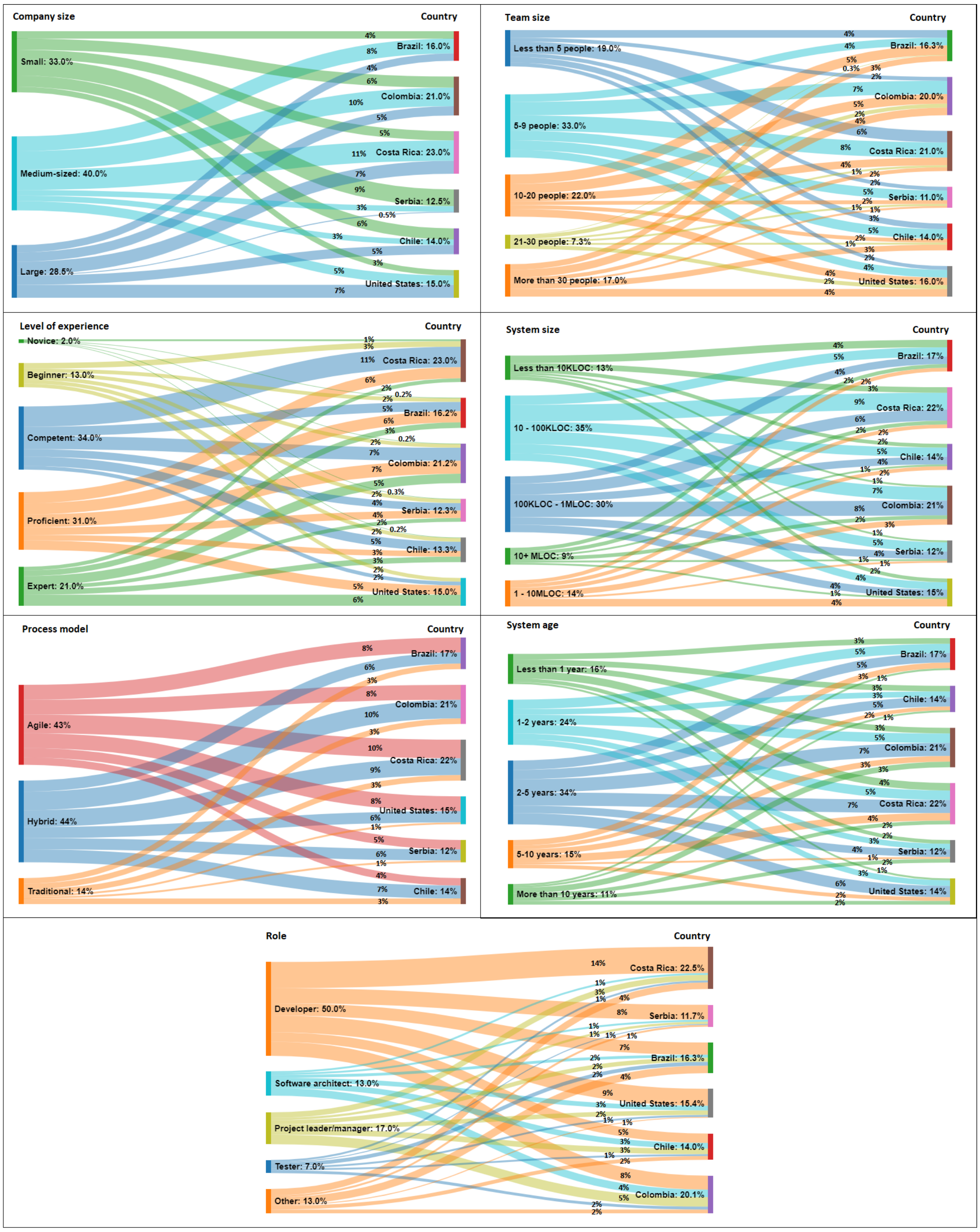


Figure 3.3 Demographics per country

3.7 CONCLUDING REMARKS

This chapter presents the research method we followed to investigate the state of the practice of TD prevention, monitoring, and payment. It also introduces the InsighTD project, which aims to establish a set of empirical data on the state of TD practice. The project is the first large-scale study in the TD field, encompassing the causes and effects of TD in software projects and investigating how software practitioners have managed the debt in their projects.

Our method allows us to identify the practices used to prevent, monitor, and pay off debt items and the PARs used to explain the TD non-prevention, non-monitoring, and non-payment. Besides, the method recognizes that software practitioners have used distinct types of practices from different natures and categories in their TD prevention, monitoring, and payment initiatives. The same occurred to PARs. Lastly, the method can relate the practices and PARs to types of debt, revealing specific practices used to deal with a specific type of debt or specific PARs used to justify the non-management of a specific type of debt.

The following chapter presents the results from the InsighTD project on TD prevention.

Chapter

4

This chapter presents the state of the practice of TD prevention obtained from analyses of the InsighTD data set.

TECHNICAL DEBT PREVENTION'S STATE OF PRACTICE

Technical debt (TD) prevention is one of the critical TD management activities. It aims to avoid potential TD items (LI; AVGERIOU; LIANG, 2015; RIOS; MENDONÇA; SPINOLA, 2018). For example, software teams can improve their current development processes to curb the occurrence of specific types of debt (DAVIS, 2013).

Although the technical literature provides evidence on TD prevention (see Chapter 2 for more details), the subject deserves a more comprehensive investigation. After all, it is fair to expect that TD prevention can sometimes be cheaper than its repayment. Moreover, prevention may also help other TD management activities. For example, setting up prevention practices helps catch inexperienced developers not-so-good solutions (YLI-HUUMO; MAGLYAS; SMOLANDER, 2016). Further, having information on TD prevention practices can guide software practitioners in choosing practices that could be used in their projects, learning from the experience of practitioners who have applied these practices.

However, TD prevention initiatives require more than awareness about TD prevention practices. In general, software teams need the support of their organization to spend effort on preventive activities in their projects. Without managerial and organizational support, a TD prevention initiative may fail to enact its practices, even when the development team is aware of them. It is crucial to recognize the factors that can hinder the execution of TD prevention activities and define strategies to minimize their effects.

This chapter aims *to investigate, from the point of view of software development practitioners, which practices can be used to prevent TD items and what Practice avoidance reasons (PARs) would justify the non-prevention of these items.* To achieve this goal, we define **RQ1: How could software development teams avoid technical debt items on their projects?** And its sub-questions: RQ1.1: What are the leading prevention practices that can be used to prevent TD? and RQ1.2: What are the leading practice avoidance reasons (PARs) considered by software practitioners for TD non-prevention? To answer these RQs, we use and analyze the InsighTD data set, as described in Chapter 3.

This chapter is based on the following papers: (i) Actions and Impediments for Technical Debt Prevention: Results from a Global Family of Industrial Surveys (FREIRE *et al.*, 2020c) and (ii) A Comprehensive View on TD Prevention Practices and Reasons for not Prevent It (FREIRE *et al.*, 2023b).

The main contributions of this chapter are:

- An up-to-date, comprehensive list of practices related to TD prevention ranked by the number of their citation by software practitioners.
- An up-to-date, comprehensive list of PARs ranked by the number of their citation by software practitioners.
- A categorization of TD practices and PARs with respect to their relationship to software planning, development, management, and methodology issues.
- A definition of types of practices and types of PARs in TD prevention initiatives.
- A definition of the nature of practices and nature of PARs with respect to technical and managerial activities.
- The relationships between types of debt and practices, and types of debt and PARs.
- A comparison of practices and PARs found in this study to those reported in the technical literature.

Next, Section 4.1 presents the perception of software practitioners on TD prevention. Section 4.2 discusses the results and compares them with those reported by related work. And Section 4.3 offers the concluding remarks of this chapter.

4.1 PERCEPTION OF INSIGHTD'S PRACTITIONERS ON TD PREVENTION

This section presents our findings from the InsighTD data set analyses concerning TD prevention (answers given to questions Q22 and Q23 of the InsighTD questionnaire). We organize these findings per RQ.

4.1.1 RQ1.1: What are the Leading Prevention Practices that can be used to prevent TD?

In Q22, ~91% of the participants indicated that the TD item described by them in Q13 could be prevented, and ~92% of those described in Q23 how TD could be prevented. For answering RQ1 and its sub-questions, we used this subset of responses, comprised of 546 answers.

We found 89 prevention-related practices that could be used to curb the presence of TD. Table 4.1 presents the 10 most commonly cited. Appendix Table B.1 presents the identified prevention-related practices and quotes from participants. Table 4.1 reports the **prevention-related practice name** and the total number (i.e., count) of citations (**#CP**), i.e., the number of projects in which the practice could be used (since each response corresponded to one particular project). The column **%PP** presents the percentage of #CP in relation to the total of projects (546) that cited at least one preventive practice. It reveals how frequently each preventive practice could be used in software projects. *Well-defined requirements* is the most cited prevention-related practice, followed by the *adoption of good programming practices*, *better project management*, and *training.*

Table 4.1 Top 10 TD prevention-related practices

NO	**Prevention-related Practice**	**#CP**	**%PP**
1st	Well-defined requirements	57	10%
2nd	Adoption of good programming practices	49	9%
3rd	Better project management	43	8%
4th	Training	36	7%
5th	Following the project planning	34	6%
6th	Improving software development process	33	6%
7th	Improve documentation	26	5%
8th	Using good design practices	26	5%
9th	Well planned deadlines	26	5%
10th	Better project planning	24	5%

Caption:
#CP - Count of prevention practices.
%PP - Percentage of CP in relation to the total of all projects (546).

In the context of this work, *well-defined requirements* is related to a good definition of the project requirements, ensuring a better understanding of the functionalities that will be developed, as described in the quote extracted from the participants answers: "more structured requirements..." and "better product functional needs forecasting." The *adoption of good programming practices* practice means the use of practices that guarantee code with higher quality, as described as follows: "don't let a developer write one thing in a different language than the rest of your project" and "better code quality following design principles and patterns."

Better project management practice is related to the activities the management team performs, for example, "better project management" and "better organization by management team." The *training* practice indicates that the development team could be trained in the technologies used in the project, or training could be offered to level the team, as indicated in the following answers: "keeping the team updated on good development practices" and "considering the study of the technologies used in the project."

Following the project planning practice indicates that the planning of project activities can prevent the occurrence of TD items, as we can observe in "following the plan that has been defined..." The *improving software development process* practice means that software teams can apply improvements on their development process to prevent TD items, as we can see in "better collaborative processes that also incrementally address quality..." and "use a shorter and iterative development cycle where milestones are clearly laid out and are achievable in a viable time frame."

The practices *improve documentation*, *using good design practices*, *well planned deadlines*, and *better project planning* were cited by 5% of the participants. The *improve documentation* practice indicates that the debt can be prevented when the software team makes the documentation available and updated, as described as follows: "keeping thorough documentation and making it readily available to everyone" and "spend more time on documenting."

The *using good design practices* practice is related to using practices to guarantee design quality, for example, "by spending more time on design and creating specification" and "good analysis and design must be carried out." The *well planned deadlines* practice means that the team needs to understand the activities that will be done to estimate the time required by the team, as we can observe in: "adequate time given to developers to READ and UNDERSTAND said requirements" and "more time to complete task proper way." Lastly, the *better project planning* practice indicates improving the planning of the project considering the estimation of each activity, for example, "planning instead of imagining..." and "better planning and estimates."

By analyzing the top ten preventive practices, we realized that practices were used to prevent TD items (e.g., *well-defined requirements* practice) or support the TD prevention activity (e.g., *training* practice). Thus, we identified one dimension for TD prevention practices, named ***type***, having two values:

- **Preventive action**: refers to practices that directly allow the prevention of TD items. For example, *well-defined requirements*, *adoption of good programming practices*, and *better project management.*
- **Enabling TD prevention**: refers to practices that increase the ability of the team to prevent TD items, such as *training*, *technical support*, and *being committed.*

Table 4.2 presents the identified types of prevention practices, reporting the **types name**, the number of unique prevention practices (**#P**), and the total number (i.e., count) of prevention practices (**#CP**) cited in each type. #CP also indicates the number of projects that could use that preventive practice in each prevention practice type. Column **%PP** corresponds to the percentage of #CP about the total of all projects, indicating how frequently each prevention practice type could be used in software projects. Appendix Table B.2 presents, for each type, the prevention-related practices ranked by number of citations.

Table 4.2 Types of TD prevention-related practices

Type of prevention-related practice	**#P**	**#CP**	**%PP**
Preventive action	46	517	95%
Enabling TD prevention	44	303	55%

Caption:
#P - Count of unique cited prevention-related practices.
#CP - Count of prevention-related practices.
%PP - Percentage of CP in relation to the total of all projects (546).

Although we have identified almost the same quantity of preventive practices in each type, the number of citations (#CP) and frequency of use (%PP) are quite different. While preventive actions were more cited and could be used in almost all projects, enabling TD prevention could be used in just over half of the projects.

Regarding the dimension ***nature*** (technical or managerial), among the top 10 cited TD prevention-related practices, the **technical subset** comprises *well-defined requirements*, *adoption of good programming practices*, *training*, *improve documentation*, and *using good design practices*, representing 36% of all projects (column %PP at Table 4.1). The **managerial subset** is composed of *better project management*, *following the project planning*, *improving software development process*, *well planned deadlines*, and *better project planning*, representing 29% of all projects (column %PP at Table 4.1). Appendix Table B.3 presents the TD prevention-related practices of each nature ranked by number of citations.

Table 4.3 presents the relation between the types and natures of all TD prevention-related practices. Most prevention actions are technical, while most enabling TD prevention practices are managerial.

Table 4.3 Relation between type and nature of prevention-related practices

Type of prevention-related practice	**Nature of prevention-related practice**	
	Technical	**Managerial**
Prevention action	33 (39%)*	10 (24%)
Enabling TD prevention	12 (9%)	34 (28%)
Total	45 (48%)	44 (52%)

*The number in parentheses represents the percentage of the total number (i.e., count) of prevention-related practices cited in each nature in relation to the total of all cited practices (820).

Another dimension, named ***category***, is used to divide the practices into groups representing software development concerns. As a result, this dimension has the following values:

- **Development issues**: groups nine practices associated with software development activities, such as *adoption of good programming practices*, *appropriate reusing of code*, and *considering technical constraints.*

- **Infrastructure**: encompasses two practices (*organizing code repository* and *use the most appropriate version of the technology*) associated with tools, technologies, and development infrastructure.

- **Internal quality issues**: brings together six practices that can be employed to address limitations that compromise the internal quality of the software. For example, *improving the maintainability of the project*, *refactoring*, and *using good design practices.*

- **Methodology**: refers to practices related to the process followed by the team. This category has 28 practices, such as *improving software development process*, *improve documentation*, and *creating tests.*

- **Organizational**: groups ten practices associated with organizational decisions. For example, *better understanding of development process by businesses*, *contracting a domain expert to architect the project*, and *technical support*.
- **People**: includes nine practices related to team characteristics. *Technical knowledge*, *organized team*, and *discipline* are examples of practices composing this category.
- **Planning and management**: encompasses 25 practices related to managerial activities. For example, *better project management*, *following the project planning*, and *allocation of qualified professionals*.

Table 4.4 presents the categories of prevention practices, reporting the **categorys name**, the number of unique practices cited (**#P**), and the total number (i.e., count) of practices (**#CP**) cited in each category. #CP also indicates the number of projects that could use a practice in each category. Lastly, the column **%PP** corresponds to the percentage of #CP in relation to the total of all projects. Appendix Table B.4 presents, for each category, the TD prevention-related practices ranked by number of citations.

Table 4.4 Categories of prevention-related practices

Category of prevention-related practices	**#P**	**#CP**	**%PP**
Planning and management	25	287	53%
Methodology	28	214	39%
Internal quality issues	6	122	22%
Development issues	9	87	16%
Organizational	10	56	10%
People	9	46	8%
Infrastructure	2	8	1%

Caption:
#P - Count of unique cited prevention practices.
#CP - Count of prevention practices.
%PP - Percentage of CP in relation to the total of all projects (546).

The *planning and management* category has the greatest number of practices, followed by *methodology*. This result indicates that managerial and methodological practices are crucial for TD prevention. Other categories that catch our eye are *internal quality issues* and *development issues*. Their practices could be used by 22% and 16% of the projects. This result indicates that practices related to the quality of the system and the code implemented by the team also matter for TD prevention.

4.1.1.1 TD Prevention Practice per Type of Debt. Figure 4.1 presents the relationship among the **types of debt** and the top ten TD prevention-related practices in percentages, while Figure 4.2 shows the absolute values of these relationships, i.e., the quantity of times that each relationship was found. All practices from the top ten could be applied to avoid code, design, requirements, and test debt items. For architecture and documentation debt items, only the practices *better project planning* and *improving software development process* could not be applied, respectively. The practices *better project planning*, *improve documentation*, *using good design practices*, and *well-defined requirements* could not be applied to preventing defect debt items. We only found five practices that could be applied to prevent infrastructure and people debt items, while people and service debt items could be prevented by applying four practices. Lastly, automation test, usability, and versioning debt items could be prevented using three practices.

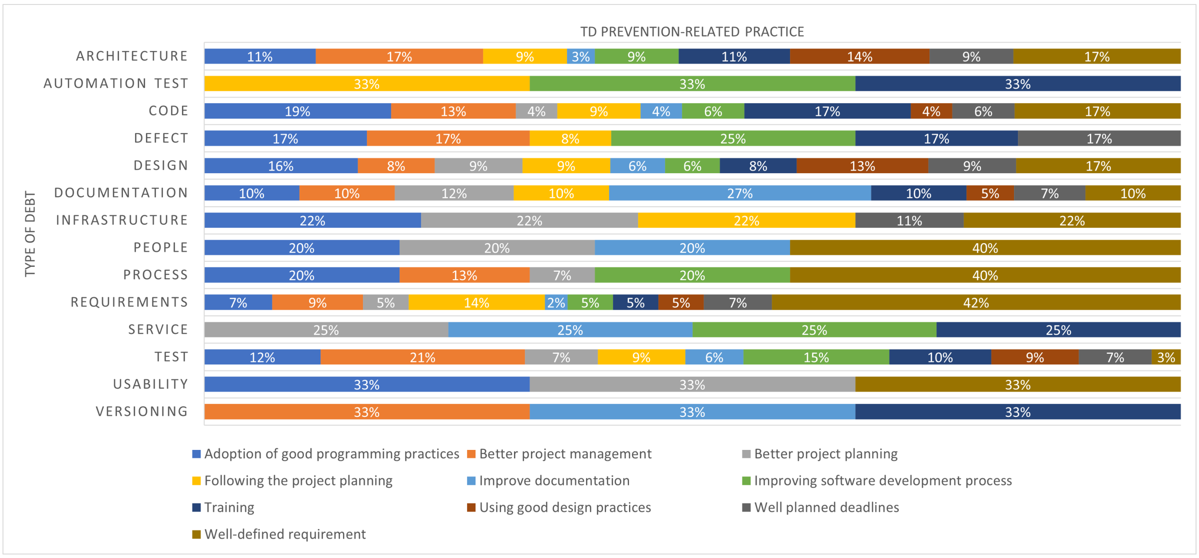


Figure 4.1 Relationship among types of debt and the top 10 TD prevention-related practices - percentage values

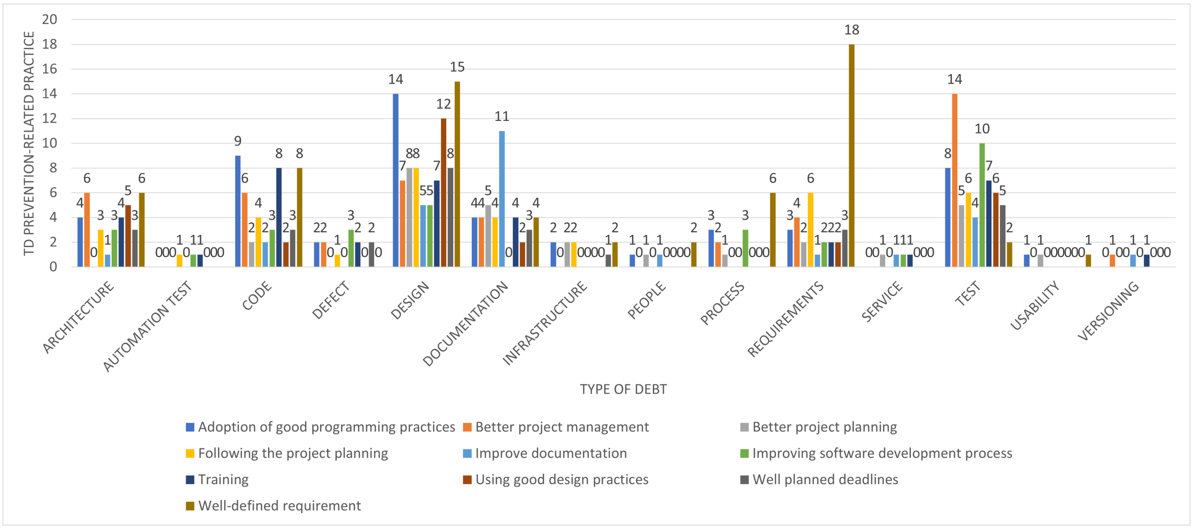


Figure 4.2 Relationship among types of debt and the top 10 TD prevention-related practices - absolute values

By analyzing Figure 4.3, we can see that practitioners could use practices to prevent all but build debt items. At the same time, practices to enable TD prevention could be used to prevent all but usability debt items. The number of times that each relationship occurred is shown in Figure 4.4.

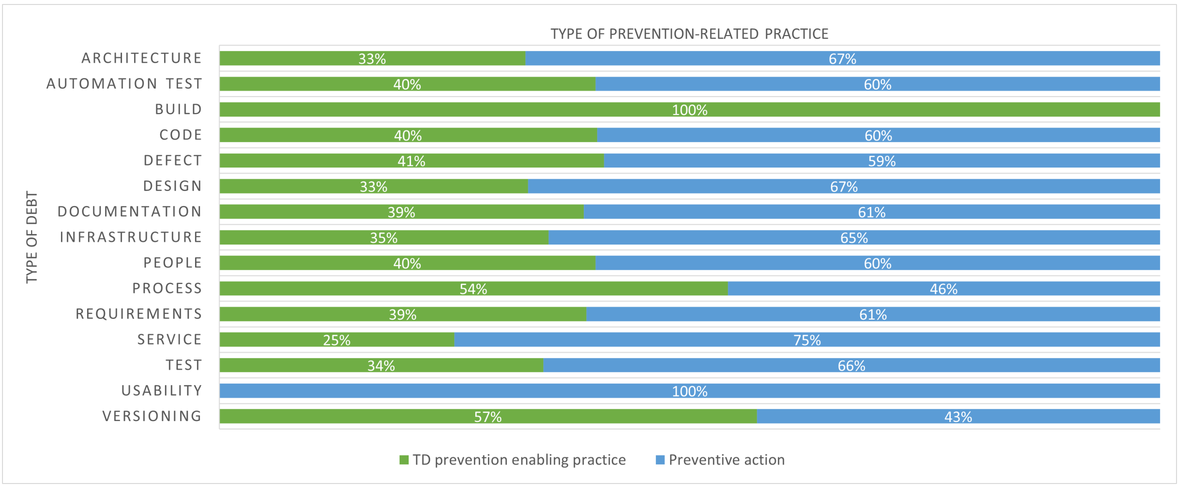


Figure 4.3 Relationship among types of debt and the types of TD prevention-related practices - percentage values

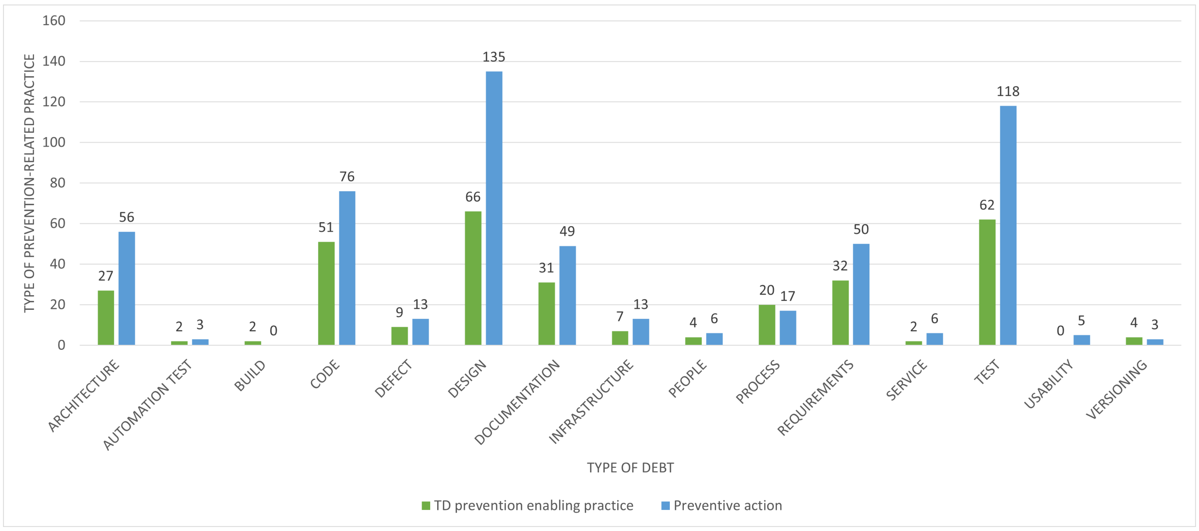


Figure 4.4 Relationship among types of debt and the types of TD prevention-related practices - absolute values

Lastly, we investigated the relationship between types of TD and the categories of TD prevention-related practices. Figure 4.5 shows this relationship, indicating *planning and management* category encompasses practices that could be used to avoid all types of debt. Practices from the category *methodology* could be used to avoid all but build debt items. Automation test, build, and versioning debt items could not be prevented by practices from the *internal quality issues* category. And practices from the *organizational* category could not be applied to prevent infrastructure, people, and usability debt items. Figure 4.6 shows the number of times of each relationships was found.

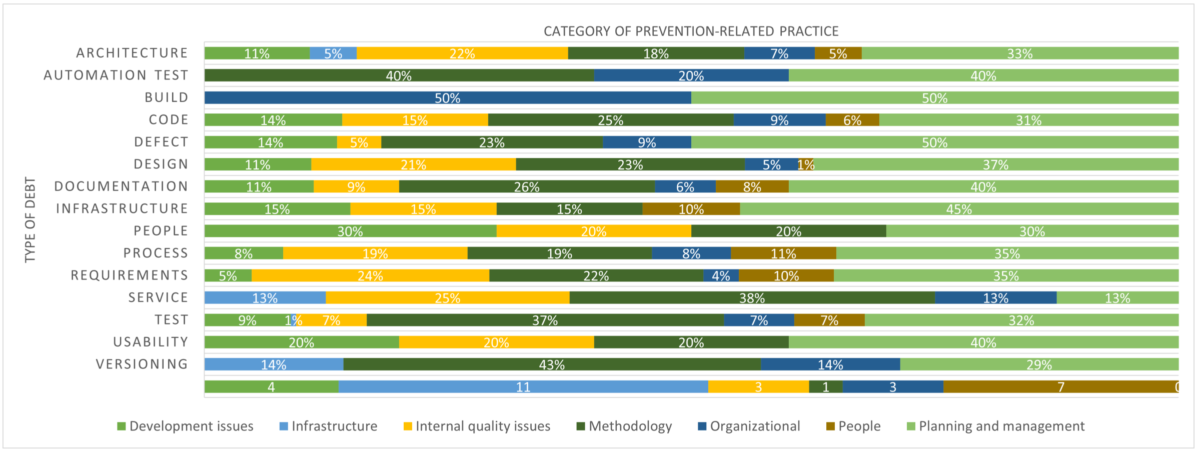


Figure 4.5 Relationship among types of debt and categories of TD prevention-related practices - Percentage values

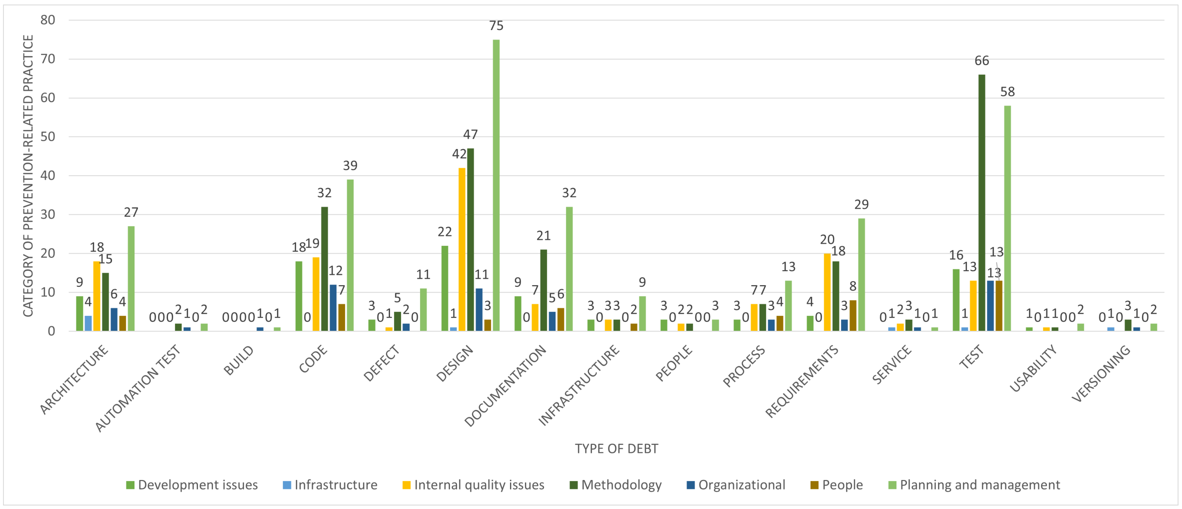


Figure 4.6 Relationship among types of debt and categories of TD prevention-related practices - Absolute values

4.1.2 RQ1.2: What are the Leading Practice Avoidance Reasons (PARs) considered by Software Practitioners for TD Non-prevention?

In total, 9% of the participants indicated that the TD item described in Q13 could not be prevented and ∼80% of those explained in Q23 why the TD could not be prevented. For answering RQ2 and its sub-questions, we used this subset of responses, composed of 49 answers.

We found 23 PARs that software practitioners could use to explain the TD non-prevention. Table 4.5 presents the ten most commonly cited. All the identified PARs and quotes from participants are available in Appendix Table B.5. Table 4.5 reports the **PAR name** and the total number (i.e., count) of citations (**#CPAR**), i.e., the number of projects in which the PAR could be considered (since each response corresponded to one particular project). The column **%PARP** presents the percentage of #CPAR in relation to the total of projects (49) that cited at least one PAR. It reveals how frequently each PAR could be considered in software projects. *Short deadline* was the most cited PAR, followed by *ineffective management*, *lack of predictability in the software development*, and *requirements change.*

The *short deadline* PAR refers to software teams that do not have time to prevent TD items, as described in the quote extracted from the participants answers: "because we always need to make trade-offs due to time pressure" and "a lot of the TD Ive encountered will not be solved because it's a symptom of market forces. There will always be a rush to get a product out." The *ineffective management* PAR is related to a lack of managerial activities that can support the TD preventive initiatives that could be performed by software teams, as described as follows: "management and process were not mature enough to prevent TD" and "there are demands that arise and must be resolved promptly."

Table 4.5 Top 10 practice avoidance reasons (PARs) for TD non-prevention

NO	Practice avoidance reason (PAR)	#CPAR	%PARP
1st	Short deadline	14	29%
2nd	Ineffective management	7	14%
3rd	Lack of predictability in the software development	5	10%
4th	Requirements change	5	10%
5th	Pressure for results	4	8%
6th	Lack of technical knowledge	3	6%
7th	Documentation issues (lack of or non-updated)	2	4%
8th	Lack of concern about maintainability	2	4%
9th	Lack of good technical solutions	2	4%
10th	Lack of process maturity	2	4%

Caption:
#CPAR - Count of practice avoidance reasons for TD non-prevention.
%PARP - Percentage of CPAR in relation to the total of all projects (49).

The *lack of predictability in the software development* PAR refers to the difficulty in knowing if the software under development will support the functionalities requested by the stakeholders, for example, "no large project ever has the foresight to account for every scenario" and "it's difficult to see into the future to know what it will take to develop software. Sometimes newly added requirements or specs can change what is expected out of a class which outdates its ability to function properly." The *requirements change* PAR indicates that changing in requirements can result in refactoring in the code or reducing its maintainability, as we can observe in "requirements are always going to change during development..." and "because when the client asks for features abruptly, no matter how generalized the architecture is towards the problem, with an outlier there may be, that can mean a refactor of the code, and that could dirty the code, reducing its maintainability."

The *pressure for results* PAR means that software teams cannot prevent TD items because it does not have an appropriate scenario due to pressure for results, as indicated in "no, because the pressure of business and priorities simply disrupts the rigorous technical work" and "the pressure for results and compliance schedules (which are always required in terms of monetary benefit), mean that there is technical debt practically from day one of the project." The *lack of technical knowledge* PAR refers to the need to increase the technical knowledge of the software team to avoid the debt, as described in "the solution adopted was one of the best considering the restrictions of available infrastructure, deadline and technical knowledge" and "I believe that it would not be possible to avoid this TD, as the main cause is the lack of knowledge of ReactJS. Also, the type of knowledge needed is awfully specific and not something you can get through documentation or courses. It is necessary to use ReactJS in practice to solve real problems to understand what the limitations/difficulties of native state control are, and only then look for alternatives."

The *documentation issues (lack of or non-updated)* PAR is related to insufficient or outdated software documentation limiting the performance of software teams activities, for example, "when youre building on a platform that does not have sufficient documentation, you may not know how your code will work with the limits until a prototype is tested" and "documentation for the customer must always be up to date." The *lack of concern about maintainability* PAR refers to the lack of concern for software maintenance, as we can observe in "maintainability is seldom a developer's concern." The *lack of good technical solutions* PAR is related to the lack of good solutions that could be chosen by the software team, for example, "at the time the decision was made, we investigated all possible solutions and came up with the least bad solution." The *lack of process maturity* PAR means that process followed by the team is not enough to curb TD items, as indicated in "management and process were not mature enough to prevent TD."

Analyzing the top ten PARs, we noticed that the PARs have distinct types. For example, the PARs *short deadline*, *pressure for results*, *lack of technical knowledge*, and *lack of good technical solutions* are PARs out of the teams control (an **impediment**). On the other side, *ineffective management*, *lack of predictability in the software development*, *requirements change*, *documentation issues (lack of or non-updated)*, *lack of concern about maintainability*, and *lack of process maturity* are **decision factors** considered by the team for TD non-prevention. Therefore, the ***type*** dimension for PARs groups reasons for TD non-prevention that are a teams decision or are out of the teams control and has the following values: **impediment** and **decision factor**.

Table 4.6 presents the identified types of PARs, reporting the **types name**, the number of unique PARs for TD non-prevention cited (**#PAR**), and the total number (i.e., count) of PARs (**#CPAR**) cited in each type. #CPAR also indicates the number of projects that considered that PAR for justifying the non-prevention of TD items in each PAR type. Column **%PARP** corresponds to the percentage of #CPAR in relation to the total of all projects, indicating how frequently each PAR type was considered in software projects. Appendix Table B.6 presents, for each type, the PARs for TD non-prevention ranked by number of citations.

Table 4.6 Types of PAR for TD non-prevention

Type of PAR for TD non-prevention	#PAR	#CPAR	%PARP
Decision factor	10	27	55%
Impediment	13	34	69%

Caption:
#PAR - Count of unique cited PAR for TD non-prevention.
#CPAR - Count of practice avoidance reasons for TD non-prevention.
%PARP - Percentage of CPAR in relation to the total of all projects (49).

We found almost the same quantity of PAR in each type, but the quantity of citations differs among them. Impediments are the most common PAR type for explaining TD non-prevention and can be considered in 69% of the projects. Decision factors, in turn, can be considered in 55% of the projects.

Regarding the dimension ***nature*** (technical or managerial), considering only the top 10 cited PARs, the **managerial subset** is composed of *short deadline*, *ineffective management*, *lack of predictability in the software development*, *pressure for results*, *lack of concern about maintainability*, and *lack of process maturity*, representing 69% of all projects (column %PARP at Table 4.5). The **technical subset** is composed of *requirements change*, *lack of technical knowledge*, *documentation issues (lack of or non-updated)*, and *lack of good technical solutions*, representing 24% of all projects (column %PARP at Table 4.5). Appendix Table B.7 presents, for each nature, the PARs for TD non-prevention ranked by number of citations.

Table 4.7 presents the relation between the types and natures of all PARs for TD non-prevention. We have 13 managerial PARs, representing 69% of the total number of citations, and 10 technical PARs, corresponding to 31% of the total number of citations. This result clearly indicates that managerial issues are decisive in improving TD prevention initiatives.

Table 4.7 Relation between type and nature of PARs for TD non-prevention

Type of PARs for TD non-prevention	Nature of PAR for TD non-prevention	
	Technical	Managerial
Decision factor	4 (15%)*	6 (30%)
Impediment	6 (16%)	7 (39%)
Total	10 (31%)	13 (69%)

*The number in parentheses represents the percentage of the total number (i.e., count) of PARs cited in each nature in relation to the total of all cited PARs.

Regarding the ***category*** dimension, we divided the PARs into the following categories:

- **Development issues**: groups PARs related to software development activities. This category is composed of five PARs, like *lack of good technical solutions*, *legacy system difficult to heal*, and *requirements change.*

- **External factors**: encompasses three PARs associated with factors that software teams cannot control. *Differences among stakeholders*, *not sure if client will accept it*, and *pressure for results* are the PARs in this category.

- **Lack of knowledge**: includes PARs related to the need for technical knowledge. This category is composed of two PARs (*lack of information* and *lack of technical knowledge*).
- **Methodology**: refers to PARs associated with the process followed by the team. *Development teams interdependency*, *documentation issues (lack of or non-updated)*, *lack of predictability in the software development*, and *lack of process maturity* are the PARs in this category.
- **Organizational**: groups PARs related to organizational decisions. This category has three PARs: *lack of financial resources*, *lack of qualified professionals*, and *restrictions on available infrastructure.*
- **People**: includes PARs associated with team characteristics. *Lack of experience* and *people issues* are the PARs in this category.
- **Planning and management**: has PARs associated with managerial activities. This category includes four PARs: *debt close to the project end*, *ineffective management*, *lack of concern about maintainability*, and *short deadline.*

Table 4.8 presents the categories of PARs for TD non-prevention, reporting the **categorys name**, the number of unique PARs cited (**#PAR**), and the total number (i.e., count) of PARs (**#CPAR**) cited in each category. #CPAR also indicates the number of projects that considered a PAR for explaining the non-prevention of TD items in each category. Lastly, the column **%PARP** corresponds to the percentage of #CPAR in relation to the total of all projects. Appendix Table B.8 presents, for each category, the PARs for TD non-prevention ranked by number of citations.

Table 4.8 Categories of PARs for TD non-prevention

Category of PARs for TD non-prevention	**#PAR**	**#CPAR**	**%PARP**
Planning and management	4	24	49%
Development issues	5	11	22%
Methodology	4	10	20%
External factors	3	6	12%
Lack of knowledge	2	4	8%
Organizational	3	4	8%
People	2	2	4%

Caption:
#PAR - Count of unique cited PAR for TD non-prevention.
#CPAR - Count of PARs for TD non-prevention.
%PARP - Percentage of CPAR in relation to the total of all projects (49).

The *planning and management* category has the greatest number of PARs, followed by *development issues* and *methodology.* One can expect that *development issues* are decisive for TD non-prevention, this category impacted 22% of the projects. However, the categories *planning and management* and *methodology* impacted 49% and 20% of the projects, respectively. The data indicates that software teams should pay close attention to planning, management, and methodology issues when concerned about TD non-prevention.

4.1.2.1 PAR for TD Non-prevention per Type of Debt. Figure 4.7 presents the relationship among the **types of debt** and the top ten PARs, while Figure 4.8 shows the number of times that each relationship occurred. The PAR *short deadline* could be considered to explain the non-prevention of all types of debt, revealing that software teams can facilitate their prevention activities by improving their planning. We can observe that the PARs *ineffective management*, *requirements change*, and *pressure for results* could be considered to justify the non-prevention of code and design debt items. As code and design debt items share almost the same PARs, software practitioners can define integrated strategies to prevent these TD types.

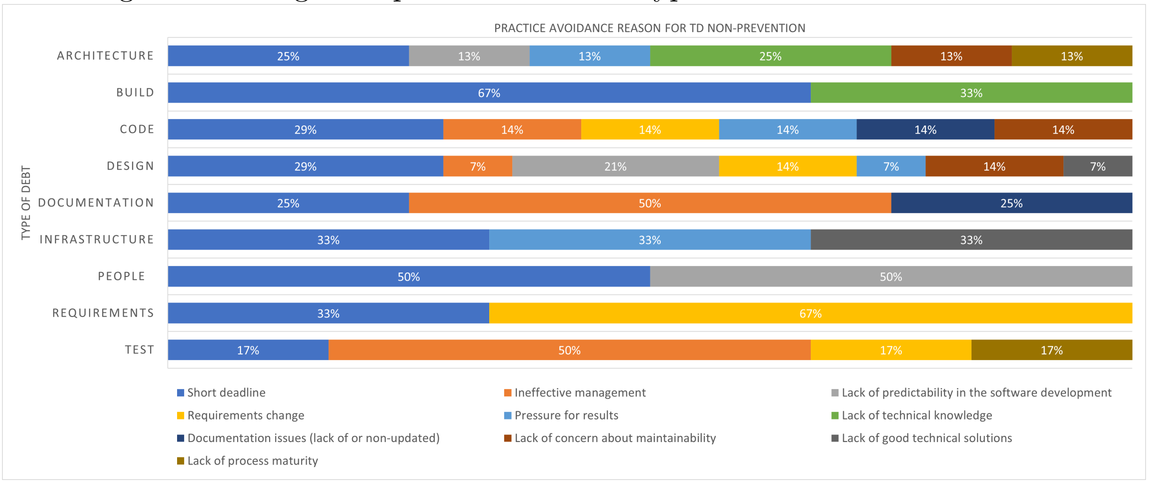


Figure 4.7 Relationship among types of debt and the top 10 PARs for TD non-prevention - Percentage values

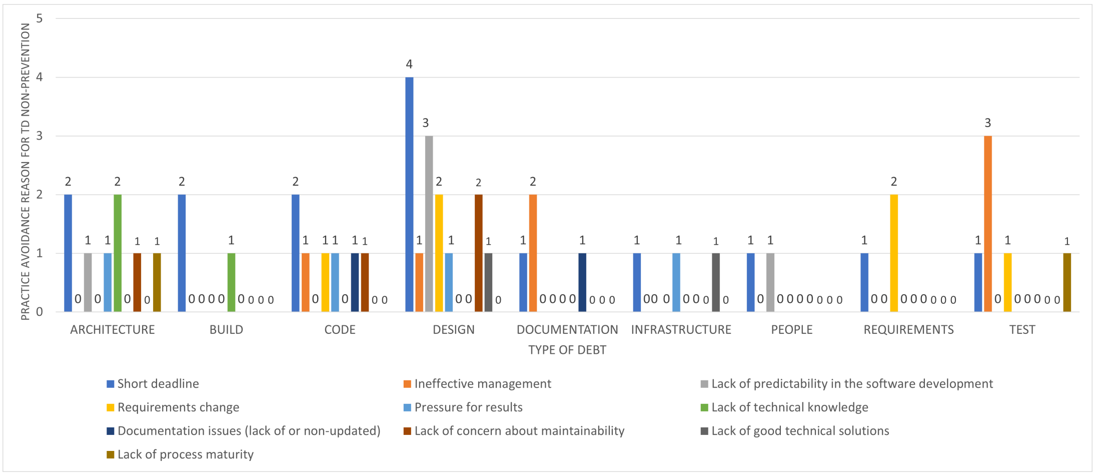


Figure 4.8 Relationship among types of debt and the top 10 PARs for TD non-prevention - Absolute values

Figure 4.9 shows the relationship between types of debt and **types of PARs**. We can see that architecture, design, infrastructure, and process debt items are not prevented due to impediments, while documentation, requirements, and test debt items are not avoided due to decision factors. This indicates that all stakeholders (team, organization, and customer) need to change their mindset regarding the importance of preventing TD. Figure 4.10 shows the number of times that each relationship occurred.

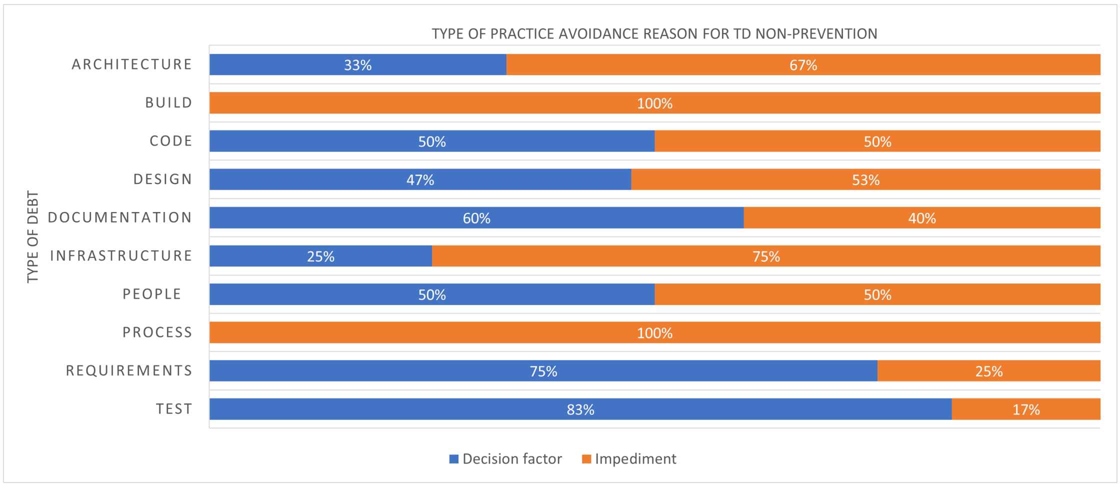


Figure 4.9 Relationship among types of debt and types of PARs for TD non-prevention - Percentage values

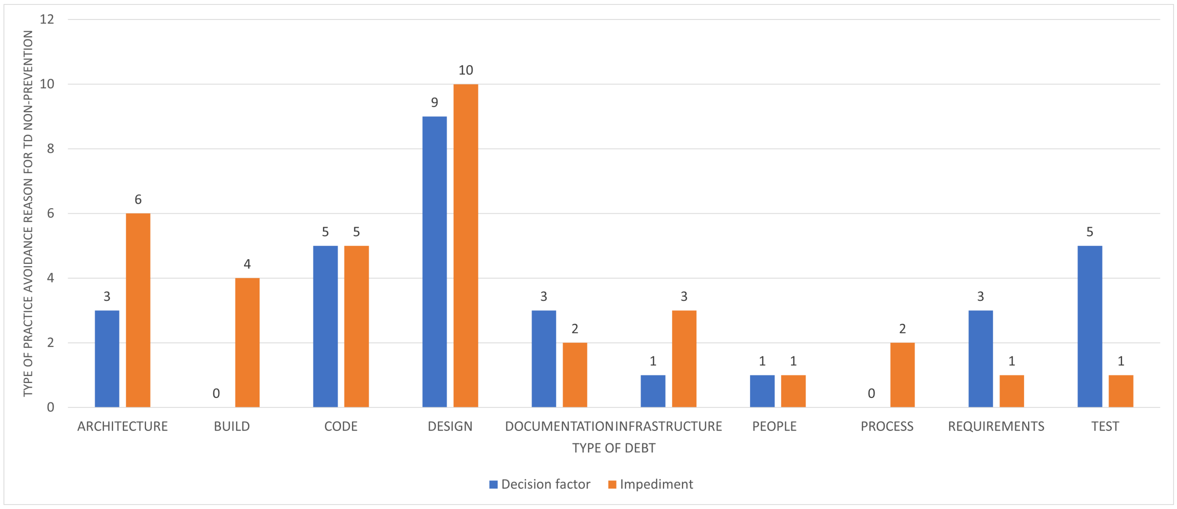


Figure 4.10 Relationship among types of debt and types of PARs for TD non-prevention - Absolute values

Figure 4.11 shows the relationship among the types of debt and **categories of PARs**, indicating how frequently PARs from these categories are considered to explain the non-prevention of each type of debt. *Planning and management* category encompasses the PARs of all but process debt items. PARs from the *development issues* category is most commonly considered to justify the non-prevention of requirements and infrastructure debt items, while process debt items are not avoided due to PARs from the *external factors* and *organizational* categories. Lastly, people debt items are not prevented due to PARs from the *methodology* category. Figure 4.12 shows the number of times that each relationship was found.

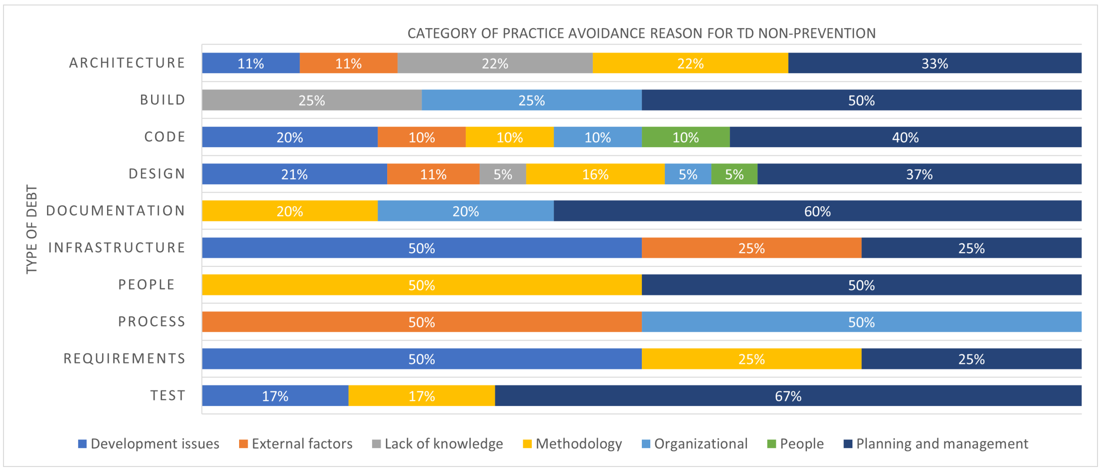


Figure 4.11 Relationship among types of debt and categories of PARs for TD non-prevention - Percentage values

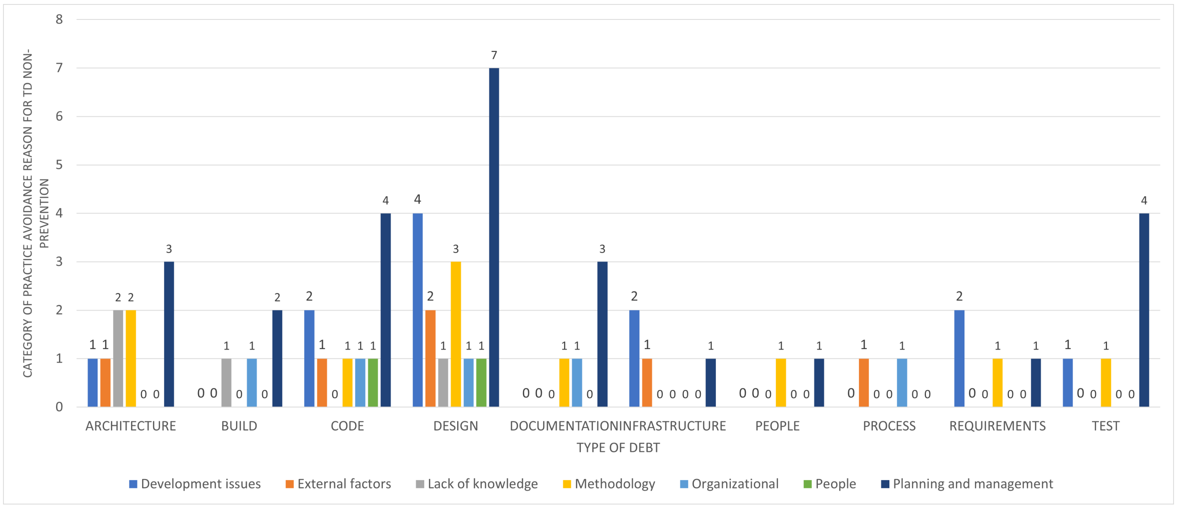


Figure 4.12 Relationship among types of debt and categories of PARs for TD non-prevention - Absolute values

4.2 DISCUSSION

In this section, we answer RQ1 and compare the findings with those reported in the related work.

4.2.1 Answering Research Question 1

We identified 89 prevention-related practices reported by practitioners (RQ1.1). The practices *well-defined requirements*, *adoption of good programming practices*, and *better project management* were the most cited, indicating that improvements applied to software requirements, code, and management are necessary to increase the teams ability to avoid debt items. Also, we found that the practices can directly prevent debt items (prevention actions) or support TD prevention initiatives (enabling practices). We categorized all practices into seven categories related to software development concerns. The categories *planning and management* and *methodology* stand out, encompassing the greatest number of prevention-related practices. Lastly, practices from *planning and management* category can be used to prevent all analyzed types, indicating that investing effort in performing its practices can be a good starting point to improve TD prevention initiatives.

Regarding PARs for TD non-prevention (RQ1.2), we found 23 of them. *Short deadline*, *ineffective management*, *lack of predictability in the software development*, and *requirements change* were the most cited. We realize that PARs can be either an external factor outside the control of the team (an impediment) or a decision of the team itself (a decision factor), revealing that the non-prevention of TD can be a decision of the team or some external factor affecting the project, such as the organization or the customer, among others.

All PARs were grouped into seven categories. The categories *planning and management*, *development issues*, and *methodology* have the largest number of PARs. It reveals that decisions taken in software development and methodology and at the managerial level are decisive for TD prevention. Lastly, we found that the *planning and management* category encompasses the PARs that can explain the non-prevention of almost all investigated types of debt.

4.2.2 Comparison to Related Work

This section presents a comparison between our findings with the ones reported in related work. The comparison was performed on two levels: practices and categories of practices.

The practices for **enabling TD prevention** were found both in our work and in at least some other past studies are *adequate technical management*, *appropriate tasks allocation*, *good communication between stakeholders*, *implementation of a TD identification strategy*, *implementation of a TD management strategy*, *implementation of a TD payment strategy*, *improve schedule to include tests and bug fixing*, *prioritization of TD payment*, *training*, *using agile practices*, *well-defined effort estimation methods*, and *well-defined metrics*. The **TD prevention actions** reported in our study and past work are *adoption of good programming practices*, *code review*, *continuous integration*, *creating tests*, *following well-defined project process*, *improve documentation*, *improve requirements elicitation*, *improving software development process*, *improving tests*, *prioritization of test and documentation*, *refactoring*, *requirement validation*, *requirements changes tracking*, *use the most appropriate version of the technology*, *using good design practices*, and *well-defined requirements*.

Regarding the distinctions, we identified that related work mainly found practices used to improve the *definition of done* and guarantee the *continuous improvement*, i.e., practices more related to process followed by the team. On the other hand, we found more practices associated with managerial and technical aspects.

Figure 4.13 shows the distinctions and overlaps between our works unique TD prevention-related practices and the related work. The complete comparison is available in Appendix Table B.9.

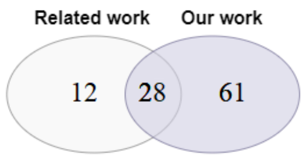


Figure 4.13 Number of unique TD prevention-related practices found in our work and the related work

Table 4.9 presents the comparison between the categories we identified in our work and the categories reported by Li, Avgeriou and Liang (2015) and Bonfim and Benitti (2022). The comparison result is marked as none, partial (indicating that our work has practices that are not considered in the related work and vice-versa), or total (meaning that all practices from a category identified in our work are considered in the related work). We can notice that we confirmed the categories *development issues*, *methodology*, *people*, and *planning and management*. The other categories (*infrastructure*, *internal quality issues*, and *organizational*) were only identified in our work.

Table 4.9 Comparison of related work on categories of TD prevention-related practices

	Categories from		**Overlapping**
Our study	**(LI; AVGERIOU; LIANG, 2015)**	**(BONFIM; BENITTI, 2022)**	**degree**
Development issues	Architecture decision making support	-	Partial
Infrastructure	-	-	None
Internal quality issues	-	-	None
Methodology	Development process improvement	Supporting the implementation of requirements specification	Partial
		Implementing requirements validation	Partial
	-	Meeting requirements elicitation	Partial
	-	Helping requirements analysis	Partial
Organizational	-	-	None
People	Human factors analysis	-	Total
Planning and management	Lifecycle cost planning	Assisting in requirements management	Partial

In summary, our findings on TD prevention-related practices and their categories complement and extend the set of information already reported in the technical literature.

4.3 CONCLUDING REMARKS

This chapter reported the software practitioners perceptions of TD prevention, discussing the practices cited for avoiding TD items and the PARs mentioned for explaining TD non-prevention. It identified distinct types of practices and PARs, revealing how the practices can be used in TD prevention initiatives and the source of PARs that can support software practitioners in improving these initiatives. Relationships between practices and TD types, and PARs and TD types were also investigated, helping to know specified practices or PARs to specific types of debt. Threats to validity were previously discussed in Chapter 3.

The following chapter presents the state of TD monitoring practice based on the analyses of the InsighTD data set.

Chapter

5

This chapter presents the state of the practice of TD monitoring obtained from analyses of the InsighTD data set.

TECHNICAL DEBT MONITORING'S STATE OF PRACTICE

Technical debt (TD) monitoring is a central activity that allows software teams to track unresolved debt items, identifying changes in their cost and benefit during the project life cycle (RIOS; MENDONÇA; SPINOLA, 2018). For example, practitioners can continuously compare the actual costs of a debt item with the current benefits of its presence by using a cost/benefit analysis (AMPATZOGLOU *et al.*, 2015).

Despite current research efforts on the topic, the discussion around TD monitoring deserves a more comprehensive investigation. The main reason is that TD monitoring is crucial in preventing TD from spreading through the project, leading to increased development costs or, in some extreme cases, project termination (RIOS *et al.*, 2020). This is because the software development process is a chain of related activities and decisions, where one activity relies on the artifacts produced by others. If the artifact produced has TD, it is passed on to all activities that depend on it (MANDIĆ *et al.*, 2021). Thus, it is essential to continuously evaluate whether the costs of TD overcome its benefits and to act accordingly.

Furthermore, we know that many factors lead to situations favorable for injecting TD during the software development and maintenance process, e.g., factors like time pressure, lack of knowledge, etc. (MANDIĆ *et al.*, 2021; RAMAČ *et al.*, 2022a). Factors can be associated with distinct aspects of an organization, as well as factors that are external to the organization. Therefore, to gain insights into TD monitoring practices and monitoring avoidance reasons, we also need to look at the associated contexts of the practices. For example, infrastructure, people, processes, or external factors can be context.

This chapter aims *to investigate, from the point of view of software development practitioners, which practices can be used to monitor TD items and what Practice avoidance reasons (PARs) would justify the non-monitoring of these items.* To achieve this goal, we define the **RQ2: How have software development teams monitored technical debt items on their projects?** And its sub-questions: RQ2.1: What are the leading practices for monitoring TD items in software projects? And RQ2.2: What are the leading practice avoidance reasons (PARs) to explain the non-monitoring of TD items? To answer these RQs, we use and analyze the InsighTD data set (as described in Chapter 3). This chapter is based on the paper: Hearing the Voice of Software Practitioners on Technical Debt Monitoring: Understanding Monitoring Practices and the Practices Avoidance Reasons (FREIRE *et al.*, 2022).

The main contributions of this Chapter are:

- An up-to-date, comprehensive list of practices related to TD monitoring ranked by the number of citations by software practitioners.
- An up-to-date, comprehensive list of PARs ranked by the number of their citation by software practitioners.
- A categorization of TD monitoring and PARs concerning their relationship to software planning, development, management, and methodology issues.
- A definition of types of practices and PARs about these contributions in TD monitoring initiatives.
- A definition of the nature of practices and PARs concerning technical and managerial activities.
- The relationships between types of debt and practices, and types of debt and PARs.
- A comparison of practices and PARs found in this study to those reported in the technical literature.

Section 5.1 presents the perception of software practitioners on TD monitoring. Section 5.2 discusses the results and compares them with those reported by related work. Lastly, Section 5.3 offers the concluding remarks.

5.1 PERCEPTION OF INSIGHTD'S PRACTITIONERS ON TD MONITORING

This section presents our findings from the InsighTD data set analyses concerning TD monitoring (answers given to Q24 and Q25). We organize these findings per RQ.

5.1.1 RQ2.1: What are the leading practices for monitoring TD items in software projects?

In Q24, 53% of the participants indicated that the TD item described in Q13 was monitored, and 75% of those described how TD was monitored in Q25. For answering RQ2.1, we used this subset of responses, comprised of 259 answers.

We identified 46 TD monitoring-related practices. Table 5.1 summarizes the ten most commonly cited ones. Appendix Table C.1 presents the identified monitoring-related practices and quotes from participants. Table 5.1 reports the **monitoring-related practice name** and the total number (i.e., count) of citations (**#CMRP**). #CMRP also indicates the number of projects that used that practice. Column **%MRPP** presents the percentage of #CMRP concerning the total of projects, revealing how frequently each practice was used in software projects. We notice that the *TD item backlog* is the most cited practice and has been used in 13% of the projects, followed by *use of tools*, *team meetings*, and *improving software development process* that has been used in 12%, 9%, and 8% of the projects, respectively.

Table 5.1 Top 10 cited TD monitoring-related practices

NO	TD Monitoring-related practice	#CMRP	%MRPP
1st	TD item backlog	34	13%
2nd	Use of tools	31	12%
3rd	Team meetings	23	9%
4th	Improving software development process	20	8%
5th	Refactoring	18	7%
6th	Improving tests	17	7%
7th	Code review	16	6%
8th	Communicating the stakeholders of TD items	16	6%
9th	Tracking TD items	12	5%
10th	TD management plan	11	4%

Caption:
#CMRP - Count of monitoring-related practices.
%MRPP - Percentage of CMRP in relation to the total of all projects (259).

In the context of this study, *TD item backlog* means that software teams have included TD items in their task list to prioritize those items and define the most appropriate time for paying them off, as described in the quotes extracted from the answers: "keep a list of all TD and regularly determine how to eliminate it" and "identified as a task and placed in the backlog for later prioritization." The practice *use of tools* indicates that TD monitoring activities can be supported by tools to extract metrics from the system, as indicated in the following answers: "we have to use tool to monitor it, this is an ongoing process" and "use of tools like SonarQube." *Team meetings* means that software teams have used meetings to track the evolution of the cost and the benefit of TD items over the project, as can be observed in the following answers from the participants: "regular meeting, usually weekly held that would track the progress" and "(...) the use of meetings every day to align the understanding of the development team about the solution that was being developed (...)." The practice *improving software development process* refers to the improvements applied to the process adopted in the project, as we can observe in "started with a training process, application of agility and the adoption of the best development practices" and "get started with programming best practices."

The practices *refactoring* and *improving tests* have been used in 7% of the projects, while *code review* and *communicating the stakeholders of TD items* in 6%. *Refactoring* means changes in the software to improve its internal quality, as we can notice in "refactoring took place before the development of other services took place" and "refactoring when appropriate and time allowed." *Improving tests* refers to applying more tests in the system or adding other testing activities, as we can observe in "a test plan was made in conjunction with the user and they were run again" and "(...) we began to incorporate more end-to-end tests to gate delivery." *Code review* means the checking of source code for identifying mistakes or improvement points, "every pull request changing the login page must be looked at by least two architects" and "(...) code reviews done during the sprint." *Communicating the stakeholders of TD items* means that stakeholders are aware of the TD items existing in the project, as we can observe in "the client is perfectly aware of it" and "it was monitored in the sense that the team had a shared awareness and discussed its ongoing impact."

Lastly, the practices *tracking TD items* and *TD management plan* have been used in 5% and 4% of the projects, respectively. *Tracking TD items* refer to providing a view of all TD items existing in the project, as we can notice in "work considered part of the technical debt was tracked" and "through the task/issue tracking system." *TD management plan* aims to define a plan to deal with TD items, as we can observe in "a schedule was developed specifically to address the case" and "we were aware of what we are doing and planned to "pay the debt" later on in maintenance stage of the project."

Looking at the top ten monitoring practices, we noticed that *TD item backlog*, *team meetings*, *communicating the stakeholders of TD items*, and *tracking TD items* are practices that allow the direct monitoring of TD items. In contrast, others provide distinct types of support to TD monitoring initiatives. Thus, we identify one dimension of TD monitoring practices, ***type***, having the following values:

- **Monitoring action**: refers to practices directly related to TD monitoring, such as *TD item backlog*, *communicating the stakeholders of TD items*, and *tracking TD items*.
- **Enabling TD monitoring**: includes practices that improve the capacity of development teams to monitor debt items. Among them, we have *use of tools*, *improving software development process*, and *assign team for TD monitoring*.
- **TD prevention**: refers to practices intended to avoid potential TD items from being incurred. Some examples are *improving tests*, *code review*, and *qualified professionals*.
- **TD identification**: groups practices that support the identification of TD items in the project. Only *identifying TD items* and *use metrics for TD identification* have this type.
- **TD payment**: includes practices for repaying TD items. We found only the practices *focusing on TD payment*, *improve documentation*, and *refactoring* within this type.

Table 5.2 shows the identified types, reporting the **types name**, the number of unique monitoring-related practices cited (**#MRP**), and the total number (i.e., count) of monitoring-related practices (**#CMRP**) cited in each type. #CMRP also indicates the number of projects that used that practice. Column **%MRPP** corresponds to the percentage of #CMRP over the total of all projects, showing how frequently each type was used in software projects. Appendix Table C.2 presents, for each type, the monitoring-related practices ranked by number of citations.

Table 5.2 Types of TD monitoring-related practices

Type of monitoring-related practice	#MRP	#CMRP	%MRPP
Monitoring action	17	127	49%
Enabling TD monitoring	15	97	38%
TD prevention	9	47	18%
TD payment	3	29	11%
TD identification	2	12	5%

Caption:
#MRP - Count of unique cited monitoring-related practices.
#CMRP - Count of monitoring-related practices.
%MRPP - Percentage of CMRP in relation to the total of all projects (259).

Monitoring action is the most cited type by the participants, being used in 49% of the projects. The type *enabling TD monitoring* was also commonly cited by the respondents and was used in 38% of the projects. The types *TD prevention*, *TD payment*, and *TD identification* were used in 18%, 11%, and 5% of the projects. This result indicates that, besides using monitoring actions, software practitioners have applied other types of practices for supporting TD monitoring initiatives. TD monitoring is part of a bigger process that encompasses TD prevention, payment, identification, and enabling TD monitoring.

Regarding the dimension *nature* (**technical** and **managerial**), among the top 10 cited TD monitoring-related practices, the **technical subset** comprises *use of tools*, *refactoring*, *improving tests*, and *code review*, representing ~32% (column %MRPP at Table 5.1). The **managerial subset** is composed of *TD item backlog*, *team meetings*, *improving software development process*, *communicating the stakeholders of TD items*, *tracking TD items*, and *TD management plan*, representing ~45% of all projects (column %MRPP at Table 5.1). We can see managerial practices are more commonly used in TD monitoring initiatives. Appendix Table C.3 presents the TD monitoring-related practices of each nature ranked by number of citations.

Table 5.3 presents the relation between types and natures of all TD monitoring-related practices. Most monitoring actions are managerial, while most others are technical.

Table 5.3 Relation between type and nature of monitoring-related practices

Type of monitoring-related practice	Nature of monitoring-related practice	
	Technical	Managerial
Monitoring action	2 (1%)*	15 (39%)
Enabling TD monitoring	7 (15%)	8 (16%)
TD prevention	7 (13%)	2 (2%)
TD payment	2 (8%)	1 (2%)
TD identification	2 (4%)	0 (0%)
Total	20 (41%)	26 (59%)

*The number in parentheses represents the percentage of the total number (i.e., count) of monitoring -related practices cited in each nature in relation to the total of all cited practices (312).

We also grouped the set of monitoring-related practices into seven categories:

- **Development issues**: groups practices that are applied during the implementation of the software. Only the practice *improve documentation* composes this category.

- **Infrastructure**: includes two practices related to tools, technologies, and development infrastructure. In this category, we have *infrastructure monitoring* and *use of tools.*

- **Internal quality issues**: encompasses four practices that can be employed to address limitations that compromise the internal quality of the software, such as *identify the worst debt areas*, *identifying TD items*, *refactoring*, and *understanding the cause of TD item.*

- **Methodology**: refers to 16 practices related to processes a software team follows. Examples of practices in this category are *improving software development process* and *improving tests.*

- **Organizational**: includes four practices related to organizational decisions. The practices *knowledge sharing*, *qualified professionals*, *team restructuring*, and *training* compose this category.

- **People**: groups two practices (*communicating the stakeholders of TD items* and *team meetings*) related to the members of software development teams.

- **Planning and management**: refer to 17 practices associated with management activities. Among them, we highlight *TD item backlog*, *tracking TD items*, and *TD management plan.*

Table 5.4 presents the categories, reporting the **categorys name**, the number of unique monitoring-related practices cited (**#MRP**), and the total number (i.e., count) of monitoring-related practices (**#CMRP**) cited in each category. We counted repeated practices in a participants response as a single count. Column #CMRP also indicates the number of projects that used a practice for monitoring TD items in each category. Lastly, the column **%MRPP** corresponds to the percentage of #CMRP over to the total of all projects. Appendix Table C.4 presents, for each category, the TD monitoring-related practices ranked by number of citations.

Table 5.4 Categories of monitoring-related practices

Category of monitoring-related practices	**#MRP**	**#CMRP**	**%MRPP**
Planning and management	17	115	44%
Methodology	16	84	32%
People	2	39	15%
Infrastructure	2	32	12%
Internal quality issues	4	27	10%
Organizational	4	9	3%
Development issues	1	6	2%

Caption:
#MRP - Count of unique cited monitoring-related practices.
#CMRP - Count of monitoring-related practices.
%MRPP - Percentage of CMRP in relation to the total of all projects (259).

By analyzing the table, we notice that the categories *planning and management* and *methodology* concentrate the most significant number of practices. Their practices have been used in 44% and 32% of the projects, respectively.

5.1.1.1 TD Monitoring Practice per Type of Debt. Figure 5.1 presents the relationship among the types of debt and the top 10 TD monitoring-related practices, while Figure 5.2 shows the absolute values of these relationships, i.e., the quantity of times that each relationship was found. All practices from the top 10 were applied to monitor code and design debt items. For architecture and test debt, only the practices *code review* and *improving software development process* were not applied, respectively. The practices *improving tests*, *communicating the stakeholders of TD items*, and *TD management plan* were not applied for supporting the monitoring of requirements debt. Documentation debt items were monitored using five practices, while software practitioners have used four practices for people, defect, and process debt. Infrastructure debt was monitored by using three practices. Lastly, build, service, usability, and versioning debt items were only monitored using two practices.

The practice *TD item backlog* is the most used practice for monitoring architecture, build, defect, service, usability, and versioning debt items. Further, the *use of tools* is the most used for enabling the monitoring of architecture, code, and design debt items. Requirements and process debt items are commonly monitored by *team meetings*, while *improving software development process* and *tracking TD items* are used for monitoring documentation and infrastructure debt items, respectively. Lastly, people and test debt items are commonly monitored by *communicating the stakeholders of TD items* and *improving tests*.

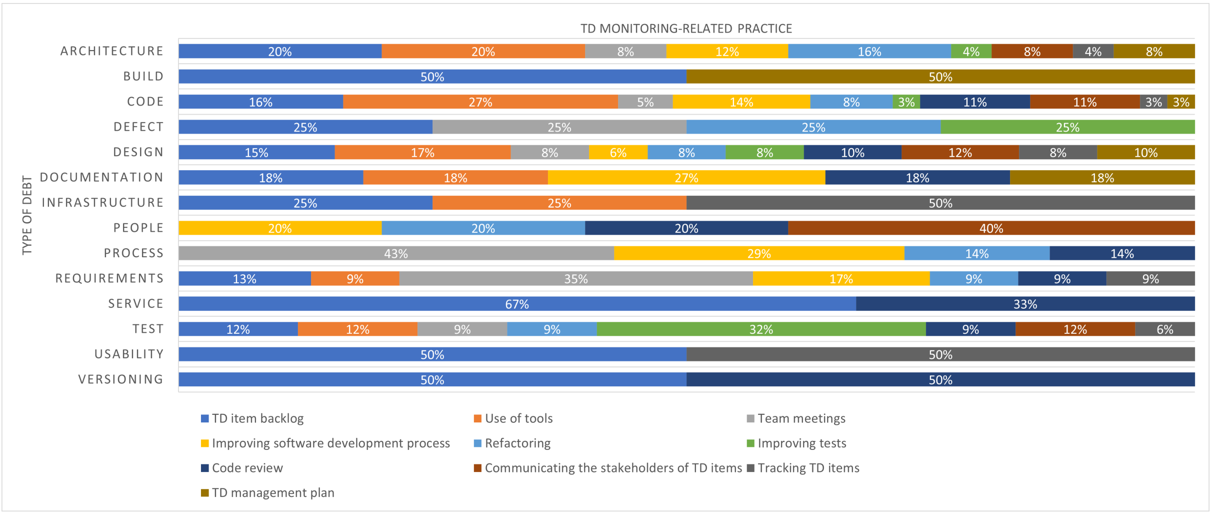


Figure 5.1 Relationship among types of debt and the top 10 TD monitoring-related practices - percentage values

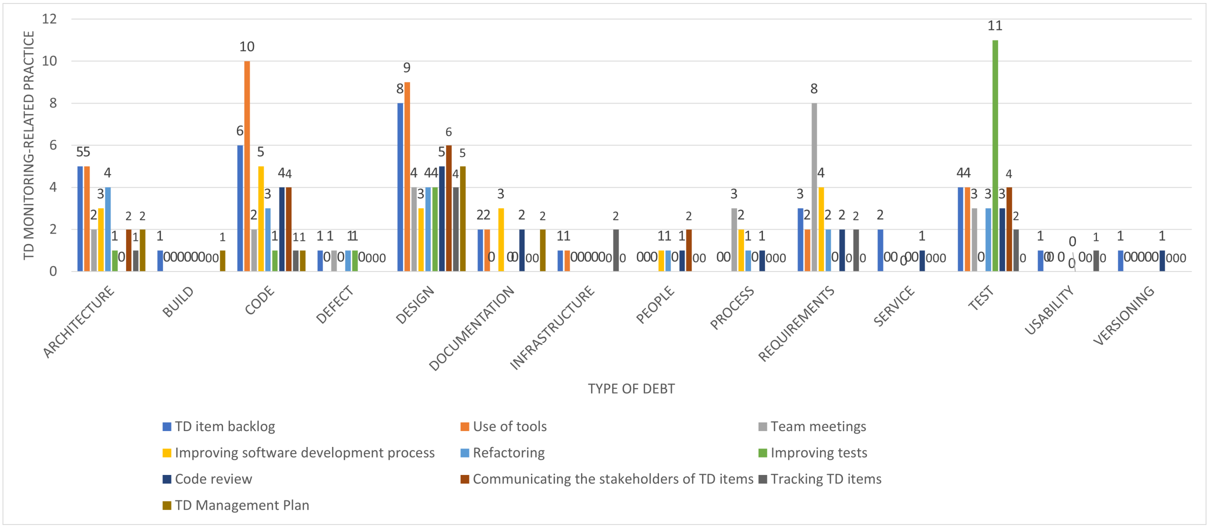


Figure 5.2 Relationship among types of debt and the top 10 TD monitoring-related practices - absolute values

From Table 5.5, we can observe that only some types of debt were directly monitored. For example, usability and versioning debt have few types of monitoring-related practices. At the same time, monitoring actions (marked in gray) were commonly cited for paying off code, design, requirements, test, and architecture debt items, evidencing that software practitioners have interested in following TD items affecting technical activities. Practitioners have used practices to enable TD monitoring (marked in orange) and to prevent (marked in blue) and pay off (marked in green) for TD items in code-related, requirements, and test activities.

Table 5.5 Relationship among the types of technical debt monitoring-related practices and types of debt

Type of debt	**Type of TD monitoring-related practice**				
	Monitoring action	**Enabling TD monitoring**	**TD prevention**	**TD payment**	**TD identification**
Architecture debt	13	14	3	8	2
Build debt	1	2	0	0	1
Code debt	22	20	6	3	1
Defect debt	1	3	1	1	2
Design debt	20	36	12	5	2
Documentation debt	9	8	4	4	0
Infrastructure debt	1	7	1	0	0
People debt	2	2	1	2	0
Process debt	3	3	1	1	0
Requirements debt	15	16	3	3	2
Service debt	1	4	1	0	0
Test debt	13	21	15	3	2
Usability debt	0	2	0	0	0
Versioning debt	0	1	4	0	0

We also investigated the relationship between types of TD and the categories of TD monitoring-related practices. Figure 5.3 shows this relationship, indicating that the category *planning and management* encompasses practices used for monitoring almost all types of debt, except people and process debt items. Practices from the category *methodology* are used for monitoring all but build, infrastructure, and usability debt items. At the same time, practices from *internal quality issues* are used for monitoring all but while documentation, service, usability, and versioning debt items. The number of times that each relationship occurred is shown in Figure 5.4.

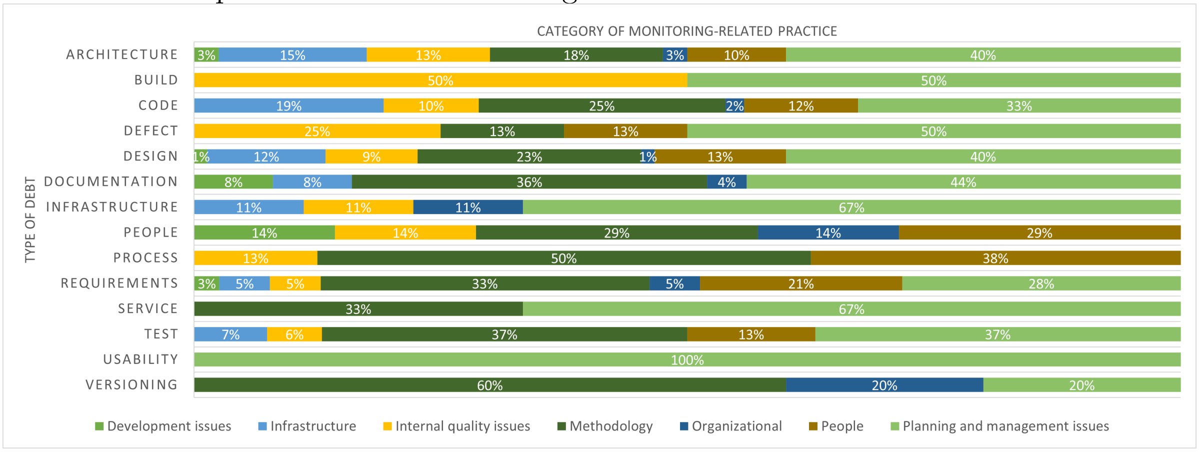


Figure 5.3 Relationship among types of debt and categories of TD monitoring-related practices - percentage values

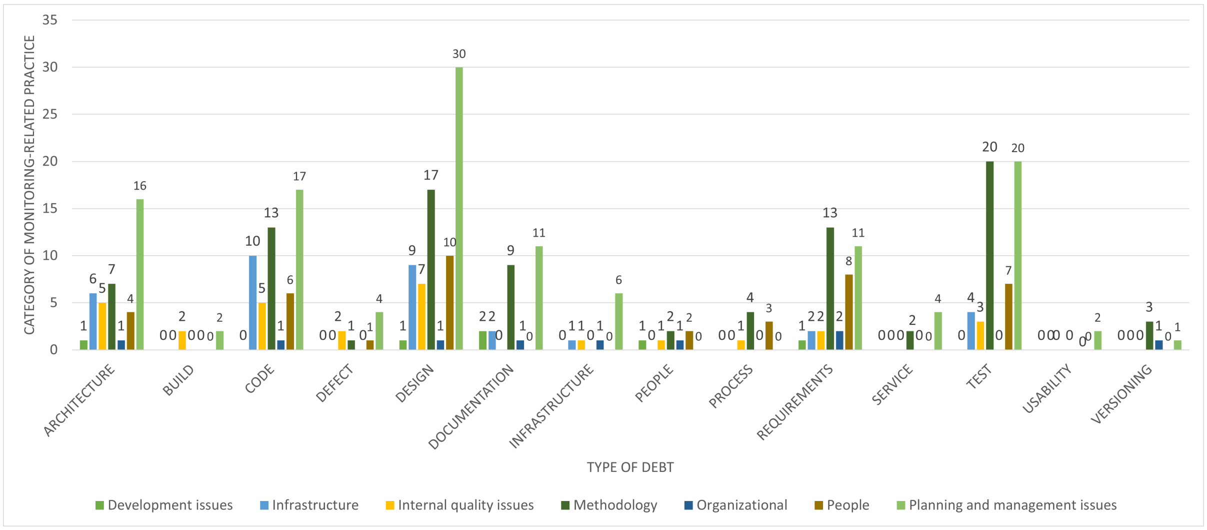


Figure 5.4 Relationship among types of debt and categories of TD monitoring-related practices - absolute values

5.1.2 RQ2.2 What are the leading practice avoidance reasons (PARs) to explain the non-monitoring of TD items?

In total, 47% of the participants did not monitor the TD item described in Q13. Of them, 64% explained in Q25 why the TD was not monitored. We used this set of 197 responses to answer RQ2.2.

We identified 35 PARs for TD non-monitoring. Table 5.6 presents the ten most commonly cited ones. All the identified PARs and quotes from participants are available in Appendix Table C.5. Table 5.6 reports the **PAR name** and the total number (i.e., count) of citations (**#CPAR**). #CPAR also indicates the number of projects that used a PAR for justifying the TD non-monitoring. Column **%PARP** presents the percentage of #CPAR to the total of all projects, revealing how frequently each PAR was considered in software projects.

Table 5.6 Top 10 practice avoidance reasons (PARs) for TD non-monitoring

NO	Practice avoidance reason (PAR)	#CPAR	%PARP
1st	Lack of interest	44	22%
2nd	Focusing on short term goals	33	17%
3rd	Lack of time	29	15%
4th	Lack of knowledge on TD	23	12%
5th	Lack of understanding about the impact of the debt	12	6%
6th	Lack of organizational culture	8	4%
7th	Lack of resources	8	4%
8th	Lack of TD monitoring process	7	4%
9th	Lack of specific team	6	3%
10th	React when becoming a problem	5	2%

Caption:
#CPAR - Count of practice avoidance reason for TD non-monitoring.
%PARP - Percentage of CPAR in relation to the total of all projects (197)

Lack of interest is the most cited PAR, being considered in 22% of the projects. The other most mentioned PARs are *focusing on short-term goals*, *lack of time*, and *lack of knowledge on TD*, which were considered in 17%, 15%, and 12% of the projects, respectively. *Lack of interest* refers to organizations or project managers that do not purse monitoring TD items, as indicated in the following answers: "little interest of the company to correct this type of situation" and "management did not care." The PAR *focusing on short-term goals* means that software teams have other priorities in their projects, as informed in "not in the priority pipeline" and "it was not critical for the success of the project." *Lack of time* refers to software projects that do not have time for TD monitoring, as we can notice in "because deadlines are tight" and "the project timeline didn't allow it." The PAR *lack of knowledge on TD* means that software teams do not monitor debt items because they did not have knowledge on TD, such as we can observe in the answers "because the concept of technical debt was not yet applied in the company" and "there is no knowledge about TD."

The PAR *lack of understanding about the impact of the debt* was considered in 6% of the projects, while *lack of organizational culture*, *lack of resources*, and *lack of TD monitoring process* were considered, each one, in 4% of the projects. *Lack of understanding about the impact of the debt* indicates that although software teams have identified TD items in their projects, they did not become aware of how much those items can impact those projects: "lack of knowledge of the impact of the TD item in question" and "technical debt was not identified as a problem at the time." The PAR *lack of organizational culture* refers to organizations that do not consider TD management as part of their development activities: "this is very dependent on the organization. In this case, it was not identified, monitored, or managed" and "because there is no permanent initiative to generate changes in the organizational culture." *Lack of resources* makes monitoring of TD items unfeasible, as we can notice in "even knowing the problem, there are no resources for immediate solution" and "the time and resources of the project were very limited." The PAR *lack of TD monitoring process* means that a process to monitor TD is missing in the organization, as evidenced in "there was no process for it" and "we weren't tracking it."

Lastly, the PARs *lack of specific team* and *react when becoming a problem* were considered in 3% and 2% of the projects, respectively. *Lack of specific team* refers to software projects that do not have a team responsible for monitoring TD: "it was known about for years, but we didn't have the headcount to refactor" and "not enough people or time too." *React when becoming a problem* indicates the existence of bad practices when managing with TD, as we can observe in "it is neglected until it becomes a problem" and "in the absence of planning, the methodology was reactionary to the problems."

As with monitoring practices, the data revealed distinct ***types*** of PARs. After analyzing the top ten PARs, we noticed that items like *lack of interest*, *focusing on short term goals*, and *react when becoming a problem* represent a **decision factor** considered by the team for not monitoring TD items. Differently, *lack of time* and *lack of knowledge on TD* represent situations in which the practitioners could have the intention of monitoring the debt. Still, they would not be able to due to issues out of their control (an **impediment**).

Table 5.7 presents the identified types of PARs, reporting the **types name**, the number of unique PARs for TD non-monitoring cited (**#PAR**), and the total number (i.e., count) of PARs (**#CPAR**) cited in each type. #CPAR also indicates the number of projects that used that PAR for justifying the non-monitoring of TD items in each type. Column **%PARP** corresponds to the percentage of #CPAR in relation to the total of all projects, indicating how frequently each type was used in software projects. Appendix Table C.6 presents, for each type, the PARs for TD non-monitoring ranked by number of citations. Impediments are the most common reason for explaining the non-monitoring of TD, being present in 64% of the software projects, while decision factors were considered in 50% of the projects.

Table 5.7 Types of PAR for TD non-monitoring

Type of PAR for TD non-monitoring	#PAR	#CPAR	%PARP
Decision factor	10	98	50%
Impediment	25	125	64%

Caption:
#PAR - Count of unique cited PAR for TD non-monitoring.
#CPAR - Count of PAR for TD non-monitoring.
%PARP - Percentage of CPAR in relation to the total of all projects (197).

Regarding the dimension ***nature***, considering only the top 10 cited PARs, the **managerial subset** is composed of *lack of interest*, *focusing on short term goals*, *lack of time*, *lack of organizational culture*, *lack of resources*, *lack of TD monitoring process*, *lack* *of a specific team*, and *react when become a problem*, representing ∼71% of all projects (column %PARP at Table 5.6). The **technical subset** is composed of *lack of knowledge* and *lack of understanding about the impact of the debt*, representing ∼18% of all projects (column %PARP at Table5.6). Appendix Table C.7 presents, for each nature, the PARs for TD non-prevention ranked by number of citations.

Table 5.8 presents the relation between types and natures of all PARs for TD non-monitoring. We have 27 managerial PARs, representing 78% of the total number of citations, and only eight technical PARs, corresponding to 22% of the total number of citations. This result clearly indicates that managerial issues are decisive in improving monitoring initiatives.

Table 5.8 Relation between type and nature of PARs for TD non-monitoring

Type of PARs for TD non-monitoring	**Nature of PAR for TD non-monitoring**	
	Technical	Managerial
Decision factor	3 (4%)*	7 (40%)
Impediment	5 (18%)	20 (38%)
Total	8 (22%)	27 (78%)

*The number in parentheses represents the percentage of the total number (i.e., count) of PARs cited in each nature in relation to the total of all cited PARs.

We also grouped the PARs into eight ***categories***:

- **Development issues**: encompasses PARs associated with software development activities. The two PARs grouped in this category are: *changing in the requirements* and *legacy system.*

- **External factors**: refers to PARs associated with factors that software teams cannot control. The three identified PARs in this category are *business pressure*, *project discontinued*, and *TD item payment do not generate revenue.*

- **Internal quality issues**: groups PARs associated with limitations compromising the softwares internal quality. The PARs *complexity of TD items*, *lack of effort to know the cause of TD*, and *too many TD items* compose this category.

- **Lack of knowledge**: includes only one PAR (*lack of knowledge on TD*) associated with the need for technical knowledge.

- **Methodology**: encompasses PARs related to processes followed by a software team. Examples of PARs in this category are *lack of TD monitoring process*, *react when becoming a problem*, and *TD item eliminated as soon as identified.*

- **Organizational**: includes PARs related to organizational decisions. Among them, we highlight *lack of interest*, *lack of organizational culture*, and *lack of resources.*

- **People**: refers to PARs associated with team characteristics, such as *emotional issues of the team*, *lack of experience*, and *team overload.*

- **Planning and management**: groups PARs related to management activities. Examples are *focusing on short term goals*, *lack of time*, and *ineffective planning and management.*

Table 5.9 presents the categories of PARs for TD non-monitoring, reporting the **categorys name**, the number of unique PARs cited (**#PAR**), and the total number (i.e., count) of PARs (**#CPAR**) cited in each category. #CPAR also indicates the number of projects considered a PAR for explaining the non-monitoring of TD items in each category. Lastly, the column **%PARP** corresponds to the percentage of #CPAR to the total of all projects. Appendix Table C.8 presents, for each category, the PARs for TD non-prevention ranked by number of citations.

Table 5.9 Categories of PARs for TD non-monitoring

Category of PARs for TD non-monitoring	#PAR	#CPAR	%PARP
Planning and management	10	90	46%
Organizational	6	70	36%
Lack of knowledge	1	23	12%
Methodology	6	20	10%
People	4	7	4%
External factors	3	5	3%
Development issues	2	4	2%
Internal quality issues	3	4	2%

Caption:
#PAR - Count of unique cited practice avoidance reasons (PARs) for TD non-monitoring.
#CPAR - Count of PARs for TD non-monitoring.
%PARP - Percentage of CPAR in relation to the total of all projects (197).

The categories *planning and management* and *organizational* have the greatest number of PARs, impacting 46% and 36% of the projects, respectively. One could assume that technical (development and internal quality) issues would have a decisive role when opting for not monitoring debt items. However, our results show the opposite. Managerial aspects are crucial to understanding why practitioners do not consider monitoring debt items in their projects.

5.1.2.1 PAR for TD Non-monitoring per Type of Debt. Figure 5.5 presents the relationship among the types of debt and the top 10 PARs, while Figure 5.6 shows the absolute values of these relationships, i.e., the quantity of times that each relationship was found. The PARs *lack of time* and *lack of knowledge on TD* are considered to explain the non-monitoring of most TD types. This shows that software teams can facilitate TD monitoring by improving their planning and spending time training on TD. We can also observe that the explanation of the non-monitoring of service, usability, and versioning debt items are only associated with the PARs *lack of knowledge on TD*, *focusing on short term goals*, and *lack of interest*, respectively. This reveals that besides promoting their knowledge on TD, software teams need to change their and their organizations mindset. Lastly, as test and documentation debt items share almost the same PARs, teams wanting to monitor these items can apply some strategies for facilitating it.

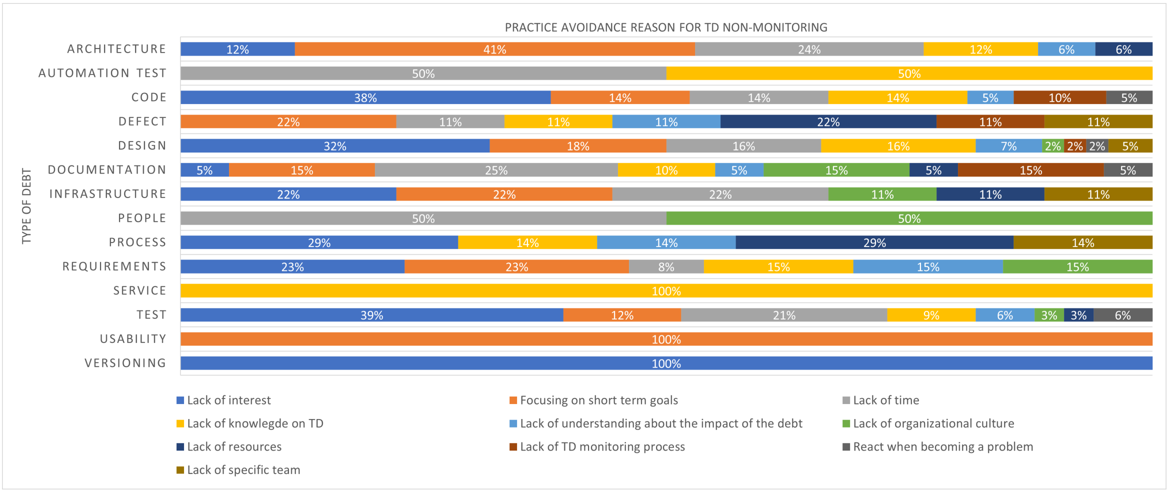


Figure 5.5 Relationship among types of debt and the top 10 PARs for TD non-monitoring - percentage values

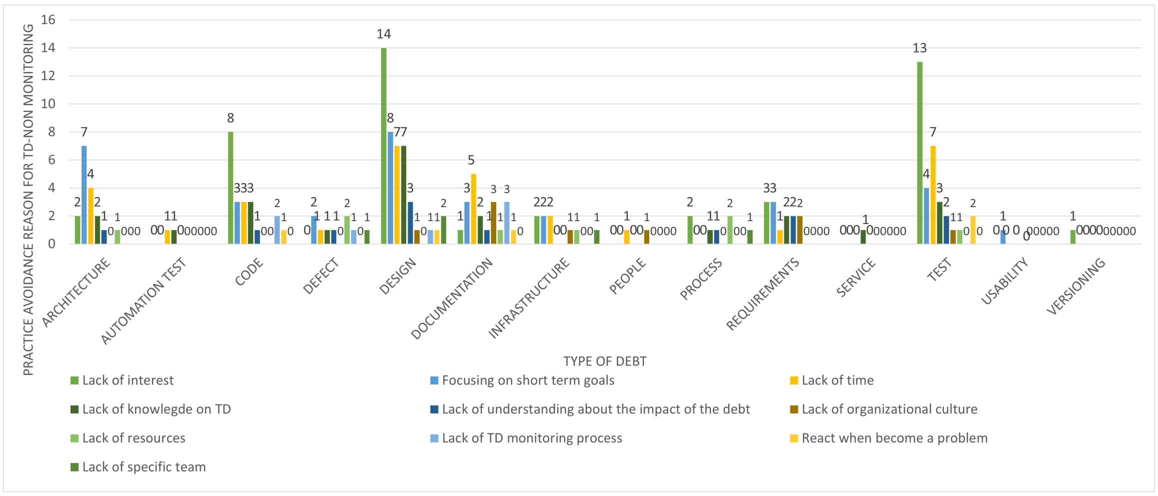


Figure 5.6 Relationship among types of debt and the top 10 PARs for TD non-monitoring - absolute values

Figure 5.7 shows the relationship between types of debt and types of PARs. We can see that except for usability and versioning, most types of debt are not monitored due to impediments, indicating that software practitioners need to change the mindset of their organizations and managers on TD monitoring. The number of times that each relationship occurred is shown in Figure 5.8.

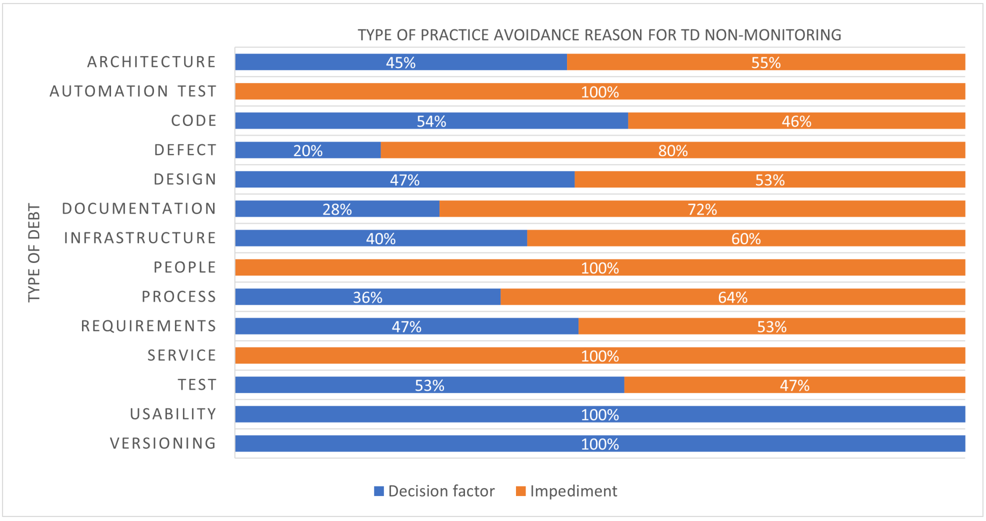


Figure 5.7 Relationship among types of debt and types of PARs for TD non-monitoring - percentage values

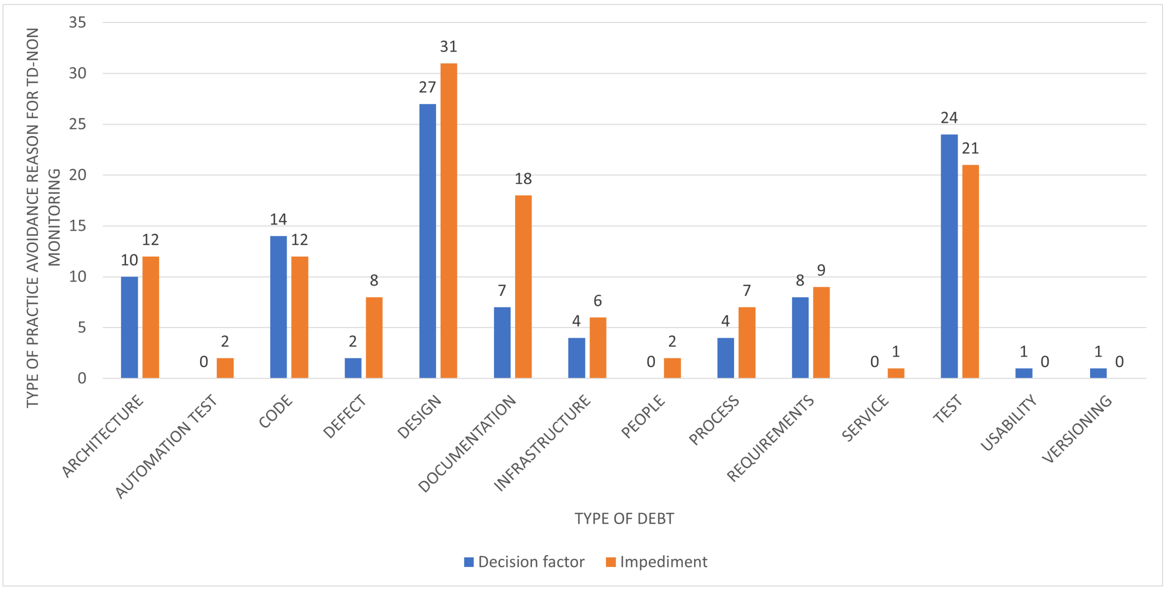


Figure 5.8 Relationship among types of debt and types of PARs for TD non-monitoring - absolute values

Figure 5.9 shows the relationship among the types of debt and categories of PARs, indicating how frequently PARs from these categories are used to explain the non-monitoring of each type of debt. The categories *planning and management* and *organizational* concentrate the PARs of almost all types of debt, indicating that software teams can improve their capacity for TD monitoring investing efforts to promote their managerial activities and organizational decisions. Although PARs from those categories are mainly used for explaining the non-monitoring of architecture, code, defect, design, documentation, infrastructure, people, process, requirements, test, usability, and versioning debt items, we noticed that the non-monitoring of automation test and service debt items are commonly justified by PARs from the category *lack of knowledge*. It means that software teams can improve their technical expertise to enable TD monitoring initiatives. The number of times that each relationship occurred is shown in Figure 5.10.

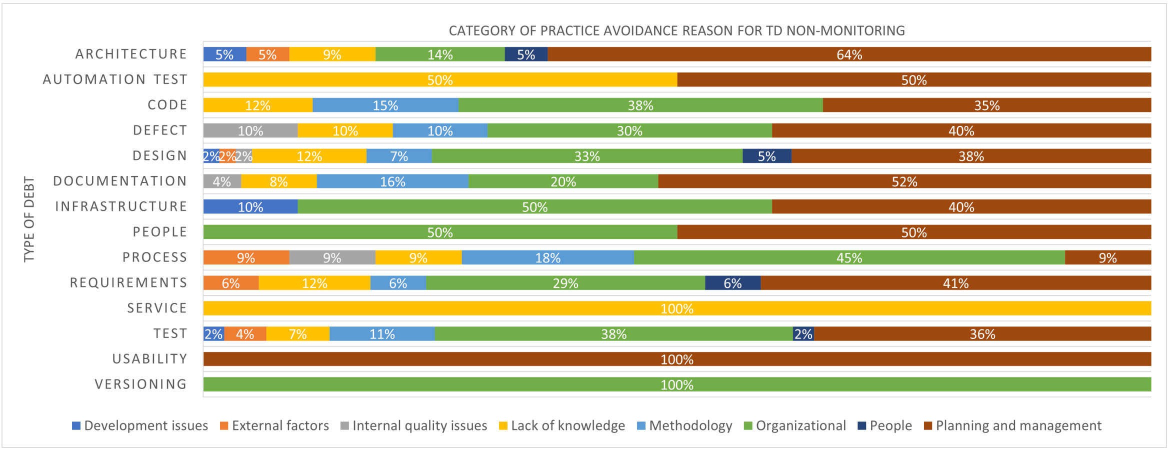


Figure 5.9 Relationship among types of debt and categories of PARs for TD non-monitoring - percentage values

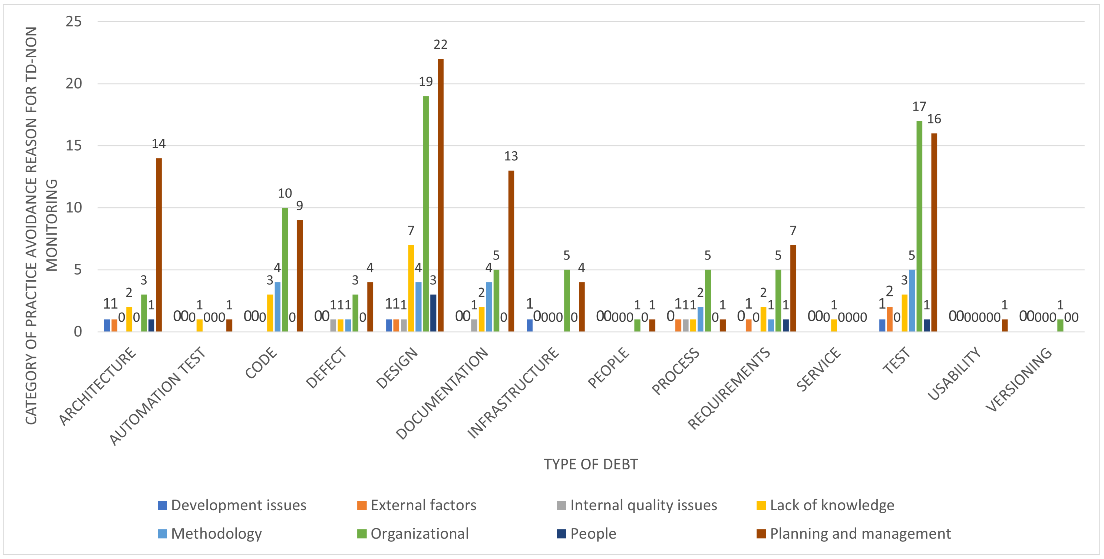


Figure 5.10 Relationship among types of debt and categories of PARs for TD non-monitoring - absolute values

5.2 DISCUSSION

In this section, we answer RQ2 and compare the findings with those reported in the related work.

5.2.1 Answering Research Question 2

We identified 46 monitoring-related practices for monitoring TD items (RQ2.1). Of these practices, *TD item backlog*, *use of tools*, *team meetings*, and *improving software development process* are the most used. Practitioners have used the identified practices to monitor, prevent, identify, and pay off TD items, and enable TD monitoring initiatives. It reveals that efforts to monitor TD items should consider the integration among these TD management activities. Also, we grouped the practices into seven categories. *Planning and management* and *methodology* encompass many of the practices, playing a leading role in TD monitoring. Lastly, the ten most cited practices are performed for monitoring the analyzed 14 types of debt. Practices from *planning and management* are used for monitoring almost all analyzed types, indicating that it would be a viable choice to start TD monitoring initiatives by them.

Concerning the PARs for explaining the non-monitoring of TD items (RQ2.2), we identified 35 of them. *Lack of interest*, *focusing on short term goals*, *lack of time*, and *lack of knowledge on TD* were the most commonly cited PARs. They are classified into two types: decision factors and impediments. In general, impediments are more commonly considered for justifying the non-monitoring of TD items than decision factors. Thus, monitoring TD is not just a matter of will but of mitigating the restrictions that curb the adoption of monitoring practices.

We grouped the PARs into eight categories. *Planning and management* and *organizational* concentrate the greatest number of PARs, revealing that decisions taken at organizational or managerial levels are decisive for monitoring TD items. Finally, we investigated the relation between PARs and types of debt. Overall, the ten most cited PARs have been considered to justify the non-monitoring of all types of debt. Particularly, PARs from *planning and management* and *organizational* have been supposed to explain most of the analyzed types.

5.2.2 Comparison to Related Work

This section presents a comparison between our findings with the ones reported in related work. As previously said, we did not find studies that investigated the PARs, thus, the comparison only takes into consideration the practices used for monitoring TD items. The comparison was performed on two levels: practices and categories of practices.

The practices for **enabling TD monitoring** were found both in our work and in at least some other past studies are *adoption of agile methodology*, *assign team for TD monitoring*, *continuous integration*, *improving the requirement management*, *TD management plan*, and *use of tools*. **TD monitoring actions** that appear both in our work and related work are *cost/benefit analysis*, *dashboard*, *identify the worst debt areas*, *measuring the effort*, *risk analysis*, *TD estimation*, *TD item backlog*, *TD status progress report*, *tracking TD items*, and *using informal practices*. The **TD prevention practices** reported in our study and past work are *code review* and *improving tests*. All **TD identification practices** reported in previous studies are found in our work (*identifying TD items* and *use of metrics for TD identification*). None of the **TD payment practices** in our work are in previous work. However, our study did not identify six practices recognized in related work, they are *setting a commonly agreed definition of done*, *using comments in the code or other artifacts*, *documenting issues in text or spreadsheets*, *making of dependencies and code problems*, *supply chain management*, and *monitor triggers*. Figure 5.11 shows the distinctions and overlaps between our work's TD monitoring-related practices and the related work. The complete comparison is available in Appendix Table C.9.

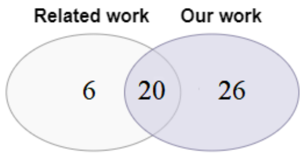


Figure 5.11 Number of unique TD monitoring-related practices found in our work and the related work

Regarding the distinctions, related work mainly found practices used to detail the definition of done, report issues using comments, spreadsheets, or text, monitor triggers, and apply supply chain management. On the other hand, we found more practices associated with other aspects of software development activities.

Table 5.10 presents the comparison considering the categories of practices. This comparison was only made with Li, Avgeriou and Liang (2015) because they reported categories of practices for monitoring TD items. The result of a comparison can be total (indicating that all practices from a category identified in our work are considered in the related work and vice-versa) or partial (indicating that our work has practices that are not considered in the related work). We notice that the categories *methodology*, *internal quality issues*, and *planning and management* were confirmed partially in the literature, while the categories *development issues*, *infrastructure*, *organizational*, and *people* were found only in our study.

Table 5.10 Comparison of related work on categories of TD monitoring-related practices

Our Categories	**Categories from Li, Avgeriou and Liang (2015)**		**Overlapping degree**
	Category Name	**Definition**	
Methodology	TD propagation tracking	Track the influences of TD through dependencies between other parts of a system and the parts of the system that contains TD.	Partial
	TD monitoring with quality attribute focus	Monitor the change of quality attributes that detrimental to TD, such as stability.	Partial
	Planned check	Regularly measure identified TD and track the change of the TD.	Partial
Internal quality issues	Threshold-based approach	Define thresholds for TD related quality metrics, and issue warnings if the thresholds are not met.	Partial
Planning and management issues	TD plot	Plot various aggregated measures of TD over time and look at the shape of the curve to observe the trends.	Partial
Development issues	-	-	-
Infrastructure	-	-	-
Organizational	-	-	-
People	-	-	-

In summary, our findings on TD monitoring-related practices and their categories complement and extend the set of information already reported in the technical literature.

5.3 CONCLUDING REMARKS

This capter reports software practitioners points of view on TD monitoring, revealing the practices used for monitoring TD and the PARs considered for explaining TD non-monitoring. We group practices and PARs into categories, indicating the crucial issues associated with TD monitoring. We also identify relationships between practices and types of debt and PARs and types of debt. These relations can support software practitioners in addressing specified practices or PARs to a type of debt commonly affecting their project. Threats to validity were previously discussed in Chapter 3.

The following chapter presents the state of technical debt payment practice from the analyses of the InsighTD data set.

Chapter

6

This chapter presents the state of the practice of TD payment obtained from analyses of the InsighTD data set.

TECHNICAL DEBT PAYMENT'S STATE OF PRACTICE

Technical debt (TD) payment refers to the activity of expending maintenance effort and resources to make up for the effects of previous technical compromises (RIOS; MENDONÇA; SPINOLA, 2018). Related work has sought to identify strategies, practices, and tools for TD payment (see Chapter 2). However, the findings are limited to the point of view of a few software practitioners.

Knowing the set of TD payment practices can help development teams choose the most appropriate practices (YLI-HUUMO; MAGLYAS; SMOLANDER, 2016), supporting the implementation of TD payment strategies or driving the development of new approaches for TD payment. In addition, having information about the Practice avoidance reasons (PARs) can help them understand the reasons that make it challenging to eliminate TD items, allowing the development of strategies to mitigate these impediments.

This chapter aims *to investigate, from the point of view of software development practitioners, which practices can be used to pay off TD items and what PARs would justify the non-payment of these items.* To achieve this goal, we define **RQ3: How have software development teams paid off technical debt items on their projects?** And its sub-questions: RQ3.1: What are the practices used by software practitioners to deal with technical debt items in their projects? And RQ3.2: What are the practice avoidance reasons (PARs) considered by software practitioners for not paying off TD? To answer these RQs, we use and analyze the InsighTD data set (as described in Chapter 3). This chapter is based on the following papers: (i) Surveying Software Practitioners on Technical Debt Payment Practices and Reasons for not Paying off Debt Items (FREIRE *et al.*, 2020b), (ii) How do Technical Debt Payment Practices Relate to the Effects of the Presence of Debt Items in Software Projects? (FREIRE *et al.*, 2021a) and (iii) Software Practitioners Point of View on Technical Debt Payment (FREIRE *et al.*, 2023).

The main contributions of this Chapter are:

- An up-to-date, comprehensive list of practices related to TD payment ranked by the number of their citations by software practitioners.
- An up-to-date, comprehensive list of PARs ranked by the number of their citations by software practitioners.
- A categorization of TD payment and PARs concerning their relationship to software planning, development, management, and methodology issues.
- A definition of types of practices and types of PARs about these contributions in TD payment initiatives.
- A definition of the nature of practices and nature of PARs concerning technical and managerial activities.
- The relationships between types of debt and practices, and types of debt and PARs.
- A comparison of practices and PARs found in this study to those reported in the technical literature.

Next, Section 6.1 presents the perception of software practitioners on TD payment. Section 6.2 discusses the results and compares them with those reported by related work. Lastly, Section 6.3 offers the concluding remarks of this Chapter.

6.1 PERCEPTION OF INSIGHTD'S PRACTITIONERS ON TD PAYMENT

This section presents our findings from the InsighTD data set analyses concerning TD payment (answers given to Q26 and Q27). We organize these findings per RQ.

6.1.1 RQ3.1: What are the practices used by software practitioners to deal with technical debt items in their projects?

In total, 259 (∼40%) participants indicated in Q26 that the TD item mentioned was paid off, and 218 (∼84%) of them described how TD was paid off in Q27. Thus, we used this set of 218 responses for answering RQ3.1.

We identified 32 payment-related practices for the payment of TD items. Table 6.1 summarizes the ten most cited ones. Appendix Table D.1 presents the identified payment-related practices and quotes from participants. Table 6.1 reports the **payment-related practice name** and the total number (i.e., count) of the 218 responses that mentioned that practice for paying off TD items (**#CPRP**). The column **%PRPP** presents the percentage of #CPRP in relation to the total of all responses (i.e., 218), revealing how frequently each practice was used in software projects. The most cited payment-related practice *code refactoring* impacts ∼37% of the projects. This result was expected, as *code refactoring* has been widely used for paying TD items off (LI; AVGERIOU; LIANG, 2015; BEHUTIYE *et al.*, 2017).

Table 6.1 Top 10 cited TD payment-related practices

NO	TD Payment-related practice	#CPRP	%PRPP
1st	Code refactoring	81	37%
2nd	Investing effort on TD payment activities	33	15%
3rd	Design refactoring	25	12%
4th	Investing effort on testing activities	22	10%
5th	Prioritizing TD items	15	7%
6th	Negotiating deadline extension	14	6%
7th	Adjusting code to follow good programming practices	10	5%
8th	Monitoring and controlling project activities	10	5%
9th	Improving the development process	9	4%
10th	Increasing the project budget	9	4%

Caption:
#CPRP - Count of payment-related practices.
%PRPP - Percentage of #CPRP in relation to the total of all projects (218).

In the context of this study, *code refactoring* practice means the changes made to the system code without changing its external behavior, as described in the quotes extracted from the responses of the participants: "we rewrote the offending code" and "refactoring the class and its dependents." The *investing effort in TD payment activities* practice indicates the effort expended by the project team to eliminate TD items, as described below: "taking sprints to pay the debt" and "internal developments were generated to improve these problems." The *design refactoring* practice concerns changes made to the design of the system, for example, "refactoring and changing the architecture" and "the identified technical debt was resolved through the design update and refactorings." The *investing effort in testing activities* practice indicates the project team's effort to carry out software testing activities, as seen in "additional development hours and code testing" and "adoption of automated unit tests and programming in pair..."

The practices *prioritizing TD items* and *negotiating deadline extension* were used, respectively, in 7% and 6% of the projects, while *adjusting code to follow good programming practices* and *monitoring and controlling project activities* were used in 5% of the projects. The *prioritizing TD items* practice indicates that the project team intends to prioritize the debt items to be paid, as can be seen in the following response from the participants: "due to the project deadline, as soon as the TD is detected, we already plan its development for the next sprints of the project." The *negotiating deadline extension* practice indicates that the project team has negotiated a deadline with the customer to be able to eliminate TD items, as seen in "it was possible to negotiate time with the user to make improvements." The *adjusting code to follow good programming practices* practice is related to applying good programming practices, for example, "benefits of shared library cutting down on dev. time duplicating or maintaining extra code." The *monitoring and controlling project activities* practice indicates that the project team used mechanisms to monitor the project, as can be seen in "With the measurement bulletins, there was the metric of progress."

Finally, the practices *improving the development process* and *increasing the project budget* were used in 4% of the projects. The *improving the development process* practice is related to the adequacy of the development process to decrease the amount of TD items in the project, for example, "improvement in the development and changes process." The *increasing the project budget* practice relates to negotiating with the client to increase the project budget to enable the payment of TD items, as indicated in the following response from the participants: "with cost overruns in money and time, updating the solution to the latest changes in technology."

Looking at the top 10 cited payment-related practices, presented in Table 6.1, only the practices *code refactoring*, *design refactoring*, and *adjusting code to follow good programming practices* allow the direct payment of TD items. The other most cited practices only support TD payment initiatives. For example, the practice *investing effort on TD payment activities* can create a favorable setting for TD payment. Still, TD items are only sometimes paid off when this practice is applied. Thus, we identify one dimension of TD payment-related practices, ***type***, having the following values:

- **Payment action**: includes practices directly resulting in TD item removals, such as *code refactoring*, *design refactoring*, and *update system documentation.*
- **Enabling TD payment**: includes practices that improve the capacity of development teams to pay off debt items. Some examples are *investing effort on TD payment activities*, *negotiating deadline extension*, and *increasing the project budget.*
- **TD prevention**: refers to practices intended to curb potential TD from being incurred. Among them, we have *monitoring and controlling project activities*, *investing effort on testing activities*, and *using short feedback iterations.*
- **TD prioritization**: is related to practices that support ranking TD items according to classification criteria. Only *prioritization of TD items* composes this category.

Indeed, the ***type*** dimension shows the inherent and unavoidable connection between TD management activities. Table 6.2 shows the number of unique TD payment practices (**#PRP**) in each type, and the total number of responses citing a practice (**#CPRP**) in each type. Column **%PRPP** corresponds to the percentage of all projects that cited a payment practice of that type, revealing how prevalent each type is in software projects. Appendix Table D.2 presents, for each type, the payment-related practices ranked by number of citations.

The most cited type, *payment action*, was applied in 67% of the projects but consisted of only eight unique actions, ranging from *code refactoring* to *update system documentation.* Turning attention to another dimension, the ***nature*** of the payment practices, two of them (*restarting the project from scratch* and *system retirement*) are related to **managerial** decisions, while the others are **technical** practices. Most of the technical practices can be applied during software implementation, but some, such as *bug fixing* and *code refactoring*, are applicable during maintenance to improve the external or internal quality of the system.

Table 6.2 Types of TD payment-related practices

Type of payment-related practice	#PRP	#CPRP	%PRPP
Payment action	8	146	67%
Enabling TD payment	11	93	43%
TD prevention	12	64	30%
TD prioritization	1	15	7%

Caption:
#PRP - Count of unique cited payment-related practices.
#CPRP - Count of payment-related practices.
%PRPP - Percentage of CPRP in relation to the total of all projects (218).

Among the top 10 cited TD payment-related practices, the subset that is technical in nature is composed of *code refactoring*, *design refactoring*, and *adjusting code to follow good programming practices*, representing 53% of all projects (column %PRPP at Table 6.1). The managerial subset is composed of *investing effort on TD payment activities*, *investing effort on testing activities*, *prioritizing TD items*, *negotiating deadline extension*, *monitoring and controlling project activities*, *improving the development process*, and *increasing the project budget*, representing 51% of all projects (column %PRPP at Table 6.1). In other words, technical and managerial practices are chosen almost equally in TD payment decisions, although there is a wider variety of managerial payment practices. Appendix Table D.3 presents the TD payment-related practices of each nature ranked by number of citations.

Table 6.3 presents the relation between types and natures of all TD payment-related practices. Most of TD payment and prevention practices are technical, while most practices for enabling TD payment are managerial.

Table 6.3 Relation between type and nature of payment-related practices

Type of payment-related practice	Nature of payment-related practice	
	Technical	Managerial
Payment action	6 (44%)*	2 (2%)
TD prevention	8 (7%)	4 (13%)
Enabling TD payment	1 (1%)	10 (28%)
TD prioritization	0 (0%)	1 (5%)
Total	15 (52%)	17 (48%)

*The number in parentheses represents the percentage of the total number (i.e., count) of payment -related practices cited in each nature in relation to the total of all cited practices (318).

Next, we organized the set of practices related to the payment of TD items into the following ***categories***:

- **Development issues**: encompasses six practices that are applied during the implementation of software, such as *update system documentation*, *adjusting code to follow good programming practices*, and *solving technical issues.*

- **External quality issues**: groups practices related to software quality aspects that users can perceive. Only *bug fixing practice* is part of this category.

- **Infrastructure**: groups two practices related to tools, technologies, and development environment, *external tools* and *organizing the project repository.*
- **Internal quality issues**: includes two practices that can be employed to address limitations that compromise the internal quality of the software, *code refactoring*, and *design refactoring.*
- **Methodology**: encompasses 12 practices associated with processes followed by a software team. Among them, we highlight *investing effort on TD payment activities*, *investing effort on testing activities*, and *using short feedback iterations.*
- **Organizational**: refers to two practices associated with organizational decisions, *hiring specialized professionals* and *changing the project management.*
- **People**: includes two practices directly related to the members of software development teams, *improving the team collaboration* and *communicating with the customer about TD items.*
- **Planning and management**: groups five practices associated with management activities. Examples of practices in this category are *monitoring and controlling project activities*, *prioritizing TD items*, and *negotiating deadline extension.*

Table 6.4 presents the categories of payment-related practices, reporting the **categorys name**, the number of unique payment-related practices cited (**#PRP**), and the total number (i.e., count) of responses (or projects) that cite practices in each category (**#CPRP**). The column **%PRPP** corresponds to the percentage of #CPRP in relation to the total of all projects. Appendix Table D.4 presents, for each category, the TD payment-related practices ranked by number of citations.

Table 6.4 Categories of payment-related practices

Category of payment-related practices	**#PRP**	**#CPRP**	**%PRPP**
Internal quality issues	2	106	49%
Methodology	12	96	44%
Planning and management	5	50	23%
Development issues	6	39	18%
Organizational	2	10	5%
People	2	7	3%
External quality issues	1	6	3%
Infrastructure	2	4	2%

Caption:
#PRP - Count of unique cited payment-related practices.
#CPRP - Count of payment-related practices.
%PRPP - Percentage of CPRP in relation to the total of all projects (218).

We can see that practices in the *internal quality issues* category were cited in almost half of the projects, highlighting the central role of internal quality in the payment of TD items. This is consistent with the view that TD primarily affects internal quality issues in software products (AVGERIOU *et al.*, 2016). The categories *methodology*, *planning and management*, and *development issues* were also commonly mentioned, appearing in 44%, 23%, and 18% of the responses, respectively (see column %PRPP at Table 6.4).

Table 6.4 also shows that only some participants use improvement of the organizational environment (*organizational* category) or technical knowledge of the team (*people* category) as part of their TD payment initiatives. Finally, practices from the categories *external quality issues* and *infrastructure* were very seldom mentioned.

6.1.1.1 TD Payment Practice per Type of Debt. Figure 6.1, Table 6.5, and Figure 6.3 all present the relationships among TD payment practices and types of TD. Figure 6.1 focuses on the top 10 TD payment-related practices, while Table 6.5 and Figure 6.3 break down the relationship with TD types according to the types and categories, respectively, of TD payment practices. Figures 6.2 and 6.4 show the absolute values of these relationships, i.e., the quantity of times that each relationship was found.

We notice in Figure 6.1 that all practices composing the top 10 were applied to design debt items, and most were also applied to code and test debt items. *Code refactoring* was cited as a TD payment practice for every type of TD, and *investing effort on TD payment activities* (the second most mentioned) was cited for nearly all types except defect debt. The number of times that each relationship occurred is shown in Figure 6.2.

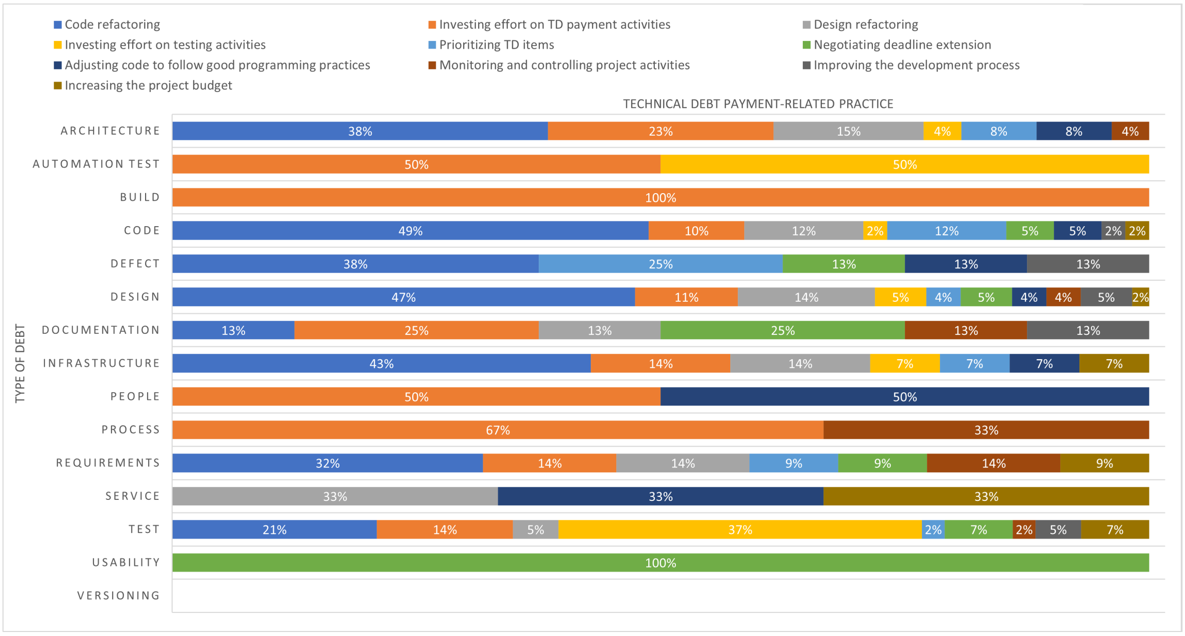


Figure 6.1 Relationship among types of debt and the top 10 TD payment-related practices - percentage values

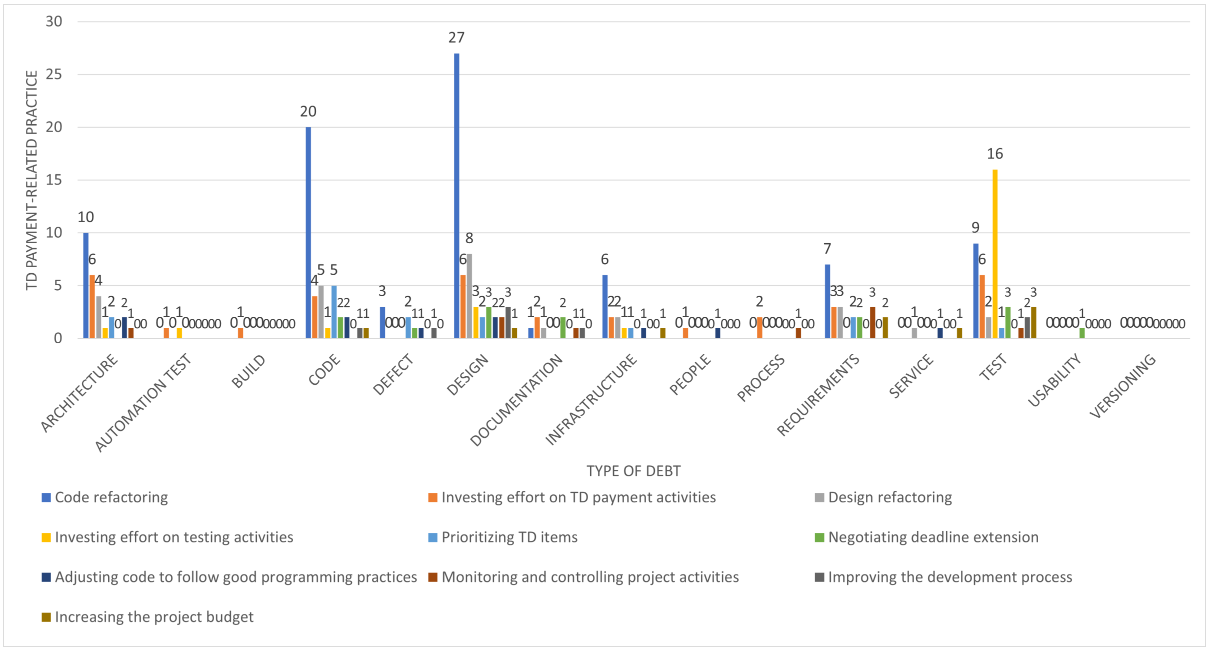


Figure 6.2 Relationship among types of debt and the top 10 TD payment-related practices - absolute values

From Table 6.5, we can observe that not all types of debt were directly paid off (e.g., automation test, process, usability, and versioning debt). At the same time, payment actions (marked in gray) were commonly cited for paying off design, code, and architecture debt items. Perhaps software practitioners focused on these items because they impact technical activities and thus can impact the softwares internal and external quality.

Table 6.5 Relationship among the types of technical debt payment-related practices and types of debt

Type of debt	Type of TD payment-related practice			
	Payment action	Enabling TD payment	TD prevention	TD prioritization
Architecture debt	21	8	5	2
Automation test debt	0	1	1	0
Build debt	1	1	0	0
Code debt	29	9	12	5
Defect debt	4	2	0	2
Design debt	41	22	18	2
Documentation debt	8	7	1	0
Infrastructure debt	12	3	2	1
People debt	2	3	0	0
Process debt	0	5	2	0
Requirements debt	13	12	5	2
Service debt	4	1	0	0
Test debt	14	19	17	1
Usability debt	0	1	0	0
Versioning debt	0	0	1	0

Practitioners have also taken steps to enable TD payment (marked in orange), as well as to prevent TD (marked in blue), for types of TD closely related to the code (i.e., design, test, code, architecture, and requirements). One might conclude that software practitioners have spent effort improving the artifacts produced in technical, code-related activities as these artifacts are strongly related to or can directly impact the code. Therefore, improving those activities facilitates the application of practices for paying off TD items and potentially prevents TD items from being incurred.

Figure 6.3 presents the relationship between the categories of TD payment-related practices and types of debt. We notice that nearly all categories of TD payment practices have been applied to code, design, and requirements debt. In general, the categories *development*, *internal quality*, *methodology*, and *planning and management issues* are widespread for most types of TD, indicating that practitioners can invest in these practices and leverage them for multiple types of debt. The number of times that each relationship occurred is shown in Figure 6.4.

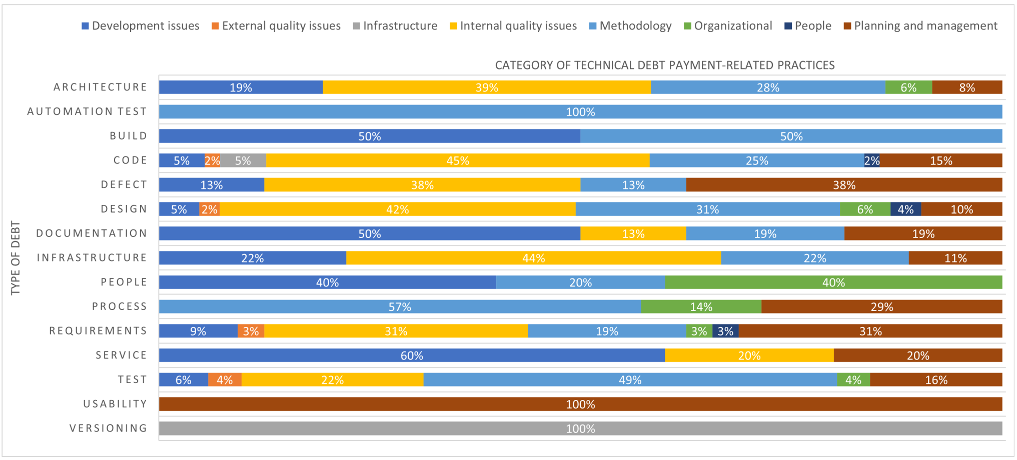


Figure 6.3 Relationship among types of debt and categories of TD payment-related practices - percentage values

6.1.2 RQ3.2 What are the practice avoidance reasons (PARs) considered by software practitioners for not paying off TD?

In total, 394 (~60%) of the participants indicated that the TD item mentioned was not paid off and 261 (~66%) of those explained why. This set of 261 responses was used for answering RQ3.2.

Table 6.6 shows the ten most cited PARs for TD non-payment (out of a total of 27) identified in Q27. All the identified PARs and quotes from participants are available in Appendix Table D.5. Table 6.6 reports the **PAR name** and the total number (i.e., count) of responses citing each PAR (**#CPAR**). Column **%PARP** shows the percentage of #CPAR in relation to the total of all projects. *Focusing on short-term goals* was the most cited PAR. It impacts 26% of the projects. One can expect that because software teams can deliver faster by focusing on short-term goals, which is a primary cause of

TD (SPINOLA *et al.*, 2013; RIOS *et al.*, 2020). These results show that it is also the most common cause of perpetuating TD by not paying it off. The *lack of organizational interest*, *lack of time*, and *cost* impacted 18%, 16%, and 13% of the projects, respectively.

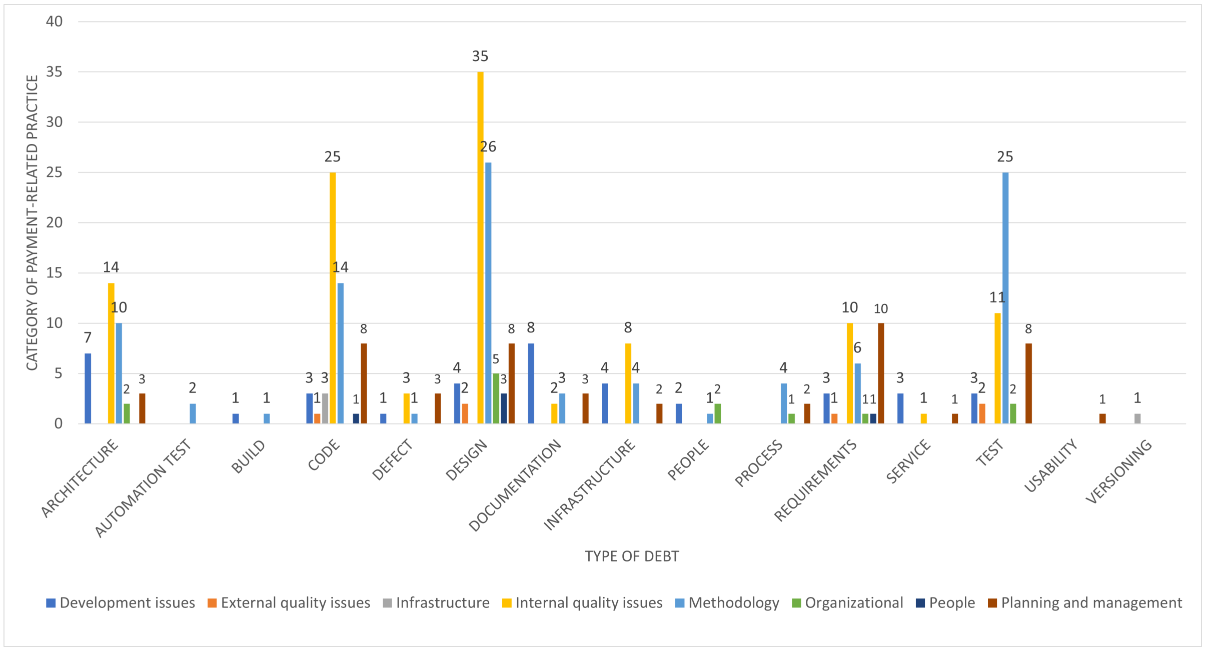


Figure 6.4 Relationship among types of debt and categories of TD payment-related practices - absolute values

Table 6.6 Top 10 practice avoidance reasons (PARs) for TD non-payment

NO	Practice avoidance reason (PAR)	#CPAR	%PARP
1st	Focusing on short term goals	69	26%
2nd	Lack of organizational interest	48	18%
3rd	Lack of time	41	16%
4th	Cost	34	13%
5th	Lack of resources	19	7%
6th	Customer decision	13	5%
7th	Complexity of the TD item	12	5%
8th	Effort	11	4%
9th	Complexity of the project	10	4%
10th	Insufficient management view about TD payment	10	4%

Caption:
#CPAR - Count of practice avoidance reason for TD non-payment.
%PARP - Percentage of CPAR in relation to the total of all projects (261)

In the context of this study, *focusing on short term goals* means that the project team did not prioritize the payment of the TD items, postponing it to a future moment, as indicated in the following responses from the participants: "focus on next release" and "other backlog items were considered higher priority." The *lack of organizational interest* PAR indicates that the payment of TD items is not encouraged by the organization, as can be seen in: "we learned enough to move forward, but no one wanted to spend the time" and "this is how the company works." The *lack of time* PAR means that there was no time available on the project to pay the TD items, as can be seen in the answers: "no time dedicated to significant redesign and rework" and "due to tight deadlines." The *cost* PAR is associated with the financial cost required to eliminate the TD item from the project, for example, "it is too expensive to fix. Too big to succeed."

The *lack of resources* PAR impacted 7% of the projects, while the *customer decision* and *complexity of the TD item* each impacted 5% of the projects. The *lack of resources* indicates that the project team did not have enough resources to make payment for the TD items possible, as observed in: "we did not have the technical resources, or the time required to carry out the activity." The *customer decision* PAR means that the TD non-payment was a decision made by the project customer, according to the following answers: "because of the clients decision" and "the customer did not want the technical debt to be resolved, as it involved more costs." The *complexity of the TD item* PAR indicates that the TD item present in the project is very complex, making its elimination unfeasible, for example, "it is something that cannot be eliminated, the errors generated must be supported."

Lastly, the PARs *effort*, *complexity of the project*, and *insufficient management vision about TD payment* impacted 4% of the projects. The *effort* PAR concerns the amount of work required to eliminate the TD item, as can be seen in: "would take a lot of time to migrate to a new back-end database and change the application code." The *complexity of the project* PAR indicates that the payment of the TD item was unfeasible due to the complexity of the entire project, as evidenced in the following answer: "It is a very large project and the interconnectedness of the data layer throughout the project makes completely rewriting it very difficult." The *insufficient management view about TD payment* PAR is related to project manager resistance and lack of vision on the importance of TD payment, for example, "resistance from the project manager."

We organized the PARs for TD non-payment into the following ***types***:

- **Decision factor**: refers to PARs for deciding not to pay off the TD. Among them, we have *focusing on short-term goals*, *lack of organizational interest*, and *insufficient management view about TD payment.*

- **Impediment**: points to situations in which the development team wanted to pay off the TD, but they could not pay it off for some reason. Examples are *lack of time*, *cost*, and *lack of resources.*

Table 6.7 presents the types of PARs, reporting the **types name**, the number of unique PARs cited (**#PAR**), and the total number of projects citing PARs (**#CPAR**) in each type. The column **%PARP** corresponds to the percentage of #CPAR in relation to the total of projects. Appendix Table D.6 presents, for each type, the PARs for TD non-payment ranked by number of citations.

Table 6.7 Types of PAR for TD non-payment

Type of PAR for TD non-payment	#PAR	#CPAR	%PARP
Decision factor	12	149	57%
Impediment	15	170	65%

Caption:
#PAR - Count of unique cited PAR for TD non-payment.
#CPAR - Count of PAR for TD non-payment.
%PARP - Percentage of CPAR in relation to the total of all projects (261).

We can see that decision factors are common, but impediments often prevent payment of TD items. About 8% of the participants indicated a combination of an impediment and a team decision factor. This suggests that team decisions can sometimes align with other factors influencing TD non-payment in software projects.

As with the payment-related practices, the PARs for TD non-payment can also be related to managerial or technical activities, as in *focusing on short term goals* and *complexity of the TD item*, respectively. Thus, the PARs for TD non-payment can be categorized according to their ***nature***:

- **Technical PARs**: are related to PARs for TD non-payment involved in the technical activities of a software project, such as requirement analysis, design, coding, and testing.
- **Managerial PARs**: are related to PARs for TD non-payment involved in management activities of the development of a software project, such as hiring, project management, and cost estimation.

Considering only the top 10 cited PARs, the **managerial subset** is composed of *focusing on short term goals*, *lack of organizational interest*, *lack of time*, *cost*, *lack of resources*, *customer decision*, *effort*, *complexity of the project*, and *insufficient management view about TD payment*, representing 98% of all projects (column %PARP at Table 6.6). The technical subset is composed of the complexity of the TD item PAR, representing only 5% of all projects (column %PARP at Table 6.6). Appendix Table D.7 presents, for each nature, the PARs for TD non-monitoring ranked by number of citations.

Table 6.8 presents the relation between types and natures of all PARs for TD non-payment. We have 21 managerial PARs, representing 93% of the total number of citations, and only six technical PARs, corresponding to 7% of the total number of citations. This result clearly suggests that dealing with managerial issues is quite decisive if we are interested in improving the use of TD payment practices.

Table 6.8 Relation between type and nature of PARs for TD non-payment

Type of PARs for TD non-payment	Nature of PAR for TD non-payment	
	Technical	Managerial
Decision factor	2 (1%)*	10 (46%)
Impediment	4 (6%)	11 (47%)
Total	6 (7%)	21 (93%)

*The number in parentheses represents the percentage of the total number (i.e., count) of PARs cited in each nature in relation to the total of all cited PARs.

We also classified the identified PARs for TD non-payment into eight ***categories***:

- **Development issues**: groups two PARs related to software development activities, *complexity of the project*, and *decision to not change the framework.*

- **External factors**: organizes four PARs related to factors that are out of the control of the development team, such as *customer decision*, *the project was discontinued*, and *TD items do not affect the user.*

- **Internal quality issues**: encompasses two PARs related to system code and structure characteristics. Those PARs are *complexity of the TD item* and *number of TD items.*

- **Lack of knowledge**: groups two PARs associated with the need for technical knowledge. We have *lack of technical knowledge* and *lack of knowledge about TD.*

- **Methodology**: includes five PARs associated with process activities. Among them, we highlight *lack of adoption of lessons learned*, *lack of testing*, and *non-application of mitigation actions on TD causes.*

- **Organizational**: encompasses three PARs associated with organizational decisions. Those PARs are *lack of organizational interest*, *lack of resources*, and *high team turnover.*

- **People**: includes three PARs related to team characteristics, *insufficient management view about TD payment*, *team overload*, and *lack of committed team.*

- **Planning and management**: organizes six PARs related to management activities, such as *focusing on short term goals*, *lack of time*, and *cost.*

Table 6.9 presents the categories of PARs, reporting the **categorys name**, the number of unique PARs cited (**#PAR**), and the total number (i.e., count) of projects citing PARs (**#CPAR**) in each category. The column **%PARP** corresponds to the percentage of #CPAR in relation to the total of all projects. Appendix Table D.8 presents, for each category, the PARs for TD non-payment ranked by number of citations.

The most cited category, *planning and management*, affects 63% of software projects. The *organizational* category was also commonly found, affecting 26% of the projects. Although one may expect that development issues are crucial for determining the non-payment of TD items, planning and management, and organizational issues seem to be more decisive.

Table 6.9 Categories of PARs for TD non-payment

Category of PARs for TD non-payment	**#PAR**	**#CPAR**	**%PARP**
Planning and management	6	164	63%
Organizational	3	69	26%
External factors	4	22	8%
People	3	20	8%
Internal quality issues	2	15	6%
Development issues	2	11	4%
Lack of knowledge	2	9	3%
Methodology	5	9	3%

Caption:
#PAR - Count of unique cited practice avoidance reasons (PARs) for TD non-payment.
#CPAR - Count of PARs for TD non-payment.
%PARP - Percentage of CPAR in relation to the total of all projects (261).

6.1.2.1 PAR for TD Non-payment per Type of Debt. Figures 6.5, 6.7, and 6.8 all present the relationships among PARs for non-payment of TD and types of TD. Figure 6.5 focuses on the top 10 PARs, while Figures 6.7 and 6.8 break down the relationship with TD types according to the types and categories, respectively, of PARs. Figures 6.6 and 6.9 show the absolute values of these relationships, i.e., the quantity of times that each relationship was found.

We notice in Figure 6.5 that all the PARs composing the top 10 were used for justifying the non-payment of architecture, design, and test debt items. For documentation and requirements items, all PARs except *complexity of the project* and *complexity of the TD item* were considered. We can also observe that *focusing on short term goals*, *lack of organizational interest*, and *lack of time* are commonly related to almost all types of debt, indicating that dealing with them could positively impact on initiatives to pay debt items off.

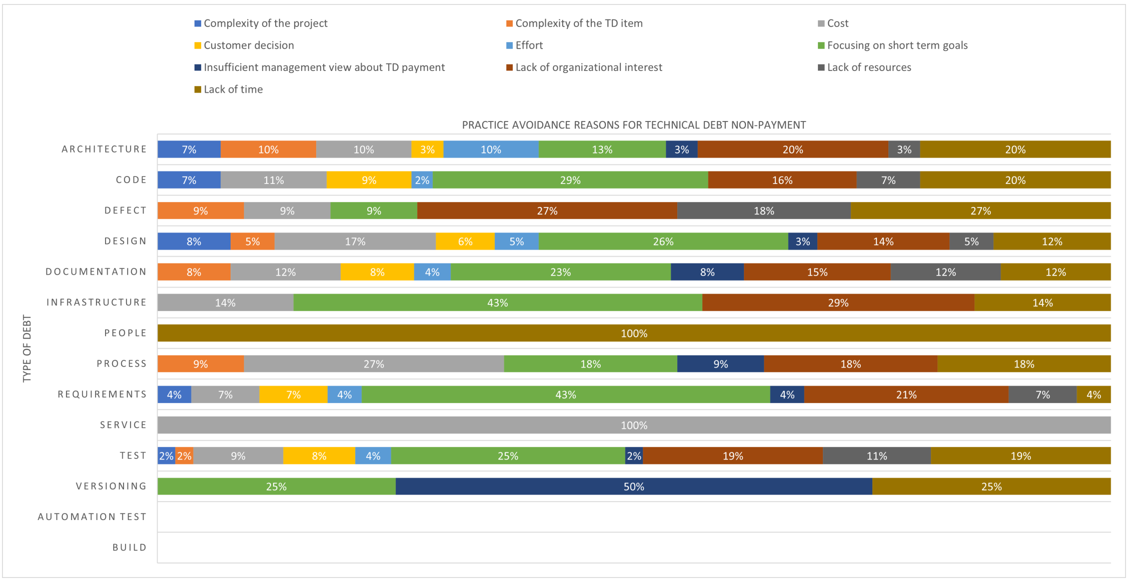


Figure 6.5 Relationship among types of debt and the top 10 PARs for TD non-payment - percentage values

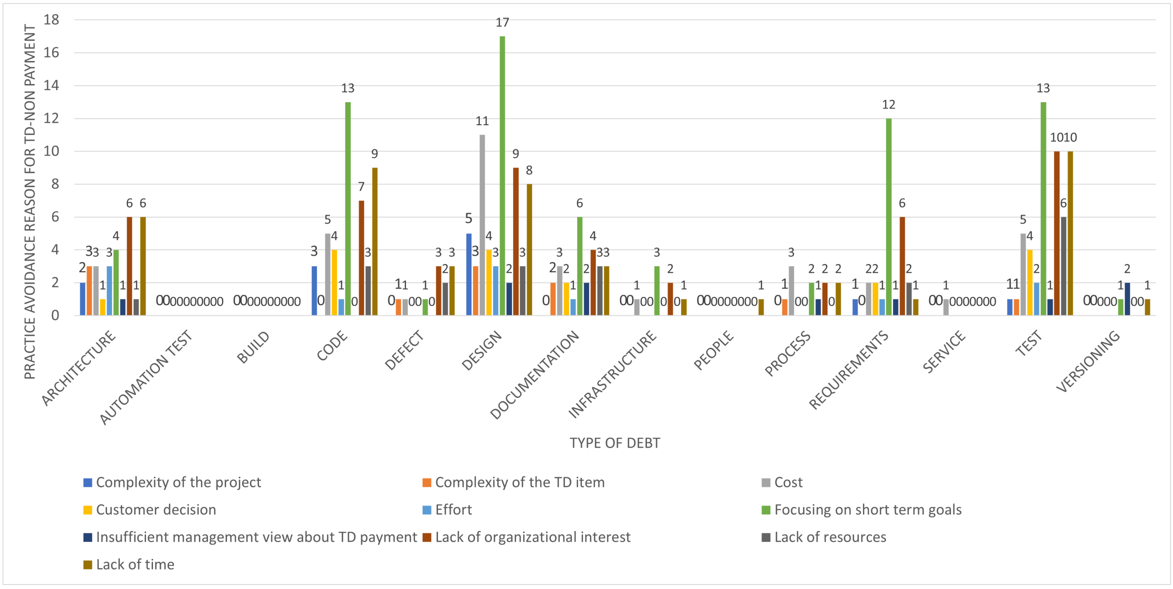


Figure 6.6 Relationship among types of debt and the top 10 PARs for TD non-payment - absolute values

In Figure 6.7, we can observe that decision factors are responsible for failing to pay almost all debt types, except build, people, service, and usability debt. Impediments are cited as PARs, particularly for the non-payment of design, test, and code debt items.

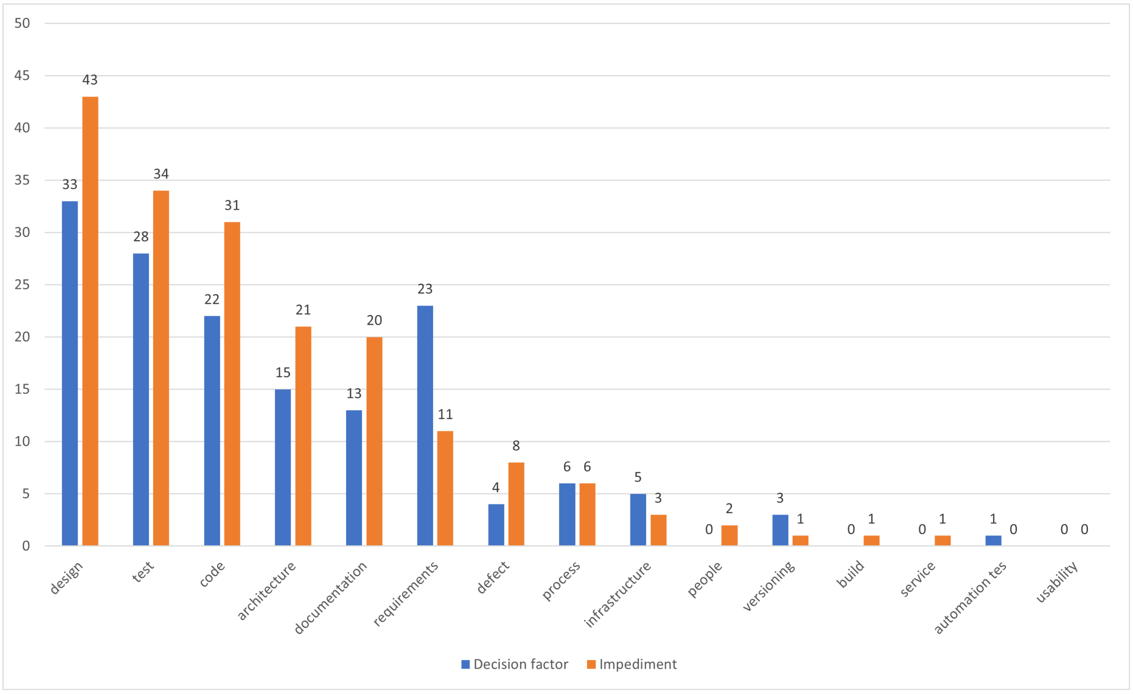


Figure 6.7 Relationship among types of debt and types of PARs for TD non-payment

Figure 6.8 presents the relationship among categories of PARs for not paying off TD and types of debt We notice that design and test debt are related to all categories of PARs for TD non-payment. Architecture, documentation, and requirements debt are associated with seven of eight categories. The *planning and management* category is related to all types of debt and the categories *organizational* and *people* are also very common.

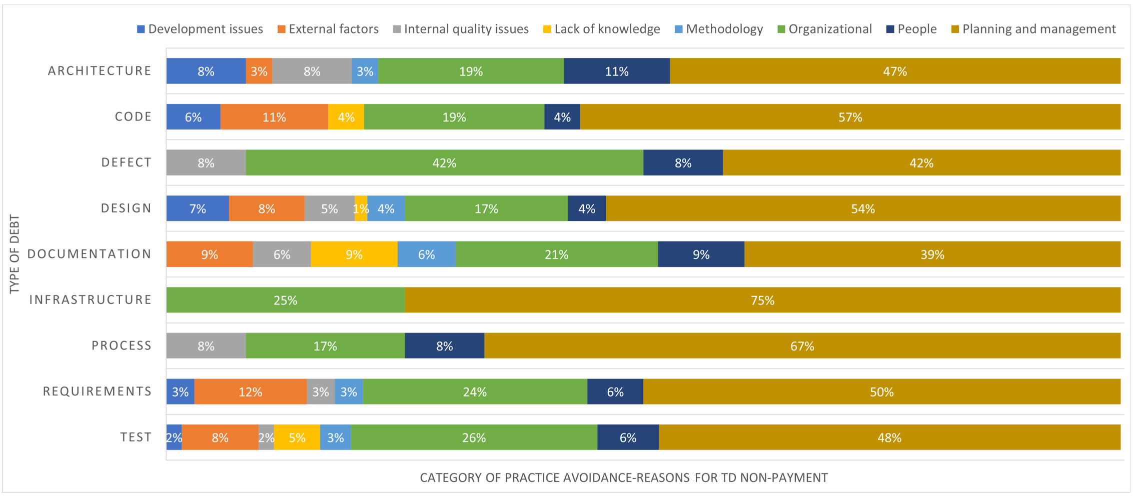


Figure 6.8 Relationship among types of debt and categories of PARs for TD non-payment - Percentage values

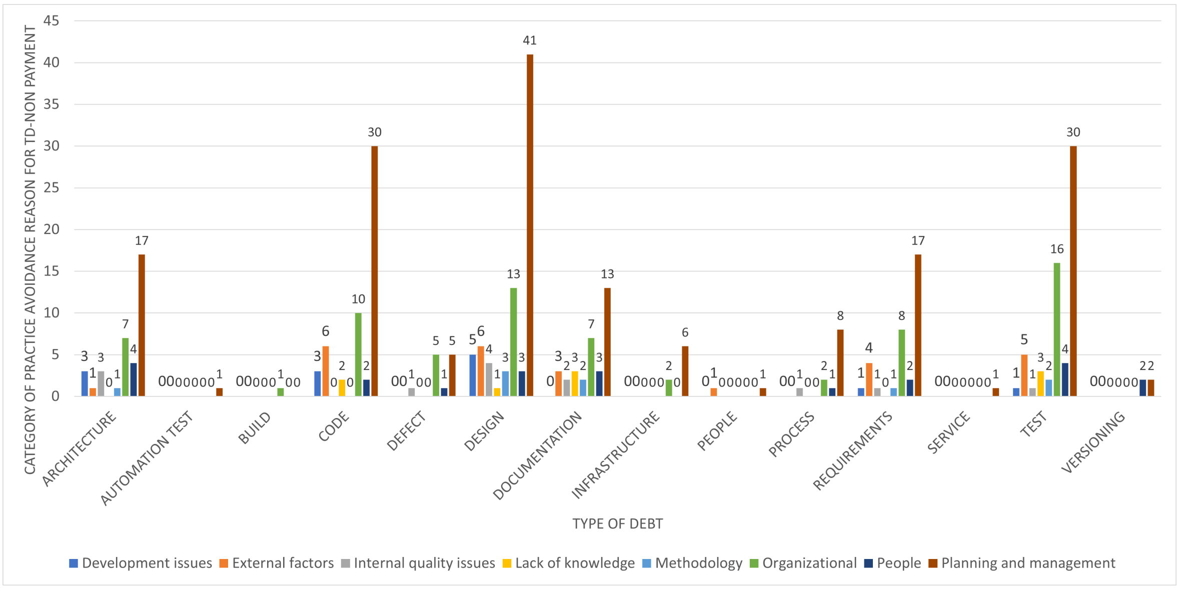


Figure 6.9 Relationship among types of debt and categories of PARs for TD non-payment - absolute values

6.2 DISCUSSION

In this section, we answer the RQ3 and compare the findings with those reported in the related work.

6.2.1 Answering Research Question 3

Although the respondents (∼60%) have not commonly paid off TD items in their projects, we could find a set of 32 practices related to the payment of TD, which fall into four broad types: practices that directly result in debt item payment, practices that help create a favorable scenario for future debt payment, TD prevention practices, and TD prioritization practices. *Code refactoring*, *investing effort on TD payment activities*, and *design refactoring* are the most cited practices.

We found eight practices for directly paying off TD items: *code refactoring*, *design refactoring*, *adjusting code to follow good programming practices*, *update system documentation*, *solving technical issues*, *restarting the project from scratch*, *system retirement*, and *bug fixing*. These practices have been used in combination with others, for example, we found that when software practitioners pay debt items off, they are also concerned with the prevention and prioritization of debt items. Moreover, they have changed their work to enable practices related to TD payment in the future. These changes are related to the practices that define a favorable scenario for debt payment.

We identified that TD payment practices are roughly balanced between technical and managerial practices, indicating that TD payment involves different types of roles in software projects. However, most of the payment actions and TD prevention practices are technical. In contrast, most practices for enabling TD payment and for TD prioritization are managerial. This implies that technical practitioners spend effort in activities for paying off and curbing TD items. Still, the execution of these activities needs the support of managers who must spend effort in making changes in the work environment to create conditions for TD payment.

We also found that the identified practices are more commonly concentrated in *methodological* issues of software development, and practitioners have used them for paying off several types of debt: architecture, build, code, defect, design, documentation, infrastructure, people, requirements, service, and test. This is an indication that the process followed by software practitioners plays a central role for TD payment. Process improvements seem to be a good starting point for TD payment initiatives.

The study identified 27 PARs for TD non-payment. Among them, *focusing on short term goals*, *lack of organizational interest*, and *lack of time* are the most cited PARs, revealing that managerial decisions are quite decisive.

We identified that these PARs can be either a decision taken by the team to intentionally not pay off debt or an impediment that hinders the payment of debt items regardless of the practitioners intentions. *Impediments* are slightly more commonly faced (∼65%) than *decision factors*. The PARs are more used for justifying the non-payment of design, test, code, architecture, documentation, and requirements debt items. Also, we found more managerial PARs for TD non-payment than technical ones, indicating that the management view is decisive for the non-payment of TD.

6.2.2 Comparison to Related Work

This section compares related work on TD payment practices and the PARs for TD non-payment.

6.2.2.1 TD Payment Practices. We compared the TD payment-related practices found in this work and the ones reported in the technical literature. During the comparison, we realized that some practices were at different abstraction levels in different studies. For instance, the practices *remove the business logic inside the communication layer* and *move the business logic to the services* were reported by Toledo *et al.* (2019). On the other hand, we have *adjusting code to follow good programming practices practice*, which is at a higher abstraction level than those from Toledo *et al.* (2019) but is similar. Then, we grouped similar practices considering their different level of abstraction.

The practices for **enabling TD payment** that were found both in our work and in at least some other past studies are: *changing the project management*, *communicating the customer of TD items*, *conducting risk analysis*, *improving the development process*, *improving the team collaboration*, *increasing the project budget*, *investing effort on TD payment activities*, and *negotiating deadline extension*. **TD payment actions** that appear both in our work and related work are *adjusting code to follow good programming practices*, *bug fixing*, *code refactoring*, *design refactoring*, *solving technical issues*, and *update system documentation*. The **TD prevention practices** reported both in our study and past work are *code reviewing*, *improving requirement elicitation process*, *investing effort on testing activities*, *monitoring and controlling project activities*, and *using external tools*. *Prioritizing TD items* practice was confirmed in two related works in addition to ours. However, our study did not identify sixteen practices recognized in related work. These practices include *quantifying TD*, *using palliative solutions*, and *service developers must learn new technologies*. The distinctions and overlaps between the unique TD payment-related practices found in our work and the related work are shown in Figure 6.10. The complete comparison is available in Appendix Table D.9.

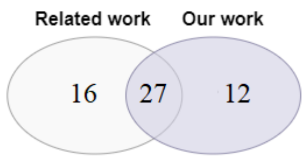


Figure 6.10 Number of unique TD payment-related practices found in our work and the related work

Regarding the distinctions, the related works mainly found practices to improve communication between stakeholders, reduce technical problems and increase the technical capacity of the team. On the other hand, we found practices associated with both technical and managerial aspects of software teams.

Table 6.10 presents the comparison between the categories we identified in this work and the categories reported by Li, Avgeriou and Liang (2015), Behutiye *et al.* (2017), and Bonfim and Benitti (2022). We confirmed the categories *internal quality issues*, *methodology*, *development issues*, and *external quality issues*. The other categories (*planning and management*, *people*, *organizational*, and *infrastructure*) were only identified in our work.

Table 6.10 Comparison of related work on categories of TD payment-related practices

	Categories from			**Overlapping degree**
Our study	**Li, Avgeriou and Liang (2015)**	**Behutiye *et al.* (2017)**	**Bonfim and Benitti (2022)**	
Internal quality issues	Refactoring	Refactoring	-	Total
	Rewriting	-	-	Total
Methodology	Automation	Test automation	-	Partial
	Repackaging	-	-	Partial
Development issues	Reengineering	Code analysis	Reducing requirements debts	Partial
	Fault tolerance	-	-	Partial
External quality issues	Bug fixing	-	-	Total
Planning and management	-	-	-	-
People	-	-	-	-
Organizational	-	-	-	-
Infrastructure	-	-	-	-

In summary, our findings on TD payment-related practices and their categories complement and extend the set of information already reported in the technical literature.

6.2.2.2 PARs for TD Non-payment. Table 6.11 compares the PARs for TD non-payment found in this work with those reported by Bomfim Jr and Santos (2017). Only the PARs *lack of testing*, *complexity of the project*, and *effort* have been reported previously. Thus, our results confirm and extend the current knowledge on PARs for TD non-payment.

6.3 CONCLUDING REMARKS

This chapter reveals the practitioners point of view on TD payment, reporting the practices used by software practitioners to pay off TD items and the PARs that justify the TD non-payment. Also, we identify several dimensions of these practices and PARs, by grouping them into distinct types, nature, and categories, and recognize relationships between practices and types of them, and PARs and types of debt. Lastly, we compare the set of practices and PARs with ones reported in related work to verify how our results expand the TD payment knowledge. Threats to validity were previously discussed in Chapter 3.

Table 6.11 Comparison of related work on PARs for TD non-payment

Type	Our Study	Bomfim Jr and Santos (2017)
Decision Factor	Decision to do not change the framework	-
	Focusing on short term goals	-
	Insufficient management view about TD payment	-
	Lack of adoption of lessons learned	-
	Lack of committed team	-
	Lack of external auditing	-
	Lack of organizational interest	-
	Lack of testing	Lack of test coverage or excessive manual testing
	Non-application of mitigation actions on TD causes	-
	Number of TD items	-
	TD items do not affect the user	-
	TD items do not have "interest"	-
Impediment	Complexity of the project	Concern of impacting some module because the team does not know all parts of the code to carry out a deeper impact analysis
	Complexity of the TD item	-
	Cost	-
	Customer decision	-
	Effort	Low impact for business and high effort
	High team turnover	-
	Lack of access on component code	-
	Lack of knowledge on TD	-
	Lack of monitoring of TD items	-
	Lack of resources	-
	Lack of technical knowledge	-
	Lack of time	-
	Risk for the project	-
	Team overload	-
	The project was discontinued	-

The next chapter presents the updated conceptual model for TD and TD management maps we proposed to organize the set of practices and PARs we found for TD prevention, monitoring, and payment. Also, it presents the follow-up survey we conducted to assess the model and map in terms of supporting TD management activities.

Chapter 7

This chapter introduces the updated version of the TD conceptual model we extended using the concepts of TD prevention, monitoring, and payment learned from the state of the practice. It also presents the conceptual map that organize the set of practices and PARs for TD prevention, monitoring, and payment. Lastly, it provides and discusses the assessment of the model and maps we performed by conducting a follow-up survey with InsighTD participants.

TD CONCEPTUAL MODEL AND TD MANAGEMENT CONCEPTUAL MAPS

In the previous chapters, we presented the state of practice of Technical debt (TD) prevention, monitoring, and payment activities, revealing and discussing the practices used to carry out these activities and the Practice avoidance reasons (PARs) used to explain the non-implementation of these practices. However, organizing this body of knowledge is essential to make it more usable by researchers and software practitioners in their activities.

This chapter aims to answer **RQ4: How to organize the body of knowledge composed of prevention, monitoring, and payment practices - and Practice avoidance reasons (PARs) for TD non-prevention, non-monitoring, and non-payment - to support TD management?** For this, we organize the body of knowledge using different views to consolidate the set of information on TD management.

Initially, Section 7.1 extends the conceptual model for TD proposed by Avgeriou *et al.* (2016) and Izurieta *et al.* (2016) and evolved by Rios, Mendonça and Spinola (2018), including the concepts of TD prevention, monitoring, and payment from the state of practice. Conceptual models are descriptions of a phenomenon in a domain at some level of abstraction. These descriptions can be expressed in a semi-formal or formal visual language (KROGSTIE, 2017). The original version of the conceptual model was defined as a Unified modeling language (UML) class diagram by consolidating the key concepts of TD in the Dagstuhl Seminar 16162 ((AVGERIOU *et al.*, 2016; IZURIETA *et al.*, 2016). By including the concepts of TD management identified from a tertiary study, Rios, Mendonça and Spinola (2018) evolved the conceptual model.

Next, Section 7.2 organizes the set of practices and PARs in conceptual maps. For this, we were inspired by evidence briefings used to disseminate research findings to practitioners (CARTAXO *et al.*, 2016). Instead of using a one-page document to transfer knowledge acquired from empirical studies, we evolved the structure previously defined by Rios *et al.* (2020), resulting in maps composed of categories and their practices and PARs, along with their types and nature. We depict a map for each TD management activity (prevention, monitoring, and payment).

To close this chapter, Section 7.3 presents the follow-up survey we conducted to assess the TD payment conceptual model and the TD payment conceptual map, allowing us to answer RQ4.1: How are the proposed artifacts characterized in terms of its support for TD management activities?. Section 7.4 discusses the results of the studies. Finally, Section 7.5 presents the concluding remarks of Chapter 7.

The main contributions of this chapter are:

- An extension of the conceptual model for TD initially proposed by Avgeriou *et al.* (2016) and Izurieta *et al.* (2016) and evolved by Rios, Mendonça and Spinola (2018), including the concepts of PARs and practices and their types, nature, and categories.
- A set of conceptual maps that organize the set of practices and PARs, supporting the practical use of the results by software practitioners.
- An assessment of the TD payment conceptual map and the TD payment map by considering the perception of software practitioners.

The material included in this chapter is based on the following papers: (i) Surveying Software Practitioners on Technical Debt Payment Practices and Reasons for not Paying off Debt Items (FREIRE *et al.*, 2020b), (ii) Software Practitioners Point of View on Technical Debt Payment (FREIRE *et al.*, 2023), (iii) Hearing the Voice of Software Practitioners on Technical Debt Monitoring: Understanding Monitoring Practices and the Practices' Avoidance Reasons (FREIRE *et al.*, 2022), and (iv) A Comprehensive View on TD Prevention Practices and Reasons for not Prevent It (FREIRE *et al.*, 2023b).

7.1 A CONCEPTUAL MODEL FOR TD MANAGEMENT

Figure 7.1 presents our extended conceptual model for TD. It shows in blue our contribution to the original conceptual model for TD. The classes in white were defined by Avgeriou *et al.* (2016), Izurieta *et al.* (2016), and Rios, Mendonça and Spinola (2018). These classes represent TD items properties, artifacts, and elements.

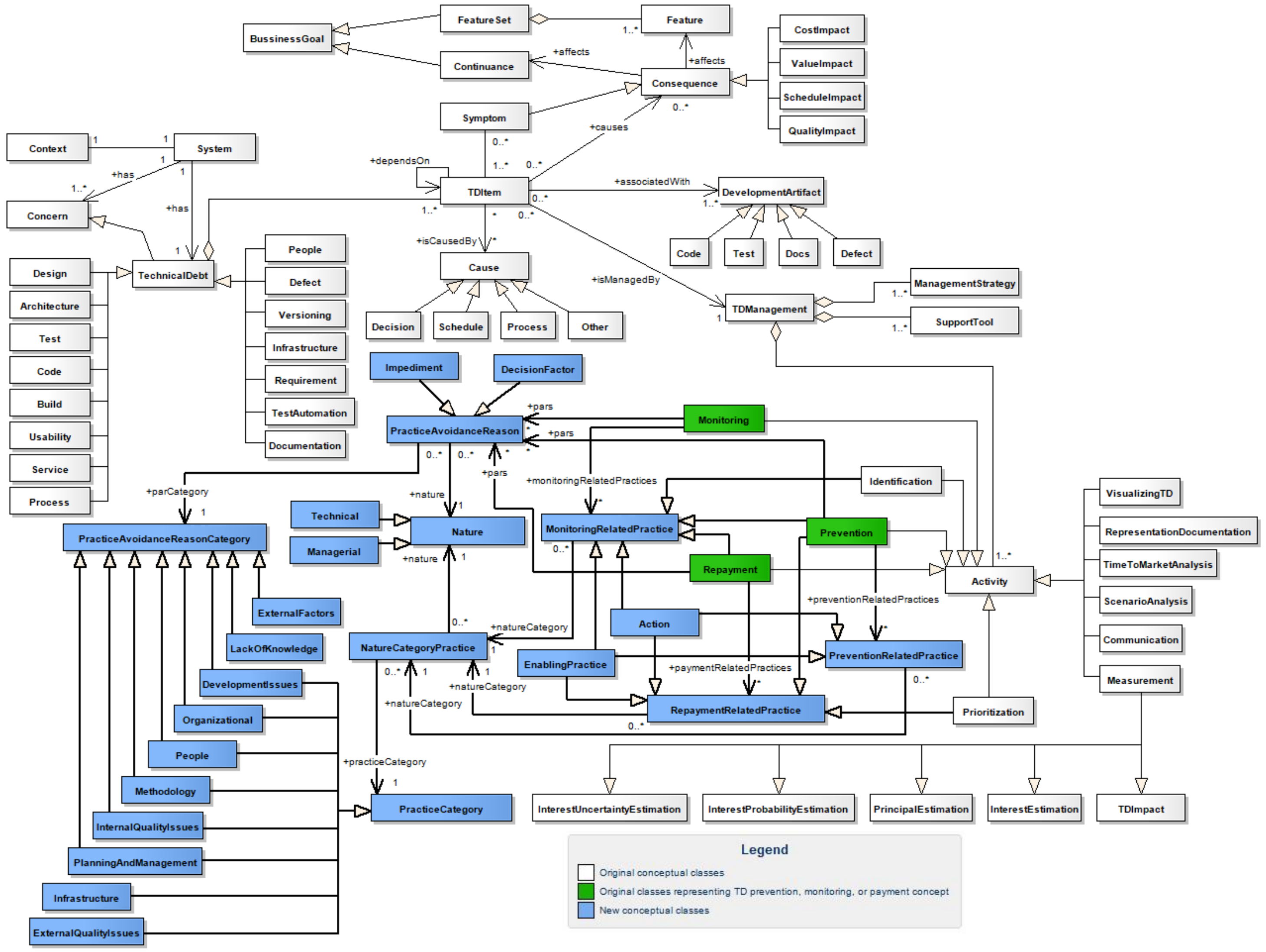


Figure 7.1 Extended conceptual model for technical debt

According to the conceptual model, a system has many concerns, where TD is one of them. TD can be specialized into 15 TD types (e.g., architecture, code, design, and test debt) and is composed of TD items. Different factors, like a decision, schedule pressure, inappropriate processes, and others, can cause a TD item. These factors are specializations of the *Cause* class. Also, TD items can bring consequences to the projects, affecting their business goals. These consequences can impact a features cost, value, schedule, or quality or the project continuance. A TD item is associated with one or more artifacts of the software development process (e.g., code, test, and documentation) represented as specializations of the *DevelopmentArtifact* class. Finally, a TD item can be related to TD management, which is defined as an aggregation of management strategies, support tools, and activities. The latter can be specialized into 11 types, such as prevention, payment, monitoring, and measurement.

Our extension to the conceptual model includes all the organized empirical evidence in this work related to the concept of TD prevention, monitoring, and payment (classes in green). These classes are associated with two lists. One is formed by the PARs (*PracticeAvoidanceReason* class), and the other represents the related practices (*PreventionRelatedPractice*, *MonitoringRelatedPractice*, and *PaymentRelatedPractice* classes).

The *PracticeAvoidanceReason* class can assume two different roles: an impediment or a decision factor. Also, a PAR has a nature that can be technical or managerial. The PARs that lead to the non-prevention, non-monitoring, or non-payment of TD items are from different categories (e.g., *ExternalFactors*, *LackOfKnowledge*, and *DevelopmentIssues*).

Through a relationship with *NatureCategoryPractice* class, the *PreventionRelatedPractice*, *MonitoringRelatedPractice*, and *PaymentRelatedPractice* classes also have a nature (technical or managerial), and the practices are organized into categories (e.g., *ExternalQualityIssues*, *InternalQualityIssues*, and *PlanningAndManagement*). These classes are specialized by the *Action* and *EnablingPractice* classes. The first allows the direct execution of the TD management activity, while the latter supports the execution of them. Lastly, a monitoring-related practice can be identification, prevention, or payment activities, and a payment-related practice can be prioritization or prevention activities.

By generalizing the *MonitoringRelatedPractice* class into the classes *Identification*, *Prevention*, and *Repayment* and the *PaymentRelatedPractice* class into the classes *Prevention* and *Prioritization*, we found a relationship among the identification, payment, prevention, and prioritization activities. This relationship indicates an integration of such kinds of TD management activities. The importance of such relationships was previously discussed by Rios, Mendonça and Spinola (2018) when the authors reported that current tools and strategies for TD management are limited in the sense that they only cover isolated TD management activities. These tools and strategies can be updated considering the different types of practices we described in the *MonitoringRelatedPractice* and *PaymentRelatedPractice* classes and their specializations. For example, suppose a strategy only focuses on supporting TD prevention practices. In that case, other practices for TD payment and prioritization can be included in the strategy to support the application of combined practices from these three TD management activities (prevention, payment, and prioritization). Also, the practices from the three TD management activities could be combined in a tool, and the user can choose one or more practices to apply together in their project. This approach would not guarantee the use of practices from different activities but would present the opportunity to do so.

7.2 TD PREVENTION, MONITORING, AND PAYMENT MAPS

As evidence briefings have been used to disseminate research findings with practitioners (CARTAXO *et al.*, 2016), we use the same approach to summarize the set of practices and PARs, along with their categories, types, and nature. Figures 7.2, 7.3, and 7.4 show the maps supporting development teams in TD prevention, monitoring, and payment activities, respectively. The structure and composition of those maps will be discussed in Section 7.2.1.

7.2.1 About the Conceptual Maps

The maps organize practices and PARs grouped by category. The rectangles with rounded edges group the entire set of practices and PARs. Rectangles with dashed lines represent the categories of practices and PARs. In each category, the map shows the percentage associated with the category and its practices or PARs. To compute the percentages associated with each practice and PARs, we summed up the number of occurrences for each of them. Then, we divided this value by the number of projects for which any practice or reason was cited. For example, *better project management* prevention practice (Figure 7.2) was cited by 43 participants. As we had 546 participants indicating that a TD item could be prevented in their projects, better project management was cited in 8% (43/546*100) of them. We performed the same process for computing the percentage of each category.

Looking at the percentages in the map, we see that TD prevention-related practices from the categories *planning and management* and *methodology* are more commonly cited in 53% and 39% of the projects, respectively. In the *planning and management* category, the *better project management* prevention action stands out, as it was cited by 8% of the projects. On the right side of the map, we can notice that the PARs from the categories *planning and management* and *development issues* are more cited for explaining the TD non-prevention in 49% and 22% of the projects, respectively. In the *planning and management* category, the *short deadline* impediment stands out, cited in 29% of the projects.

Small rectangles indicate the nature of a practice or PAR, with black rectangles indicating a technical nature, while white rectangles denote managerially. For example, the *development issues* practice category (Figure 7.4) has three managerial (*changing project scope*, *restarting the project from scratch*, and *system retirement*) and three technical practices (*adjusting code to follow good programming practices*, *update system documentation*, and *solving technical issues*).

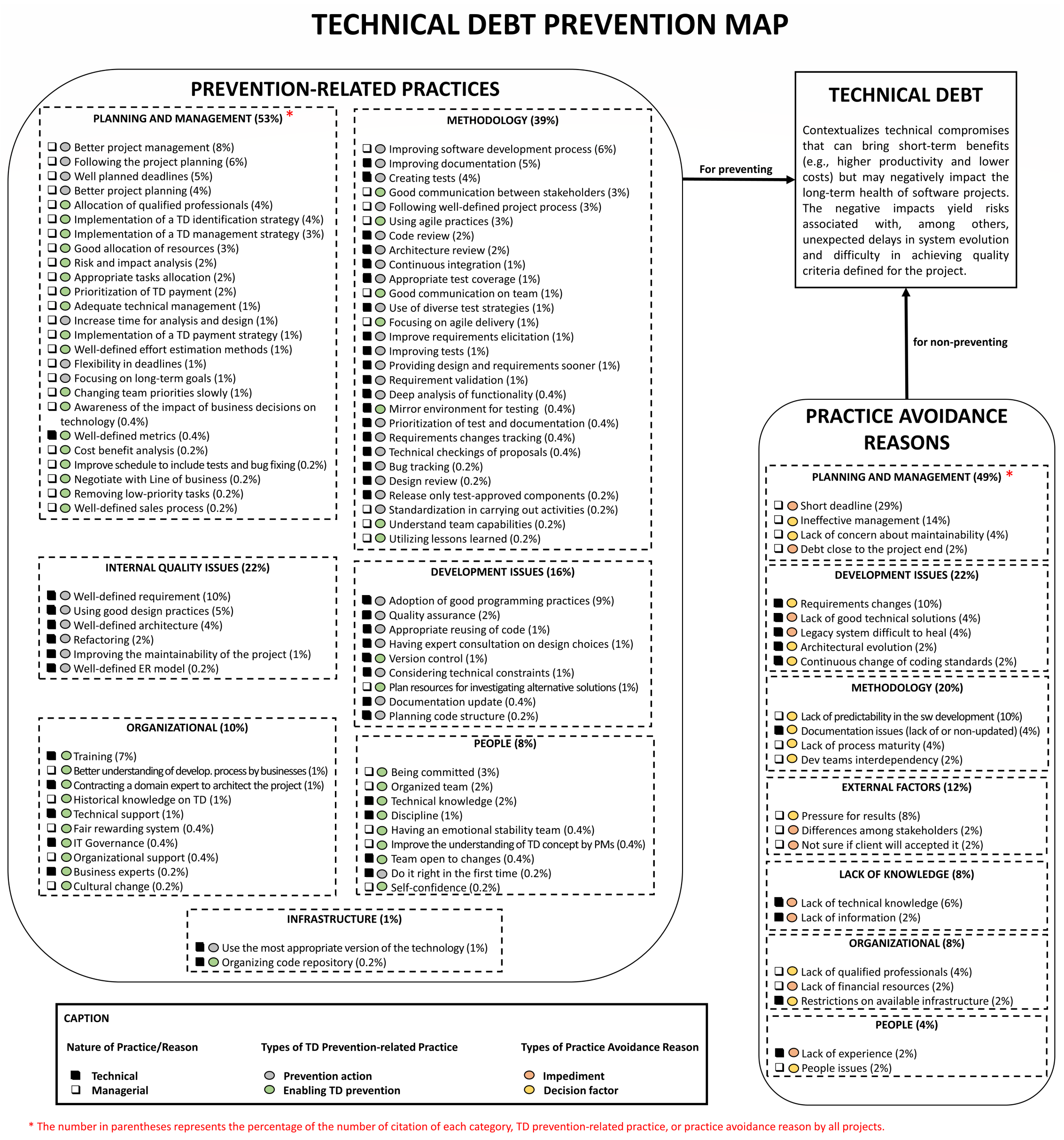

TECHNICAL DEBT PREVENTION MAP
PREVENTION-RELATED PRACTICES
PLANNING AND MANAGEMENT (53%) *
Better project management (8%)
Following the project planning (6%)
Well planned deadlines (5%)
Better project planning (4%)
Allocation of qualified professionals (4%)
Implementation of a TD identification strategy (4%)
Implementation of a TD management strategy (3%)
Good allocation of resources (3%)
Risk and impact analysis (2%)
Appropriate tasks allocation (2%)
Prioritization of TD payment (2%)
Adequate technical management (1%)
Increase time for analysis and design (1%)
Implementation of a TD payment strategy (1%)
Well-defined effort estimation methods (1%)
Flexibility in deadlines (1%)
Focusing on long-term goals (1%)
Changing team priorities slowly (1%)
Awareness of the impact of business decisions on technology (0.4%)
Well-defined metrics (0.4%)
Cost benefit analysis (0.2%)
Improve schedule to include tests and bug fixing (0.2%)
Negotiate with Line of business (0.2%)
Removing low-priority tasks (0.2%)
Well-defined sales process (0.2%)
METHODOLOGY (39%)
Improving software development process (6%)
Improving documentation (5%)
Creating tests (4%)
Good communication between stakeholders (3%)
Following well-defined project process (3%)
Using agile practices (3%)
Code review (2%)
Architecture review (2%)
Continuous integration (1%)
Appropriate test coverage (1%)
Good communication on team (1%)
Use of diverse test strategies (1%)
Focusing on agile delivery (1%)
Improve requirements elicitation (1%)
Improving tests (1%)
Providing design and requirements sooner (1%)
Requirement validation (1%)
Deep analysis of functionality (0.4%)
Mirror environment for testing (0.4%)
Prioritization of test and documentation (0.4%)
Requirements changes tracking (0.4%)
Technical checkings of proposals (0.4%)
Bug tracking (0.2%)
Design review (0.2%)
Release only test-approved components (0.2%)
Standardization in carrying out activities (0.2%)
Understand team capabilities (0.2%)
Utilizing lessons learned (0.2%)
INTERNAL QUALITY ISSUES (22%)
Well-defined requirement (10%)
Using good design practices (5%)
Well-defined architecture (4%)
Refactoring (2%)
Improving the maintainability of the project (1%)
Well-defined ER model (0.2%)
DEVELOPMENT ISSUES (16%)
Adoption of good programming practices (9%)
Quality assurance (2%)
Appropriate reusing of code (1%)
Having expert consultation on design choices (1%)
Version control (1%)
Considering technical constraints (1%)
Plan resources for investigating alternative solutions (1%)
Documentation update (0.4%)
Planning code structure (0.2%)
ORGANIZATIONAL (10%)
Training (7%)
Better understanding of develop. process by businesses (1%)
Contracting a domain expert to architect the project (1%)
Historical knowledge on TD (1%)
Technical support (1%)
Fair rewarding system (0.4%)
IT Governance (0.4%)
Organizational support (0.4%)
Business experts (0.2%)
Cultural change (0.2%)
PEOPLE (8%)
Being committed (3%)
Organized team (2%)
Technical knowledge (2%)
Discipline (1%)
Having an emotional stability team (0.4%)
Improve the understanding of TD concept by PMs (0.4%)
Team open to changes (0.4%)
Do it right in the first time (0.2%)
Self-confidence (0.2%)
INFRASTRUCTURE (1%)
Use the most appropriate version of the technology (1%)
Organizing code repository (0.2%)
For preventing
TECHNICAL DEBT
Contextualizes technical compromises that can bring short-term benefits (e.g., higher productivity and lower costs) but may negatively impact the long-term health of software projects. The negative impacts yield risks associated with, among others, unexpected delays in system evolution and difficulty in achieving quality criteria defined for the project.
for non-preventing
PRACTICE AVOIDANCE REASONS
PLANNING AND MANAGEMENT (49%) *
Short deadline (29%)
Ineffective management (14%)
Lack of concern about maintainability (4%)
Debt close to the project end (2%)
DEVELOPMENT ISSUES (22%)
Requirements changes (10%)
Lack of good technical solutions (4%)
Legacy system difficult to heal (4%)
Architectural evolution (2%)
Continuous change of coding standards (2%)
METHODOLOGY (20%)
Lack of predictability in the sw development (10%)
Documentation issues (lack of or non-updated) (4%)
Lack of process maturity (4%)
Dev teams interdependency (2%)
EXTERNAL FACTORS (12%)
Pressure for results (8%)
Differences among stakeholders (2%)
Not sure if client will accepted it (2%)
LACK OF KNOWLEDGE (8%)
Lack of technical knowledge (6%)
Lack of information (2%)
ORGANIZATIONAL (8%)
Lack of qualified professionals (4%)
Lack of financial resources (2%)
Restrictions on available infrastructure (2%)
PEOPLE (4%)
Lack of experience (2%)
People issues (2%)
CAPTION
Nature of Practice/Reason
■ Technical
□ Managerial
Types of TD Prevention-related Practice
● Prevention action
● Enabling TD prevention
Types of Practice Avoidance Reason
● Impediment
● Decision factor
* The number in parentheses represents the percentage of the number of citation of each category, TD prevention-related practice, or practice avoidance reason by all projects.

Figure 7.2 Technical debt prevention map

Small circles represent the type of practice or PAR. We used a specific color for each type. For example, the *internal quality issues* category (Figure 7.3), from the monitoring-related practices rectangle, has one white circle (*identifying TD items*) representing a practice of the *TD identification* type, one brown circle (refactoring) of the *TD payment* type, and two gray circles (*understanding the cause of TD item* and *identify the worst debt areas*) of the *monitoring action* type.

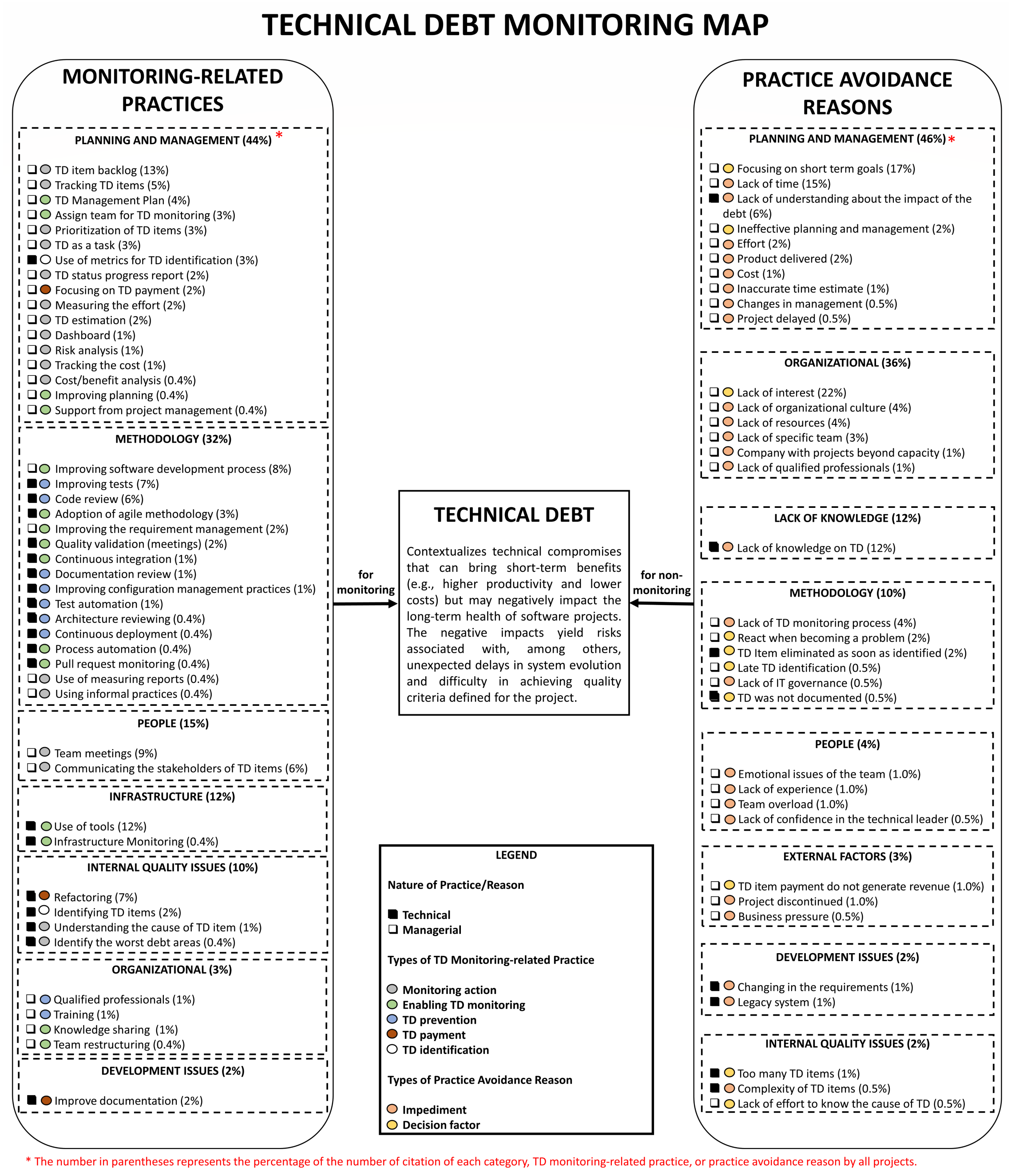

TECHNICAL DEBT MONITORING MAP
MONITORING-RELATED PRACTICES
PLANNING AND MANAGEMENT (44%) *
TD item backlog (13%)
Tracking TD items (5%)
TD Management Plan (4%)
Assign team for TD monitoring (3%)
Prioritization of TD items (3%)
TD as a task (3%)
Use of metrics for TD identification (3%)
TD status progress report (2%)
Focusing on TD payment (2%)
Measuring the effort (2%)
TD estimation (2%)
Dashboard (1%)
Risk analysis (1%)
Tracking the cost (1%)
Cost/benefit analysis (0.4%)
Improving planning (0.4%)
Support from project management (0.4%)
METHODOLOGY (32%)
Improving software development process (8%)
Improving tests (7%)
Code review (6%)
Adoption of agile methodology (3%)
Improving the requirement management (2%)
Quality validation (meetings) (2%)
Continuous integration (1%)
Documentation review (1%)
Improving configuration management practices (1%)
Test automation (1%)
Architecture reviewing (0.4%)
Continuous deployment (0.4%)
Process automation (0.4%)
Pull request monitoring (0.4%)
Use of measuring reports (0.4%)
Using informal practices (0.4%)
PEOPLE (15%)
Team meetings (9%)
Communicating the stakeholders of TD items (6%)
INFRASTRUCTURE (12%)
Use of tools (12%)
Infrastructure Monitoring (0.4%)
INTERNAL QUALITY ISSUES (10%)
Refactoring (7%)
Identifying TD items (2%)
Understanding the cause of TD item (1%)
Identify the worst debt areas (0.4%)
ORGANIZATIONAL (3%)
Qualified professionals (1%)
Training (1%)
Knowledge sharing (1%)
Team restructuring (0.4%)
DEVELOPMENT ISSUES (2%)
Improve documentation (2%)
for monitoring
TECHNICAL DEBT
Contextualizes technical compromises that can bring short-term benefits (e.g., higher productivity and lower costs) but may negatively impact the long-term health of software projects. The negative impacts yield risks associated with, among others, unexpected delays in system evolution and difficulty in achieving quality criteria defined for the project.
for non-monitoring
PRACTICE AVOIDANCE REASONS
PLANNING AND MANAGEMENT (46%) *
Focusing on short term goals (17%)
Lack of time (15%)
Lack of understanding about the impact of the debt (6%)
Ineffective planning and management (2%)
Effort (2%)
Product delivered (2%)
Cost (1%)
Inaccurate time estimate (1%)
Changes in management (0.5%)
Project delayed (0.5%)
ORGANIZATIONAL (36%)
Lack of interest (22%)
Lack of organizational culture (4%)
Lack of resources (4%)
Lack of specific team (3%)
Company with projects beyond capacity (1%)
Lack of qualified professionals (1%)
LACK OF KNOWLEDGE (12%)
Lack of knowledge on TD (12%)
METHODOLOGY (10%)
Lack of TD monitoring process (4%)
React when becoming a problem (2%)
TD Item eliminated as soon as identified (2%)
Late TD identification (0.5%)
Lack of IT governance (0.5%)
TD was not documented (0.5%)
PEOPLE (4%)
Emotional issues of the team (1.0%)
Lack of experience (1.0%)
Team overload (1.0%)
Lack of confidence in the technical leader (0.5%)
EXTERNAL FACTORS (3%)
TD item payment do not generate revenue (1.0%)
Project discontinued (1.0%)
Business pressure (0.5%)
DEVELOPMENT ISSUES (2%)
Changing in the requirements (1%)
Legacy system (1%)
INTERNAL QUALITY ISSUES (2%)
Too many TD items (1%)
Complexity of TD items (0.5%)
Lack of effort to know the cause of TD (0.5%)
LEGEND
Nature of Practice/Reason
Technical
Managerial
Types of TD Monitoring-related Practice
Monitoring action
Enabling TD monitoring
TD prevention
TD payment
TD identification
Types of Practice Avoidance Reason
Impediment
Decision factor
* The number in parentheses represents the percentage of the number of citation of each category, TD monitoring-related practice, or practice avoidance reason by all projects.

Figure 7.3 Technical debt monitoring map

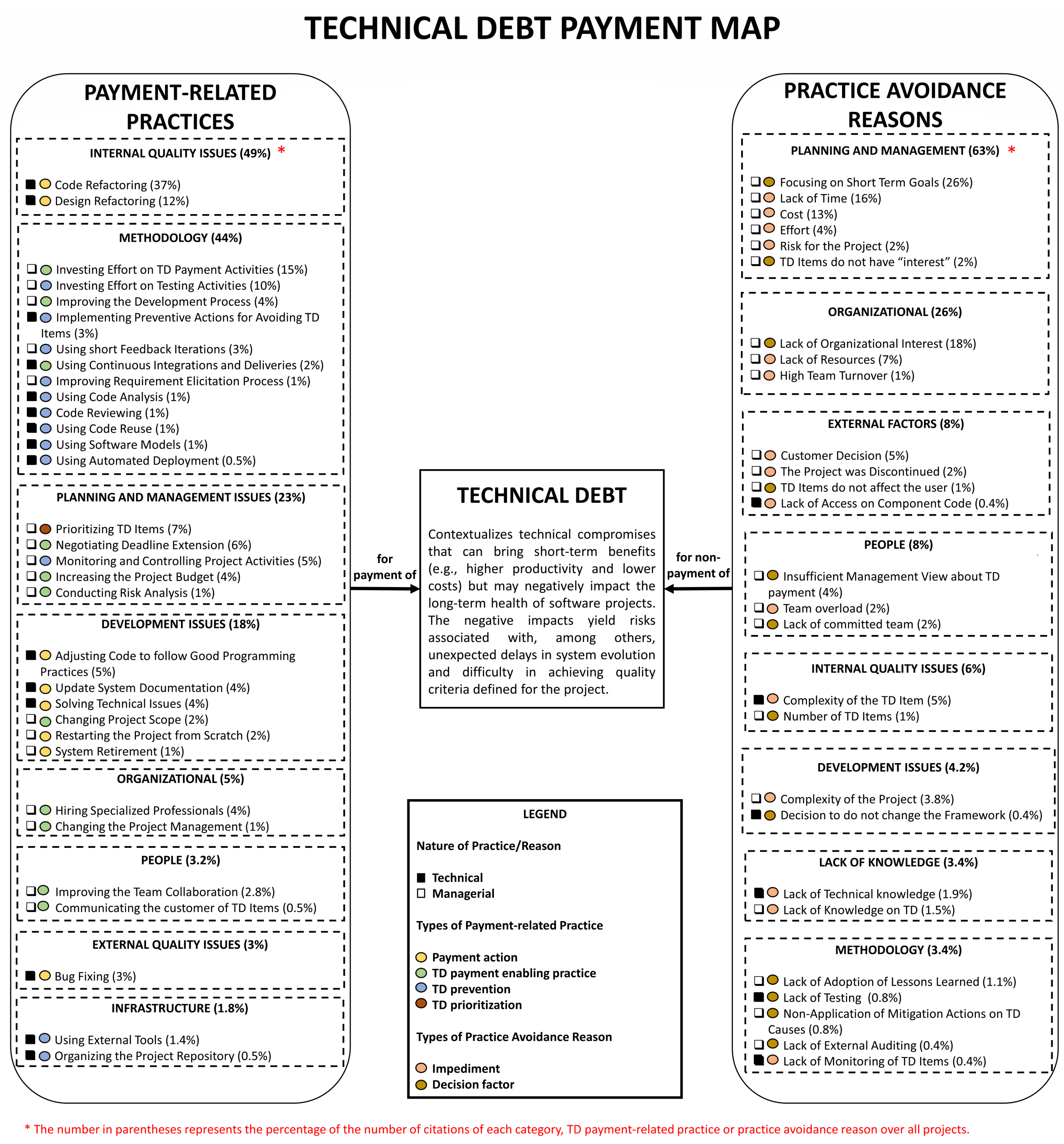

TECHNICAL DEBT PAYMENT MAP
PAYMENT-RELATED PRACTICES
INTERNAL QUALITY ISSUES (49%) *
■ 🟡 Code Refactoring (37%)
■ 🟡 Design Refactoring (12%)
METHODOLOGY (44%)
□ 🟢 Investing Effort on TD Payment Activities (15%)
□ 🔵 Investing Effort on Testing Activities (10%)
□ 🟢 Improving the Development Process (4%)
■ 🔵 Implementing Preventive Actions for Avoiding TD Items (3%)
□ 🔵 Using short Feedback Iterations (3%)
■ 🟢 Using Continuous Integrations and Deliveries (2%)
□ 🔵 Improving Requirement Elicitation Process (1%)
■ 🔵 Using Code Analysis (1%)
■ 🔵 Code Reviewing (1%)
■ 🔵 Using Code Reuse (1%)
■ 🔵 Using Software Models (1%)
■ 🔵 Using Automated Deployment (0.5%)
PLANNING AND MANAGEMENT ISSUES (23%)
□ 🔴 Prioritizing TD Items (7%)
□ 🟢 Negotiating Deadline Extension (6%)
□ 🔵 Monitoring and Controlling Project Activities (5%)
□ 🟢 Increasing the Project Budget (4%)
□ 🟢 Conducting Risk Analysis (1%)
DEVELOPMENT ISSUES (18%)
■ 🟡 Adjusting Code to follow Good Programming Practices (5%)
■ 🟡 Update System Documentation (4%)
■ 🟡 Solving Technical Issues (4%)
□ 🟢 Changing Project Scope (2%)
□ 🟡 Restarting the Project from Scratch (2%)
□ 🟡 System Retirement (1%)
ORGANIZATIONAL (5%)
□ 🟢 Hiring Specialized Professionals (4%)
□ 🟢 Changing the Project Management (1%)
PEOPLE (3.2%)
□ 🟢 Improving the Team Collaboration (2.8%)
□ 🟢 Communicating the customer of TD Items (0.5%)
EXTERNAL QUALITY ISSUES (3%)
■ 🟡 Bug Fixing (3%)
INFRASTRUCTURE (1.8%)
■ 🔵 Using External Tools (1.4%)
■ 🔵 Organizing the Project Repository (0.5%)
for payment of
TECHNICAL DEBT
Contextualizes technical compromises that can bring short-term benefits (e.g., higher productivity and lower costs) but may negatively impact the long-term health of software projects. The negative impacts yield risks associated with, among others, unexpected delays in system evolution and difficulty in achieving quality criteria defined for the project.
for non-payment of
PRACTICE AVOIDANCE REASONS
PLANNING AND MANAGEMENT (63%) *
□ 🟤 Focusing on Short Term Goals (26%)
□ 🟠 Lack of Time (16%)
□ 🟠 Cost (13%)
□ 🟠 Effort (4%)
□ 🟠 Risk for the Project (2%)
□ 🟤 TD Items do not have “interest” (2%)
ORGANIZATIONAL (26%)
□ 🟤 Lack of Organizational Interest (18%)
□ 🟠 Lack of Resources (7%)
□ 🟠 High Team Turnover (1%)
EXTERNAL FACTORS (8%)
□ 🟠 Customer Decision (5%)
□ 🟠 The Project was Discontinued (2%)
□ 🟤 TD Items do not affect the user (1%)
■ 🟠 Lack of Access on Component Code (0.4%)
PEOPLE (8%)
□ 🟤 Insufficient Management View about TD payment (4%)
□ 🟠 Team overload (2%)
□ 🟤 Lack of committed team (2%)
INTERNAL QUALITY ISSUES (6%)
■ 🟠 Complexity of the TD Item (5%)
□ 🟤 Number of TD Items (1%)
DEVELOPMENT ISSUES (4.2%)
□ 🟠 Complexity of the Project (3.8%)
■ 🟤 Decision to do not change the Framework (0.4%)
LACK OF KNOWLEDGE (3.4%)
■ 🟠 Lack of Technical knowledge (1.9%)
□ 🟠 Lack of Knowledge on TD (1.5%)
METHODOLOGY (3.4%)
□ 🟤 Lack of Adoption of Lessons Learned (1.1%)
■ 🟤 Lack of Testing (0.8%)
□ 🟤 Non-Application of Mitigation Actions on TD Causes (0.8%)
□ 🟤 Lack of External Auditing (0.4%)
■ 🟠 Lack of Monitoring of TD Items (0.4%)
LEGEND
Nature of Practice/Reason
■ Technical
□ Managerial
Types of Payment-related Practice
🟡 Payment action
🟢 TD payment enabling practice
🔵 TD prevention
🔴 TD prioritization
Types of Practice Avoidance Reason
🟠 Impediment
🟤 Decision factor
* The number in parentheses represents the percentage of the number of citations of each category, TD payment-related practice or practice avoidance reason over all projects.

Figure 7.4 Technical debt payment map

7.2.2 Using the Conceptual Map

Software practitioners can use the map to identify and understand TD prevention, monitoring, and payment practices. Considering the map as a benchmark, practitioners can choose the practices that best fit their projects' context. In addition, they can also identify new practices to use in combination with ones they have not used yet. For example, if a team applies *well-defined requirements* and *refactoring* prevention practices from *internal quality issues* category (see Figure 7.2), the team can apply other practices from this category (*well-defined architecture*, *using good design practices*, *improving the maintainability of the project*, and *well-defined ER model*).

Likewise, software practitioners can also use the map to identify and understand PARs for the non-application of TD prevention, monitoring, or payment practices. For example, suppose a team realizes the non-payment of TD items occurs due to *cost*, through the map (Figure 7.4). In that case, the team can perceive that this reason is from the *planning and management* category and identify other PARs composing this category, like *effort*, *risk for the project*, and *lack of time*. A complete view of the PARs can support the team in defining strategies to increase its capacity to manage TD items.

Through the analysis of the nature (technical or managerial) of the PARs, software practitioners can see if their TD prevention, monitoring, or payment difficulties are more due to technical or managerial issues. This can help the organization focus on improvements. For example, primarily managerial PARs might indicate that communication on technical issues needs to be improved so that managers understand the impact of those issues on business concerns. Conversely, primarily technical PARs might suggest that new training, processes, or tools would help.

Lastly, the map can define a favorable environment for preventing, monitoring, or paying off the debt, making the practices and PARs visible to all stakeholders involved in a software project. As a communication device, software teams can identify the problems affecting the prevention activities and discuss these problems with their managers and companies. Then, they can be aware that their decisions are decisive for supporting or not TD prevention, monitoring, or payment initiatives, reducing or increasing the difficulty of preventing, monitoring, or paying off TD items. These problems arise from managers and companies depicted in the map from the PAR categories *planning and management* and *organizational*. Besides, the practice categories *planning and management* and *organizational* can hint at possible solutions (practices) for mitigating those PARs. For example, if a team realizes that *lack of qualified professional* is a PAR existing in its project for TD non-prevention (Figure 7.2), as this PAR is from the *organizational* category, the team along with its company can verify what practices from the organizational category can be applied to eliminate this PARs, such as the practices *contracting a domain expert to architect the project* and *business experts*.

7.3 ASSESSING THE TD CONCEPTUAL MODEL AND TD PAYMENT MAP

We conducted a follow-up survey with some InsighTD participants to investigate the perception of software practitioners on the accuracy and completeness of the proposed TD conceptual model and map. However, for reasons of cost and focus, we only considered (i) a version of the model containing only the extension for TD payment concepts, as shown in Figure 7.5, and (ii) the TD payment conceptual map, as shown in Figure 7.4.

7.3.1 Data collection

The follow-up survey was composed of eleven questions, as shown in Table 7.1, divided into two sections. In the first, the participants provided their perception of the TD conceptual model (Q1 to Q6), and the latter captured the participants perceptions of the TD payment map (Q7 to Q11). Appendix E presents the questionnaire in more detail.

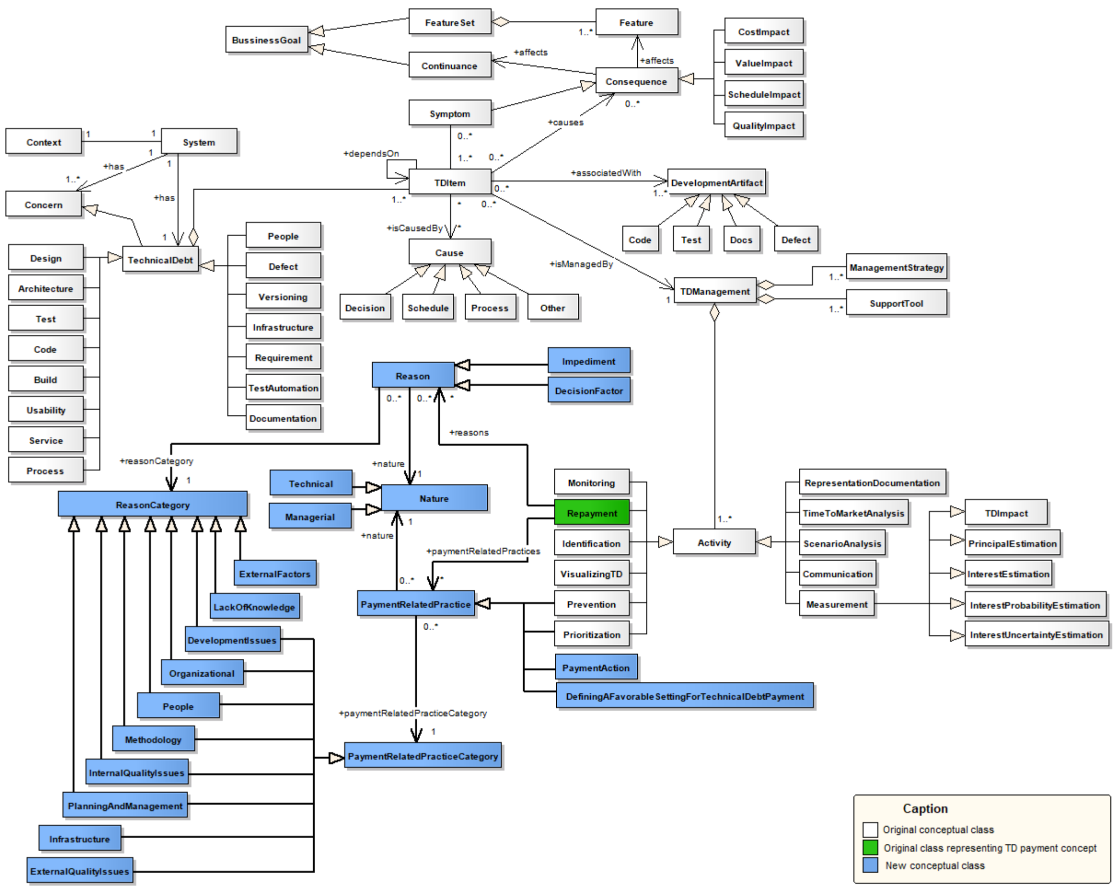


Figure 7.5 Extended conceptual model for technical debt payment

As we only invited practitioners who had answered the InsighTD survey, we only needed to ask participants to provide their email addresses, which we could then use as participant identifiers. This allowed us to match the answers collected in InsighTD to the ones collected in the complementary survey. We then instructed the practitioners to watch a 5.3-minute explanatory video[1] that summarizes the TD conceptual model. We decided to use an explanatory video instead of text to facilitate the practitioners participation. The focus of the video is on our extension of the model. Also, we provided the video transcript and the image of the model in high resolution[2]. The participants then answered

[1]<https://www.youtube.com/watch?v=h9Tlx3ZxuRM>
[2]<https://bit.ly/3P0xB9S>

Q1 to Q6, in which we asked if anything about their perception of TD payment changed and what they learned, if they disagreed with any concept or relationship proposed in the models extension, and how accurately and completely the model represents TD payment concepts.

Next, we instructed the practitioners to watch an 8.5-minute explanatory video[3] summarizing the TD payment map. We also provided the video transcript and the image of the map in high resolution[4]. The participants answered Q7 to Q10, in which we asked if the map would influence participants decisions about how and when to pay debt items. Finally, the participants analyzed three statements (Q11) associated with the usefulness and self-predicted future use of the map, indicating the option that best represented their point of view, according to a 5-point scale from (1) strongly agree to (5) strongly disagree. For example, the statement "I find the TD payment map easy to read, follow and use to support decisions about TD payment" is related to the maps usefulness.

Table 7.1 Follow-up survey questions

Section	No.	Question (Q) Description	Options	Type
1: Technical debt conceptual model	Q1	After watching the video, what changed in your perception about TD payment? For example, are there concepts or ideas that you did not know about before, or that you now think about differently, or that you now understand more completely?	-	Open
	Q2	What have you learned about TD payment from the model?	-	Open
	Q3	Do you disagree with any concept or relationship presented in the conceptual model on TD payment (classes in blue)?	Yes No	Closed
	Q4	If yes, how? If not, why?	-	Open
	Q5	In your opinion, and considering a 0-10 scale, with 0 representing very low accuracy and 10 representing high accuracy, do you think the conceptual model is accurate to represent the TD payment concepts?	0-10 scale	Closed
	Q6	In your opinion, and considering a 0-10 scale, with 0 representing very low completeness and 10 representing high completeness, do you think the conceptual model is complete to represent the TD payment concepts?	0-10 scale	Closed
2: Technical debt payment map	Q7	Would our results, as captured in this map, influence any of your decisions about HOW to pay TD items?	Yes No	Closed
	Q8	If yes, how? If not, why?	-	Open
	Q9	Would our results, as captured in this map, influence any of your decisions about WHEN to pay TD items?	Yes No	Closed
	Q10	If yes, how? If not, why?	-	Open
	Q11	Concerning the map, choose the option that best represents your opinion. a) I find the TD payment MAP easy to read, follow and use to support decisions about TD payment. (Useful and Controllable). b) I find the TD payment MAP easy to read, follow and use to support decisions about TD non-payment. (Useful and Controllable). c) Assuming the payment map was available at my job, I predict that I would use it on a regular basis in the future.	Strongly agree Agree Neutral Disagree Strongly disagree	Closed

[3]<https://www.youtube.com/watch?v=OL5dCSPpXWM>
[4]<https://bit.ly/3uhoE4k>

7.3.2 Data Analyses

We matched the data collected in the complementary survey with the data collected in the InsighTD survey, allowing us to identify the demographic data of each participant. For closed questions and demographic data, we used descriptive statistics and calculated the share of participants choosing each option to understand the data better. For open-ended questions, we coded the answers to identify their central ideas. For example, consider the following answer to Q1: "I realize now that reasons for delay in paying of TD may be much more complex than it seems." We defined the code complexity of TD payment reasons. Two researchers performed the coding process. One of them coded all answers, and the other checked the extracted codes. The divergences were resolved in a consensus meeting.

7.3.3 Results

In total, we sent the survey to twenty-seven practitioners considering those we have contact with and could quickly participate in this research stage. We received eight answers[5], representing a ∼30% response rate.

7.3.3.1 Demographics. Most of the participants work in small (38%; organizations with up to 50 employees) and medium-sized (38%; 51 to 1000 employees) organizations, and two participants (25%) work in large (more than 1000 employees) organizations. Most of them identified themselves as developers (63%), but requirements analysts (25%) and software architects (13%) also answered the survey. Regarding the participants experience level, we have yet to receive answers from novices or beginners. Most of them are competent (50%), followed by proficient (25%) and expert (25%). The participants mainly adopted the agile software development process (50%), but others followed hybrid (25%) and traditional (25%) ones. These distributions are very close to the more extensive data set used in this dissertation (see Figure 3.2).

7.3.3.2 Perceptions on TD payment conceptual model. Only one participant indicated that his/her perception did not change due to the model (Q1). However, this participants response revealed their appreciation that the TD payment elements were "better defined in categories." The remaining participants indicated that their perception had changed as the model (i) introduced new concepts (three answers, e.g., "there some concepts related to payment and TD reasoning that I didn't know previously"), (ii) provided a more complete view on TD payment (e.g., "I now understand more completely about TD payment"), and (iii) revealed TD payment as complex (e.g., "I realize now that reasons for delay in paying of TD may be much more complex than it seems").

Most participants (five) reported learning more about TD payment by studying the model (Q2). Some quotes from the responses indicate new understandings of concepts, e.g., that the part of the model about "the definition of a favorable context for the payment of technical debt" added to the respondents knowledge. Others indicate a greater

[5]The raw data is available at <https://bit.ly/3QmiPes>.

appreciation of the importance of this knowledge for paying off TD efficiently (e.g., “I learned that understanding the reasons behind TD and defining the payment strategy in terms of activities, is indeed prerequisite for efficient TD payment,” or “I understood that the categorization strategy helps a lot in the payment and prioritization of TD”).

Only one participant disagreed with something in the model (Q3), specifically the lack of relationship between ExternalFactors and PaymentRelatedPracticeCategory, pointing out that “various external factors can put pressure on the payment of technical debt.” The other participants agreed with the model and indicated that it is well organized and described.

Lastly, the participants rated the model in terms of its accuracy (Q5) and completeness (Q6). In the participants opinion, the model seems accurate (answers ranging from 7 to 10) and complete (answers ranging from 8 to 10). For both, the mode was nine, with four answers.

7.3.3.3 Perceptions on TD payment map. Most of the participants (seven) indicated that it would influence their decisions about how to pay TD items (Q7). They explained (Q8) that the map can be used as a supporting tool for TD payment activities, e.g., “it gives some insights or even a checklist to support a strategy to avoid and solve technical debts,” and “it could be an initial guide for planning technical debt payment activities.” Also, one participant indicated that the percentages could contribute to comparing the TD payment initiatives of projects from the same organization. Only one participant claimed that the map would not influence their decisions on how to pay TD items and cautions that “it is hard to generalize this data... Percent are different from case to case.”

Most of the participants (six) affirmed that the map could influence decisions about when to pay TD items (Q9) and explained (Q10) that “when analyzing the map, the first thing I think about is: Paying technical debt and managing to reduce technical debts should be constant activities. In other words, the when is today, not to do everything at once, but to create the culture of starting to visualize the map and apply constant actions to make decisions about the technical debt and its payment” and “yes. It presents what should be improved and which phase you should tackle problems to avoid technical debts”. On the other hand, a participant reported that “most of time Im not a decision maker when and if that will be done. Anyhow, the map gives valuable insights to make such decision.”

Finally, in Q11, most of the participants indicated that the map is easy to read, follow, and use to support decisions about TD payment (strongly agree: four answers, agree: three answers, and neutral: one answer) and about TD non-payment (strongly agree: two answers, agree: five answers, and neutral: one answer). Also, five participants predicted that they would use the map regularly in the future if the map is available at their jobs (strongly agree: three answers, agree: two answers, and neutral: three answers).

7.3.4 Threats to Validity

As in any empirical study, this study has threats to validity (WOHLIN *et al.*, 2012). We attempted to remove them when possible and mitigate their effects when removal was not possible.

Construct validity. A threat arises from the questionnaire because the participants could misunderstand its questions. To reduce this threat, we performed three questionnaire validations conducted by researchers from the InsighTD project. The objective of the validation was to check the questions for clarity and completeness. Then, following the feedback we received, several adjustments were applied to the questionnaire and the survey procedures, such as including the video transcripts. Also, we piloted the questionnaire before its execution by inviting three participants through personal contacts with varying levels of industry experience. These participants were asked to tell us how much time it took to complete the task (the mean time was about 20 min), impressions about questions (e.g., clarity, ease of understanding, size), and improvement points. Finally, we performed one last review of the questionnaire.

Conclusion validity. A threat emerges from the coding process we performed to analyze the answers collected in the open-ended questions. As this process is subjective, it was conducted by two different researchers. One of them coded the answers, and the other checked the extracted codes. Inconsistences were solved in a meeting.

External validity. A threat arises from the fact that study participants were chosen by convenience because we invited only practitioners who answered the InsighTD survey. Also, due to the small sample, the results are not generalizable. To this end, further studies must be performed to evaluate the extended TD conceptual model and the proposed TD payment map.

7.4 ANSWERING RESEARCH QUESTION 4 - MODEL AND MAP

By including the TD prevention, monitoring, and payment concepts from the state of practice, we extended the TD conceptual model found in the technical literature. The new conceptual model reveals valuable information for researchers and practitioners. By analyzing the model, researchers and practitioners can understand the big picture related to TD and its entities and relationships. After extending the model, they can explore the payment, prevention, and prioritization activities and realize that integrating them can be performed by combining practices from these activities. The same occurred with monitoring, prevention, and payment activities. The model can drive new initiatives on this type of integration and support the definition or improvement of tools for TD management. Lastly, the model can help researchers characterize the context of the projects they are studying. Adding these new classes and relationships will help ensure that prevention, monitoring, and payment issues will be considered and described in detail in future studies.

Evaluation results from the conceptual model to TD payment indicate that it is well organized, accurate and complete, as well as provides valuable information to define strategies for TD payment. However, we cannot extrapolate these results to the full version of the model because the follow-up survey only focuses on the TD payment concept.

Regarding the conceptual maps, we defined a map for each TD management activity (prevention, monitoring, and payment). Each map comprises practices, and PARs are divided into categories. Also, each practice and PAR is accompanied by its types, nature, and percentage of the occurrence. Concerning the practices, the map can be useful for practitioners in two scenarios. If a team does not have experience performing TD prevention, monitoring, or payment, it can base its first steps on the experience of others and use the percentages as a criterion for choosing practices to set its TD management initiative. In the second scenario, if a team already has experience with TD prevention, monitoring, or payment, the map serves as a benchmarking tool. Based on the experience of others, the team can compare its practices and identify new ones that could be used. In addition, when looking at the categories level, if a team already uses a practice from a specific category, it can discover other practices related to that practice.

About the PARs, the map sheds light on possible improvement points in the teams capability to make the application of practices to feasible prevent, monitor, or pay off the debt. These points can be divided into two scenarios. First, let us consider a team with no experience in preventing, monitoring, or paying off TD items, the map can support the team in identifying PARs used in practice, and the percentages can be used as a criterion for verifying what PARs are more common for impeding TD management initiatives. Lastly, if the team has already experienced TD prevention, monitoring, or payment, the map can reveal new PARs from practitioners experience, improving the teams perception of factors that curb TD management. Also, the map categories can support the team in identifying other PARs related to ones already used by the team associated with the same TD prevention, monitoring, or payment issue. In both scenarios, as PARs are divided into impediments and decision factors, software teams can understand whether the TD non-prevention, non-monitoring, or non-payment occurs due to their decision or an impediment posed by other stakeholders.

Results from the TD payment map assessment reveal that the map seems useful as a support tool in TD payment activities, but it must be adapted according to practitioners context. Unfortunately, we cannot extrapolate these results to TD prevention and monitoring maps, requiring new investigations on those artifacts.

7.5 CONCLUDING REMARKS

This chapter presents the conceptual model for TD we extended by including the concepts of the state of TD prevention, monitoring, and payment practice and the conceptual maps that organize the set of practices and PARs into categories, types, and nature. We also performed a follow-up survey to assess the model and the map. Although the assessment only considered the version of the model on TD payment and the TD payment map, the results show that the TD payment conceptual model is well organized, accurate and complete, as well as provides valuable information to define strategies for TD payment. Also, the TD payment map seems useful as a support tool in TD payment activities, but it must be adapted according to practitioners context.

The following chapter presents the Impediments, decision factors, enabling practices, and actions (IDEA) diagrams and discusses how they can be used to analyze TD prevention, monitoring, and payment capabilities and issues. It also discusses the complementary studies we performed to assess the diagrams in terms of supporting TD management activities.

Chapter 8

This chapter introduces the IDEA (Impediments, Decision factors, Enabling practices, and Actions) diagrams that organize the set of practices and PARs and allow the analysis of capabilities and issues in TD management activities. Lastly, it offers and discusses the assessment of the diagrams we performed by two complementary empirical studies.

IDEA DIAGRAMS

Technical debt (TD) management comprises several activities, such as prevention, monitoring, and payment (LI; AVGERIOU; LIANG, 2015; RIOS; MENDONÇA; SPINOLA, 2018). By performing TD prevention, software teams can avoid potential TD items, while TD monitoring follows the identified TD items to measure their cost/benefits along with their elimination. This elimination is performed during the payment activity. Knowing the practices to prevent, monitor, and pay off debt items can support software teams in choosing the most appropriate practices in their context. On the other hand, and for varied reasons, teams sometimes avoid the application of these practices. Having information on these Practice avoidance reasons (PARs) can aid software teams in increasing their ability in TD management, revealing internal or external factors resulting in TD non-prevention, non-monitoring, and non-payment.

Related work has investigated TD prevention, monitoring, and payment practices and PARs (LI; AVGERIOU; LIANG, 2015; BOMFIM; SANTOS, 2017; APA *et al.*, 2020b; RIOS *et al.*, 2020; ARAGÃO *et al.*, 2022). For instance, Bomfim and Santos (2017) identified TD payment practices and PARs in the agile software development process. Rios *et al.* (2020) identified prevention and payment practices for managing documentation debt items, while Aragão *et al.* (2022) investigated prevention, monitoring, and payment practices for test debt items. Despite the valuable contributions of the current literature for the area, there is still a need for organizing the current body of practices and PARs into artifacts that can effectively be applied to support the management of TD in software projects. The previous chapter presented the conceptual maps for TD prevention, monitoring, and payment which organize the set of practices and PARs into categories, types, and nature. However, we want to provide an artifact to provide guidance on how to read and analyze the practices or PARs in isolation as well as in combination. In the absence of this guidance, development teams rely on textual information spread through several tables, thus hindering the use of current knowledge on TD management.

This chapter aims to answer **RQ4: How to organize the body of knowledge composed of prevention, monitoring, and payment practices - and Practice avoidance reasons (PARs) for TD non-prevention, non-monitoring, and non-payment - to support TD management?** For this, we organize the set of practices and PARs into Impediments, decision factors, enabling practices, and actions (IDEA) diagrams. Moreover, we assessed them to verify their support in TD management activities, supporting us to answer RQ4.1: How are the proposed artifacts characterized in terms of its support for TD management activities?

This chapter is based on the following papers: (i) Pitfalls and Solutions for Technical Debt Management in Agile Software Projects (FREIRE *et al.*, 2021c) and (ii) Assessing IDEA Diagrams for Supporting Analysis of Capabilities and Issues in Technical Debt Management (FREIRE *et al.*, 2022).

The main contributions of this chapter are:

- A set of IDEA diagrams that organize practices and PARs, supporting the analysis of capabilities and issues in TD management.
- Specializations of IDEA diagrams per process model and type of debt.
- Assessment of the IDEA diagrams by considering the point of view of students and software practitioners.

Section 8.1 presents the IDEA diagrams. Section 8.2 offers the empirical studies we performed to assess the diagrams. Section 8.3 discusses the results of the studies. Lastly, Section 8.4 presents the concluding remarks.

8.1 IDEA DIAGRAMS FOR TD PREVENTION, MONITORING, AND PAYMENT

IDEA diagrams organize issues decision factors and impediments and capabilities actions and enabling practices related to TD management into four quadrants. We design them inspired by the Strengths, weaknesses, opportunities, and threats (SWOT) analysis (SHAHIR; DANESHPAJOUH; RAMSIN, 2008). But unlike SWOT, the scope of the IDEA diagrams is not organizational planning but is to support software teams in increasing their ability to manage debt items (FREIRE *et al.*, 2021c). The diagrams can be defined for any TD management activity, and their practices and PARs can be specialized considering the types of debt (such as code, design, and requirements) and project context variables, such as the process model.

8.1.1 Diagrams Structure

Figure 8.1 presents the diagrams structure and how the quadrants are related to each other. Each quadrant is depicted by a specific color and contains a set of practices or PARs. On the left side of the diagram, practices are concentrated in the actions and enabling practices quadrants. The former groups actions for managing debt items, while the latter has the practices which support the TD management initiatives. On the right side, the diagram presents the PARs in the decision factors and impediments quadrants, representing the decisions made by the team itself or another external agent (i.e., a customer or organization), respectively.

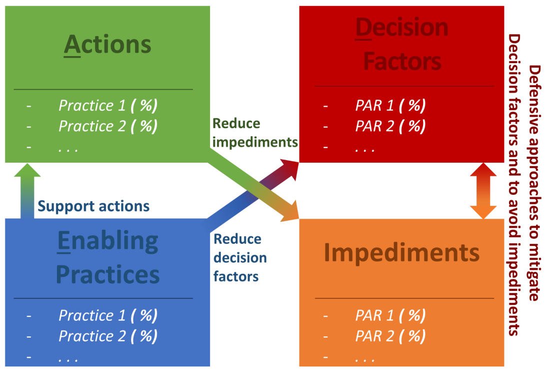


Figure 8.1 The IDEA diagrams structure

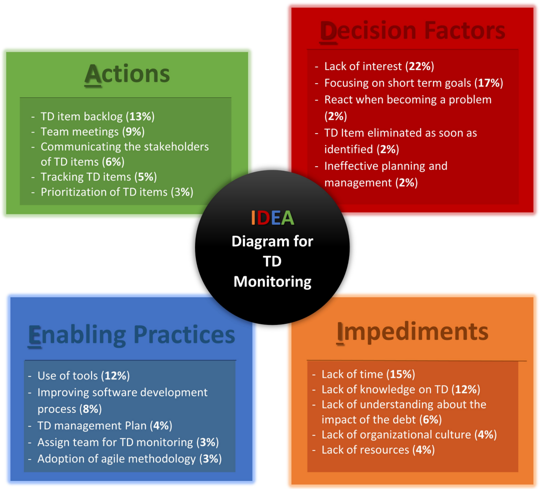

Actions
- TD item backlog (13%)
- Team meetings (9%)
- Communicating the stakeholders of TD items (6%)
- Tracking TD items (5%)
- Prioritization of TD items (3%)
Decision Factors
- Lack of interest (22%)
- Focusing on short term goals (17%)
- React when becoming a problem (2%)
- TD Item eliminated as soon as identified (2%)
- Ineffective planning and management (2%)
IDEA Diagram for TD Monitoring
Enabling Practices
- Use of tools (12%)
- Improving software development process (8%)
- TD management Plan (4%)
- Assign team for TD monitoring (3%)
- Adoption of agile methodology (3%)
Impediments
- Lack of time (15%)
- Lack of knowledge on TD (12%)
- Lack of understanding about the impact of the debt (6%)
- Lack of organizational culture (4%)
- Lack of resources (4%)

Figure 8.2 A summarized version of the IDEA diagram for TD monitoring

In all quadrants, the practices and PARs are ordered by a criterion that software teams can define. For example, a sorting criterion could be how frequently practices and PARs were used in the project. Although these percentages do not indicate if a practice or a PAR is critical, they can highlight the best-positioned practices and PARs in the predeterminate criterion.

8.1.2 Specializing the Diagram

We can populate the IDEA diagrams structure using a set of practices and PARs related to any TD management activities and specialize the diagrams according to any project context variable, such as the process model adopted by the team and type of debt. Specializations can be helpful to support more directed analyses of capabilities (enabling practices and actions) and issues (decision factors and impediments) by practitioners when defining their strategies to improve their TD management initiatives.

In this dissertation, we define IDEA diagrams for TD prevention, monitoring, and payment, using data from the InsighTD project to populate the structure and specialize it per process model and type of debt. Figure 8.2 depicts the IDEA diagram for TD monitoring. To make the diagram concise, the work considers only the five most frequently cited practices and PARs. The percentages with practices and PARs inform how often they were used in the InsighTD participants software projects. Appendix Figure F.1 shows the complete IDEA diagram for TD monitoring.

To specialize the IDEA diagrams per process model, we filtered the InsighTD data set considering the values of this project context: agile, hybrid, and traditional. As a result, we have the set of practices and PARs associated with each process model. Figure 8.3 shows the set of IDEA diagrams for (a) TD prevention, (b) TD monitoring, and (c) TD payment in the agile process with the five most cited elements per quadrant. Appendix Figure F.2 shows the complete complete IDEA diagrams for agile process. We followed a similar process to specialize the IDEA diagrams per TD type. However, we only considered design and documentation debt as values for TD type. Figure 8.4 depicts the IDEA diagram for design debt payment with the five most cited elements per quadrant. Appendix Figure F.4 and F.3 show the complete IDEA diagrams for design and documentation debt, respectively.

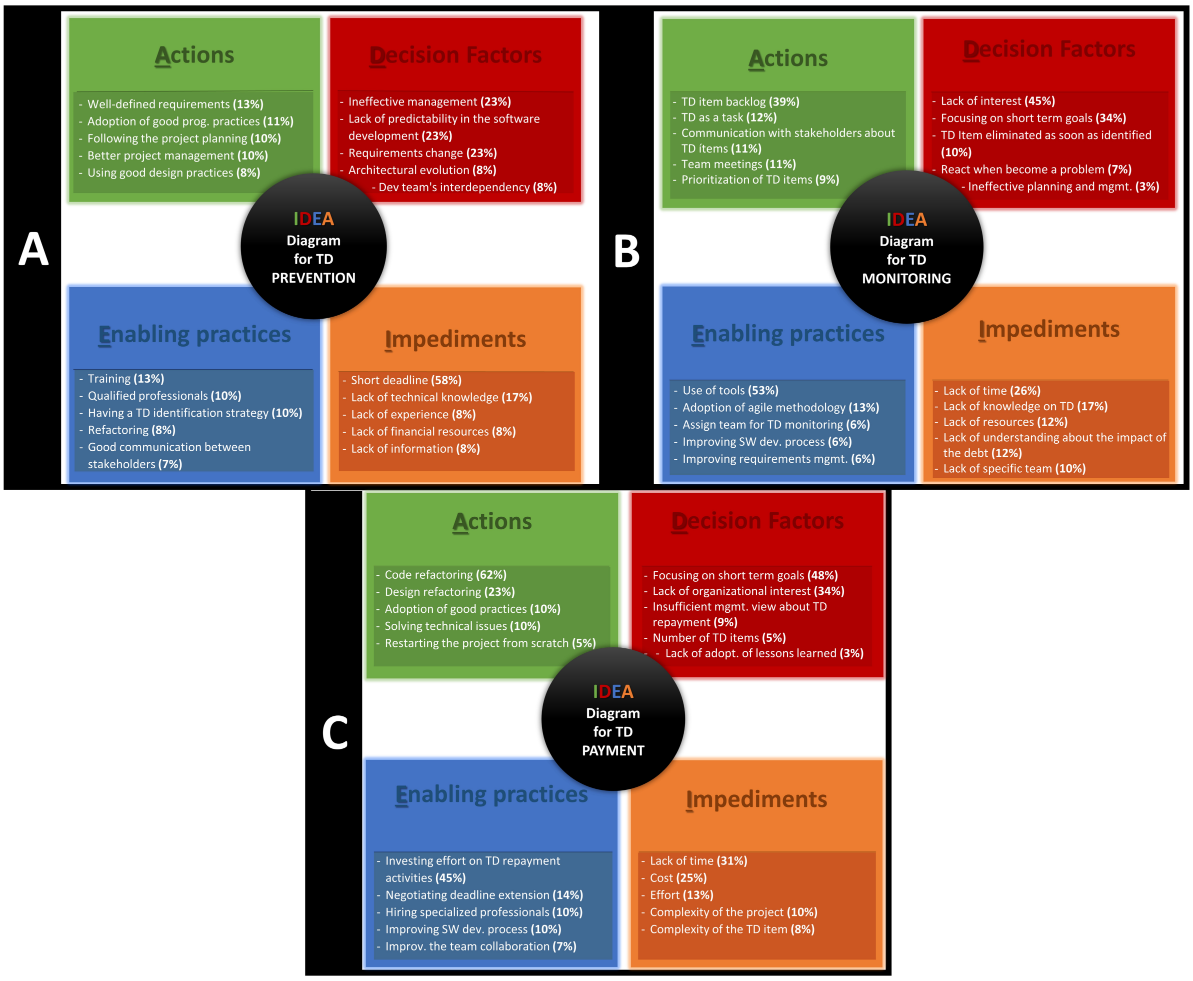

A
Actions
- Well-defined requirements (13%)
- Adoption of good prog. practices (11%)
- Following the project planning (10%)
- Better project management (10%)
- Using good design practices (8%)
Decision Factors
- Ineffective management (23%)
- Lack of predictability in the software development (23%)
- Requirements change (23%)
- Architectural evolution (8%)
- Dev team's interdependency (8%)
IDEA Diagram for TD PREVENTION
Enabling practices
- Training (13%)
- Qualified professionals (10%)
- Having a TD identification strategy (10%)
- Refactoring (8%)
- Good communication between stakeholders (7%)
Impediments
- Short deadline (58%)
- Lack of technical knowledge (17%)
- Lack of experience (8%)
- Lack of financial resources (8%)
- Lack of information (8%)
B
Actions
- TD item backlog (39%)
- TD as a task (12%)
- Communication with stakeholders about TD items (11%)
- Team meetings (11%)
- Prioritization of TD items (9%)
Decision Factors
- Lack of interest (45%)
- Focusing on short term goals (34%)
- TD Item eliminated as soon as identified (10%)
- React when become a problem (7%)
- Ineffective planning and mgmt. (3%)
IDEA Diagram for TD MONITORING
Enabling practices
- Use of tools (53%)
- Adoption of agile methodology (13%)
- Assign team for TD monitoring (6%)
- Improving SW dev. process (6%)
- Improving requirements mgmt. (6%)
Impediments
- Lack of time (26%)
- Lack of knowledge on TD (17%)
- Lack of resources (12%)
- Lack of understanding about the impact of the debt (12%)
- Lack of specific team (10%)
C
Actions
- Code refactoring (62%)
- Design refactoring (23%)
- Adoption of good practices (10%)
- Solving technical issues (10%)
- Restarting the project from scratch (5%)
Decision Factors
- Focusing on short term goals (48%)
- Lack of organizational interest (34%)
- Insufficient mgmt. view about TD repayment (9%)
- Number of TD items (5%)
- Lack of adopt. of lessons learned (3%)
IDEA Diagram for TD PAYMENT
Enabling practices
- Investing effort on TD repayment activities (45%)
- Negotiating deadline extension (14%)
- Hiring specialized professionals (10%)
- Improving SW dev. process (10%)
- Improv. the team collaboration (7%)
Impediments
- Lack of time (31%)
- Cost (25%)
- Effort (13%)
- Complexity of the project (10%)
- Complexity of the TD item (8%)

Figure 8.3 A summarized version of the IDEA diagram for TD prevention (a), monitoring (b), and payment (c) in agile processes

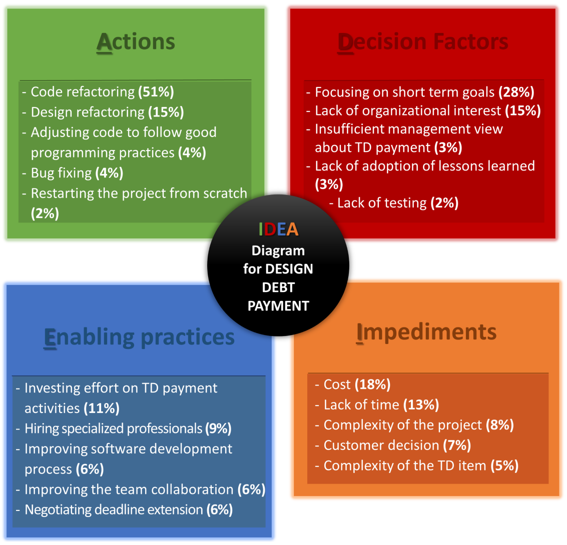

Actions
- Code refactoring (51%)
- Design refactoring (15%)
- Adjusting code to follow good programming practices (4%)
- Bug fixing (4%)
- Restarting the project from scratch (2%)
Decision Factors
- Focusing on short term goals (28%)
- Lack of organizational interest (15%)
- Insufficient management view about TD payment (3%)
- Lack of adoption of lessons learned (3%)
- Lack of testing (2%)
IDEA
Diagram for DESIGN DEBT PAYMENT
Enabling practices
- Investing effort on TD payment activities (11%)
- Hiring specialized professionals (9%)
- Improving software development process (6%)
- Improving the team collaboration (6%)
- Negotiating deadline extension (6%)
Impediments
- Cost (18%)
- Lack of time (13%)
- Complexity of the project (8%)
- Customer decision (7%)
- Complexity of the TD item (5%)

Figure 8.4 A summarized version of the IDEA diagram for design debt payment

8.1.3 Strategy of Use

IDEA diagrams can support the definition of TD management strategies by analyzing one or two quadrants simultaneously. When looking at isolated quadrants, software teams can identify the actions used to manage the debt (actions quadrants) and the practices that support these actions (enabling practices quadrants) shown on the left of the diagram. Further, software teams can identify the issues that hamper TD management from decisions made by the team (decision factors quadrant) or by an external factor (impediments quadrant).

Analyzing the relationships between quadrants can support software teams in boosting their TD management initiatives. Consider 8.4 as an example:

- **Actions and Enabling practices quadrants** can provide teams with a way to increase their TD management ability by suggesting other practices that could be implemented. For example, a software team that uses *code refactoring* and *design refactoring* actions to pay off design debt items can use *investing effort on TD payment activities* and *negotiating deadline extension* enabling practices to support these actions.

- **Decision factors and Impediments quadrants** can support teams in understanding why they are not managing TD. For example, a software team can identify that *focusing on short term goals* and *lack of testing* decision factors and *customer decision* impediment are the reasons for not paying off design debt items.

- **Enabling practices and Decision factors quadrants** can reduce weak areas related to TD management. For instance, if a team realizes that *lack of adoption of lessons learned* decision factor is the reason for design debt non-payment, the team can apply *improving software development process* and *improving the team collaboration* to change the teams mindset.

- **Actions and Impediments quadrants** can help teams to reduce the impediments for TD management. For example, if a team identifies that *complexity of the project* impediment hampers the payment of design debt, the team can apply *code refactoring*, *design refactoring*, and *adjusting the code to follow good programming practices* actions for reducing external factors in TD payment decisions.

To assess the IDEA diagrams in terms of their support of TD prevention, monitoring, and payment activities, we performed two empirical studies presented in the following section.

8.2 ASSESSING THE IDEA DIAGRAM

This section offers the complementary studies we performed to assess the IDEA diagrams in academic and industrial settings.

8.2.1 First Study - Assessing the ease of use, usefulness, and potential future use of the IDEA diagrams

The goal of this study is to **analyze** the IDEA diagrams **with the purpose of** characterizing them **with respect to** ease of use, usefulness, and potential future use **from the point of view of** undergraduate students enrolled in a software engineering course **in the context of** software development projects. As our intention is to investigate the perception on the use of a new technology (IDEA diagrams), we conducted the evaluation by applying the Technology acceptance model (TAM) (DAVIS, 1989). It captures the participants opinions on three constructs (perceived usefulness, ease to use, and self-predicted future use), measured by a set of questions. Our questions were adapted from those used by (RIOS *et al.*, 2019).

8.2.1.1 Project Context. The study consisted of analyzing the ease of use, usefulness, and potential future use of the IDEA diagrams through the simulation of TD management activities, whose objective was to identify, from a list of TD items, the prevention, monitoring, and payment practices and PARs that could be applied for the project. The list of debt items was extracted from an actual software project called National Transplantation System (NTS)[1].

The NTS is responsible for controlling and monitoring transplants of organs, tissues, and parts of the human body for therapeutic purposes in Brazil. The product consists of a medium-large database-driven web application. It includes several modules distributed through 212 use cases. The application was written in Java and based on the Model-View-Controller (MVC) framework. It contains 365K lines of code in 1377 domain classes. The project was developed with the following infrastructure: Eclipse IDE, Subversion,

[1]The National Transplantation System (NTS) was developed by a partner organization (the Fraunhofer Project Center at the Federal University of Bahia).

and Trac. The development team comprised one project manager, one technical leader, three requirements analysts, and eight developers. The project followed a Scrum-like development process to continuously integrate features and deliver working versions to the customer. The project team manually identified TD items and organized them into a spreadsheet, constituting the list of TD items used in this study.

8.2.1.2 Procedure and Instrumentation. Initially, the participants filled in a characterization and a consent form. Afterwards, the first author trained[2] the participants on TD and its concepts associated with the study, such as TD definition, design and documentation debt, and TD prevention, monitoring, and payment activities. An example of identifying practices and PARs related to those activities was also explained (**step 1**). As we wanted to reduce bias during the identification of TD management practices and PARs step of the study, we used an example in the context of house maintenance. For instance, a payment practice for *the kitchen sink* is *showing a slow flow of water could be using a plunger.*

In **step 2**, the participants, individually, analyzed in an *ad hoc* manner a design and a documentation debt item (see Table 8.1) to suggest practices and PARs associated with the TD prevention, monitoring, and payment of those items. We chose them because they were described in detail in the list of TD items our industry partner provided.

Table 8.1 TD items used in the study

Step	TD type	TD item description
2	Documentation	The allocation module does not have a requirements specification document.
2	Design	A verification with the name of the activity is necessary when it is required to identify a type of service or bill. This information is fixed in the code and can bring errors when some update is performed, or the data in the database has incorrect names.
4	Documentation	The documentation should be up-to-date, and requirements gathering should be conducted in accordance with the customer's needs. Frequent changes to these modules caused a lot of reworks.
4	Design	The invoice printing functionality needs to be simplified. The functionality is working correctly but needs to be adjusted in the future to be more adherent to the system design.

Step 3 focused on the training on the IDEA diagrams, explaining how to use them to support the analysis of practices and PARs associated with TD prevention, monitoring, and payment. We presented an example of how to analyze a debt item using the diagram, but also in the context of the house maintenance scenario.

In **step 4**, the participants received two new TD items, shown in Table 8.1, and analyzed them using the IDEA diagrams to suggest practices and PARs associated with the prevention, monitoring, and payment of those items. The participants received a set

[2]The training material is available at <http://bit.ly/3Zr5GGr>.

of IDEA diagrams for the types of analyzed debt items (documentation or design) for each TD management activity.

Lastly, the participants individually completed the evaluation form, containing a set of questions associated with the three constructs (perceived usefulness, ease to use, and self-predicted future use) considered in the TAM (**Step 5**). The evaluation form is available in Appendix G. To answer the questions in the form, the participants indicated the option that best represented their point of view on the IDEA diagrams, according to the following 5-point scale: (1) I totally agree; (2) I agree partially; (3) Neutral; (4) Partially disagree; and (5) Strongly disagree. At the end of the form, the participants also described the positive and negative aspects of the diagrams and suggestions for improvements and indicated whether the diagrams helped them to identify practices and PARs that they would not have placed without using them.

8.2.1.3 Data analysis. All answers were validated by following three criteria: (i) the participant filled in the consent and characterization forms, (ii) the participant performed the two activities of analysis of TD elements (steps 2 and 4), and (iii) the participant filled in the evaluation form.

For the closed questions, we used descriptive statistics and calculated the share of participants choosing each option to understand the data better. For open-ended questions, we applied a coding process to identify the central idea described in the answers (STRAUSS; CORBIN, 1998; SEAMAN, 1999). For example, a participant indicated the following positive aspect of the IDEA diagrams: "items properly separated and placed, easy to locate." As this answer is related to the diagram representation, we coded it as *adequate representation structure*. The coding process was performed by one researcher and revised by another. Divergences were resolved in a consensus meeting. In the end, we had a list of codes and their respective number of occurrences.

8.2.1.4 Results. This section provides the results obtained from the participants TAM study.

8.2.1.4.1 Characterization of the Participants. The participants were undergraduate students enrolled in a software engineering course. In total, 72 participants[3] completed all required steps. Half of them indicated some experience with software development (19% had at least one year of experience). Participants also indicated their level of experience in nine areas related to the software development process. We present the results in Table 8.2. We can notice that there are participants with experience in all areas of industry. Lastly, most of the participants have some level of knowledge on TD ranging from low (53%) to good (10%) and expert (3%).

8.2.1.4.2 Participants Point of View on IDEA Diagrams for TD Prevention, Monitoring, and Payment. Figures 8.5, 8.6, and 8.7 present the TAM statements for **perceived usefulness** constructs and the results for each of them for TD prevention, monitor-

[3]The raw data is available at <http://bit.ly/3jYjbg8>.

ing, and payment IDEA diagrams, respectively. Analyzing the statements in relation to usefulness (U1 to U8), most of the participants agreed with the affirmations for IDEA diagrams for TD prevention (more than 86% of the participants), TD monitoring (more than 81%), and TD payment (more than 89%). Thus, the following benefits are expected when using the diagrams: high productivity (U1 and U5), increased performance (U2, U4, U6, and U8), and efficacy (U3 and U7) when identifying practices and PARs related to TD prevention, monitoring, and payment.

Table 8.2 Level of Experience of Participants

Knowledge area	**Level of experience***				
	1	**2**	**3**	**4**	**5**
Project management	23	33	8	2	6
Monitoring and correction of software defects	29	27	4	4	8
Software maintenance	31	22	5	5	9
Software architecture	28	28	5	6	5
Software design	22	28	5	9	8
Software documentation	27	34	2	2	7
Requirement specification	20	40	7	1	4
Implementation	19	16	5	22	10
Software testing	22	28	6	8	8

* Levels of experience: (1) none, (2) studied in class, (3) practiced in classroom projects, (4) used in personal projects, and (5) used in projects in the industry.

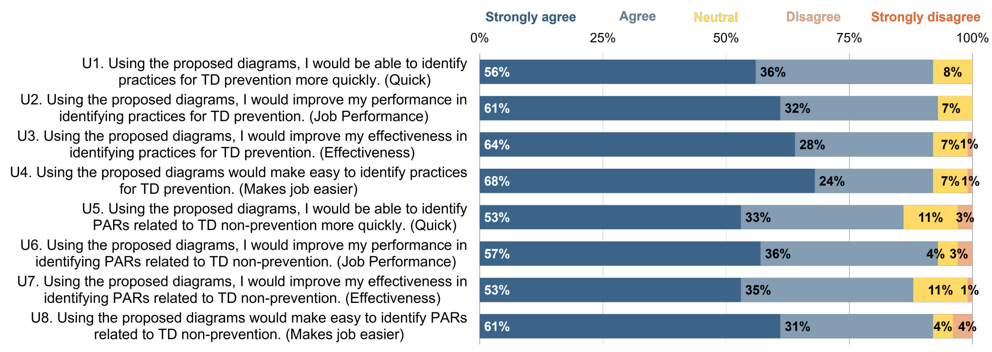


Figure 8.5 Perceived usefulness for IDEA Diagram for TD Prevention

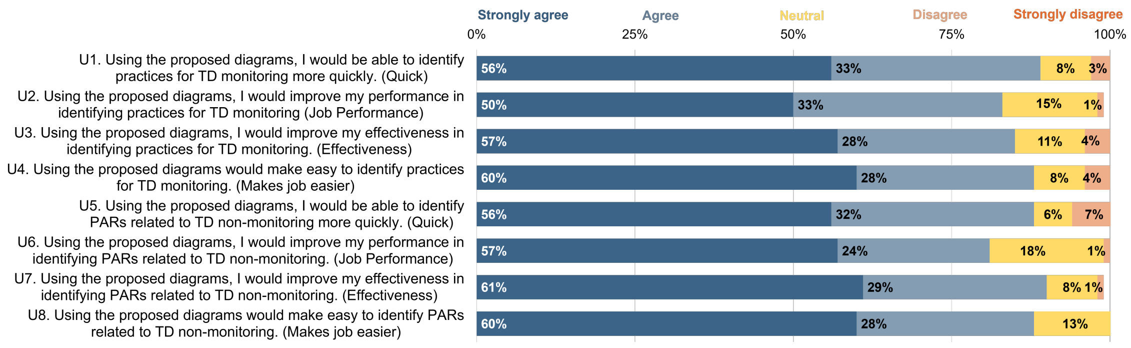


Figure 8.6 Perceived usefulness for IDEA Diagram for TD Monitoring

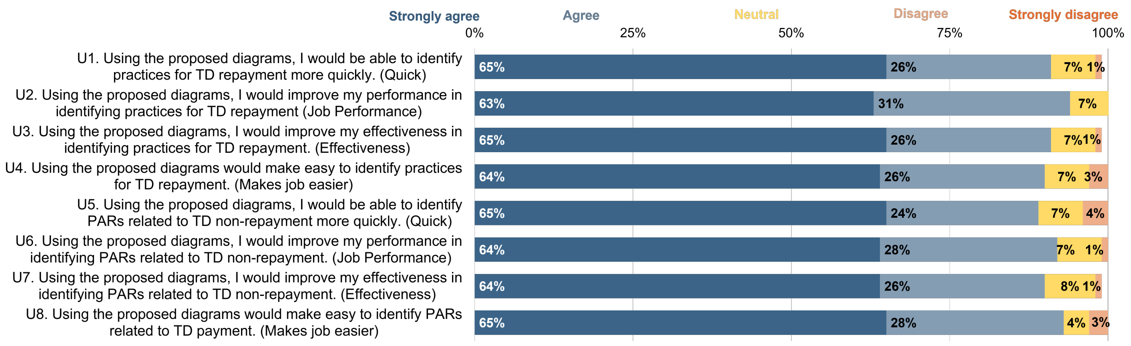


Figure 8.7 Perceived usefulness for IDEA Diagram for TD Payment

Moreover, 90% of the participants agreed with the following statements: "using the diagrams, I would increase my productivity in identifying practices and PARs" (strongly agree: 65%, agree: 25%, and neutral: 10%) and "I believe the proposed diagrams would be useful to support technical debt management" (strongly agree: 72%, agree: 18%, and neutral: 10%).

Figure 8.8 presents TAM statements for **easy-to-use** construct. At least 80% of the participants agreed with the statements associated with the benefits: easy to learn (E1), clear and understandable (E2), controllable (E3-E8), skillful (E9), remember (E10), and easy to use (E11).

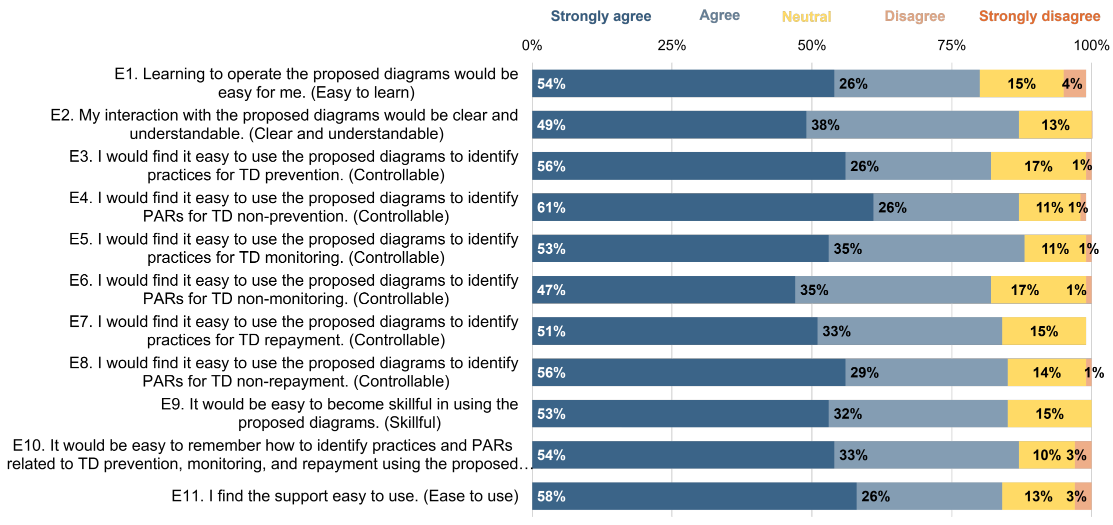


Figure 8.8 Easy-to-use for IDEA Diagrams for TD Prevention, Monitoring, and Payment

The participants also provided their opinion on the **self-predicted future use** construct. We found that (i) 92% of the participants agreed with "Assuming the proposed diagrams would be available to manage technical debt, I would use them in the future" (strongly agree: 63% and agree: 29%) and (ii) 63% of the participants agreed with "I would prefer to use the proposed diagrams to identify practices and PARs associated with TD prevention, monitoring and payment activities than in the usual way (without the diagrams supported). Only 15% of the participants disagreed with this statement (strongly agree: 45%, agree: 18%, neutral: 22%, disagree: 7%, and strongly disagree: 8%).

Lastly, most of the participants indicated that using the diagrams helped them identify (i) TD prevention, monitoring, or payment practices (93% of the participants) and (ii) PARs for TD non-prevention, non-monitoring, and non-payment (94% of the participants) that they would not have identified without the diagrams.

8.2.1.4.3 Positive Points Reported. The participants indicated that the IDEA diagrams allow an *easy identification of practices and PARs* (number of occurrences (NO) = 34), as described in the participants quote, “it is easier to identify reasons and practices for prevention, monitoring, and payment...” The participants also explained that the IDEA diagrams have an *adequate representation structure* (NO = 25), for instance, “detailed, items properly organized, easy to locate.” Lastly, participants argued that the diagrams *facilitate the decision making* (NO = 6; e.g., “better performance and effectiveness in decisions.”). Other mentioned positive points are *easy of use*, *provide a variety of practices and PARs*, and *facilitate the understanding.*

8.2.1.4.4 Negative Points Reported. Participants indicated that *it is possible that practitioners only consider the practices/PARs in the diagram* (NO = 10), as illustrated in “it can create a false impression that everything has been listed and cause a lack of interest in identifying other elements.” Also, the participants affirmed that *the diagrams have many practices or PARs* (NO = 5; e.g., “it contains a lot of information...”). Other negative points mentioned were *they do not present all practices and PARs* and *lack of dynamic manipulation of the diagram.*

8.2.1.4.5 Improvement Points Reported. The participants suggested the following improvement points: (i) *better organize information* (NO = 10; “better distribution of data in the space in each quadrant.”), (ii) *enable dynamic manipulation of the diagram* (NO = 4; “there could be some way to navigate through the diagrams...”), (iii) *enable the inclusion of new elements* (NO = 4; “Open a checkbox with the option to include new reasons.”), (iv) *simplify the diagram* (NO = 3; “use diagrams as simple as possible.”), (v) *remove redundant practices and PARs* (NO = 1; “some items can be merged as long as they look similar.”), and (vi) *better explain percentages* (NO = 1; “explain the percentages present in the diagrams.”).

8.2.2 Second Study - Perception of Software Practitioners

The goal of this study is to **analyze** the IDEA diagrams **with the purpose of** characterizing them **with respect to** their support to TD management activities **from the point of view of** software practitioners with experience in their roles **in the context of** software development projects.

8.2.2.1 Procedure and Instrumentation. We conducted semi-structured individual interviews[4] composed of three steps, as shown in Table 8.3. In the first step (**opening**), we presented the consent form and the concept of TD. Then, the participant answered questions on TD management, such as the level of experience with TD management and the strategies and tools used to manage the debt. In the second step (**perception about the IDEA diagrams**), we presented the IDEA diagrams and provided some examples of using the diagrams for supporting TD management. Then, we asked participants whether the diagrams (i) are easy to use and follow, (ii) could influence their decision about how to manage the debt, and (iii) could be used in their daily activities. In the last step (**closing**), we asked participants if they had anything more to say about the diagrams and asked them to fill in a characterization form.

Table 8.3 Interview script

Section	Question
Opening	We present the consent form. We present the concept of technical debt as defined by McConnell (2007). Q1. Concerning you experience with TD management: a) Never heard of it. b) I have heard of it, but I have not been on projects where I recognized TD. c) I have been on projects where I recognized TD but the project did not explicitly manage it. d) I have been on projects where we attempted to actively manage TD. Q1.1. Have you or your project ever used any strategies for managing TD items? Q1.2. What are these strategies? Can you explain how you have used them? Q1.3. Have you or your project ever used any tools for managing TD items? Q1.4. What are these tools? Can you explain how you have used them?
Perception of IDEA diagrams	We present the IDEA diagrams, explaining their structure and use for supporting TD management. Q2. Would you think the IDEA diagrams easy to read and follow to support decisions about TD management? Why? Q3. Would the IDEA diagrams influence any of your decisions about how to manage TD items? Why? Q4. Would you think percentages are useful to guide the choice of practices and PARs? Why? Q5. Assuming the IDEA diagrams were available at your job, would you use them for supporting your TD management activities? Why? Q5.1. Do you think IDEA diagrams fit your reality? Q5.2. What adjustments do you suggest?
Closing	Q6. Is there anything else you would like to add? Please, fill the characterization questionnaire.

We carried out the interviews remotely. Each of them lasted around 30 minutes and was recorded with the interviewees permission.

[4]The material used in the interview is available at <http://bit.ly/3WNGGY3>.

8.2.2.2 Data analysis. We transcribed the interviews and organized the answers by question. Then, we coded the transcriptions to identify the main idea presented in each answer (STRAUSS; CORBIN, 1998; SEAMAN, 1999). For example, a participant explained why IDEA diagrams can be used in daily activities: "you can communicate better with your team to avoid future problems." We coded it as *assist team communication.* This process was performed by the first author and revised by the last author. Divergences were resolved in a consensus meeting. Finally, we had a list of codes and their respective number of occurrences.

Concerning the characterization questions, we used descriptive statistics and calculated the share of participants choosing each characterization form option to understand the data better.

8.2.2.3 Results. This section presents the results obtained from the interviews.

8.2.2.3.1 Characterization of the Participants. We invited eleven practitioners[5]. Most of them work in medium-sized companies (organizations with 51 to 1000 employees; six participants), followed by large (more than 1000 employees; 4 participants) and small (up to 50 employees; one participant). The participants identified themselves as project manager or leader (3 participants), developer (2), product owner (2), process analyst (1), agile coach (1), tester (1), and software architect (1). Regarding the participants experience level, we interviewed five experts (authoritative knowledge of discipline and deep tacit understanding across area of practice), five proficient (depth of understanding of discipline and area of practice), and one competent (good working and background knowledge of area of practice). The participants mostly adopted agile software development (7), and the others followed hybrid methodologies.

Three participants do not have previous experience inn TD management, while eight of them have experience participating in projects in which they have identified TD items or have tried to manage them actively. The identified debt items are commonly registered in the product backlog or managerial tools.

8.2.2.3.2 Easy to read and follow. Most of the participants (nine participants) affirmed that the IDEA diagrams are easy to read and follow to support decisions on TD management because the diagrams: (i) *facilitate TD decision making* ("Because you can extract data from their topics (actions, impediments, decisions...). They are evident in aiding decision making"), (ii) *are succinct and clear* ("Yes, because I think they are very succinct and clear"), (iii) *can be understood by all stakeholders* ("I also clearly see how to use them in a very didactic way, even the product owner could understand"), (iv) *present in a summarized way both internal and external issues* ("I can see how I can get an X-ray of internal and external issues that are still not leading me to manage well debt items"), (v) *can be used in reviewing and planning meetings* ("...artifact to be considered at each review and backlog planning meeting"), and (vi) *facilitate TD items identification* ("It makes it easier to perceive the TD"). Lastly, three of these participants warned that the

[5]The raw data is available at <http://bit.ly/3VP8mKA>.

diagrams are easy to use but are not self-explanatory ("...Having the explanation is very useful to have visibility and put them into practice").

8.2.2.3.3 Influence decisions about how to manage the debt. Only one participant indicated that the diagram would not influence his/her decisions ("...in my case not so much. I have already implemented some of the practices you mentioned there"). The other participants reported that the diagrams could influence their decisions. They explained that the diagrams (i) *facilitate the communication between stakeholders* ("even with people who are not part of the team, you can take a picture of the situation and try to negotiate strategies to improve it."), (ii) *support the decision making on TD items* ("...from that diagram, make decisions about what would be relevant to do"), (iii) *support to identify problems* ("I would have clarity of the reasons that prevent me from managing them."), (iv) *have a customizable catalogue of practices used in the software industry* ("A catalog of best practices could be customized for each team."), and (v) *allow an effective risk management* ("As if it was an effective risk management, but for debt management. I can map impediments and internal factors and at the same time put together this action plan to improve management").

In addition, almost all participants (nine) indicated that the percentages were useful for choosing a practice or a PAR, highlighting that they *support the practices and PARs prioritization*, *present the most representative elements*, and *are based on previous experience.* Lastly, one participant was unsure about the usefulness percentage because it represents the consensus of other organizations, which not necessarily is related to her/his current context. However, the same participant indicated that the diagrams could be adapted to her/his context.

8.2.2.3.4 Can be used in daily activities. All participants indicated they could use the IDEA diagrams to support TD management activities. The participants explained that the diagrams (i) *enable continuous improvement of TD management actions* ("I see very clearly their use within a team, having a complete view of management and allowing us to set up a continuous improvement plan of actions to improve management"), (ii) *assist in tracking TD items*, (iii) *indicate possible problems and solutions to resolve them*, (iv) *help in identifying debt items and prioritizing them*, and (v) *assist team communication.*

Most of the participants (six) indicated that the diagrams could be adapted to their current context because they *would assist in negotiating project constraints* and *highlight the problems.* The participants also indicated the following necessary adjustments in the diagrams: (i) *remove practices that do not fit the developer's scope* ("I have a programmer's point of view. I am not on the manager side. I would cut some things out to make the set of actions more streamlined"), (ii) *include arrows between quadrants to indicate how the analysis should be done*, and (iii) *make it automated by suggesting relationships between quadrants.*

8.2.3 Threats to Validity

As in any empirical study, there are threats to validity in this study (WOHLIN *et al.*, 2012). We attempt to remove them when possible and mitigate their effects when removal is not possible.

8.2.3.1 First Study. *Construct validity.* A threat emerges from the material used to perform the TAM study because the TD items analyzed by the participants can influence their perceptions about the IDEA diagrams. Although we have used actual TD items provided by an industry partner, only the replication of our study with variation in the material can reduce this threat. Another threat arises from the TAM questionnaire due to its questions and length. The participants could misunderstand the questions, and the number of questions could fatigue participants. To mitigate this threat, we conducted two internal validations to identify improvements in the study structure and its material (questionnaire and training materials). We then piloted the questionnaire before its execution. None of these participants reported issues in answering the questionnaire.

Conclusion validity. The primary threat is that the participants were not allowed to participate in the software project that provided the TD items used in the study nor to talk to the project members. Therefore, it can affect the analysis of practices and PARs conducted by the participants. We assumed this threat as a limitation of the study. As we are not evaluating the final list of practices and PARs but the use of the diagrams, the participants are able to simulate the work of identifying TD management practices and PARs based on the description of each TD item and their experience with software development activities.

Another threat is related to the Hawthorne effect (PARNAS, 2003), which occurs when subjects know that they are studied, and affects the outcome of the empirical study. We mitigated this effect by clearly stating that study participation was entirely voluntary.

External validity. A threat arises from the fact that the study participants were chosen by convenience and were all students (some with industry experience in software development). To better investigate the possible effect of this threat, we divided the participants into two groups: without (levels 1-4 from Table 8.2) and with (level 5 from Table 8.2) experience in the software industry. Then, we compared the results from the two groups. In total, we have ten participants with experience ranging from two months to four years in software development activities. The other 62 participants do not have any experience. The students without experience have an average overall agreement of 90%, 9% are neutral, and only 2% reported some disagreement with the statements in the survey. The experienced participants presented the following results: 75%, 20%, and 5%, respectively. This analysis suggests that the level of experience has not influenced the study results, thus reducing the possible effects of this threat. This result is aligned with the previous findings of Falessi *et al.* (2018) that indicated that using students as participants remains a valid simplification of reality needed in laboratory contexts. Thus, although the results are not as generalizable as they could have been with a more representative sample, they provide initial evidence on the investigated topic.

8.2.3.2 Second Study. *Construct validity.* A threat emerges from the interview script in that the participants could misunderstand its questions. To mitigate this threat, we performed two internal validations and piloted the interviews with two participants with distinct levels of experience. Our goal was to identify the time necessary to run interviews (the mean time was about 30 min) and collect impressions about the questions and improvement points. All were considered acceptable during the validations and pilot.

Conclusion validity. The primary threat arises from our coding process to analyze the interview transcripts. As this process is subjective, one researcher coded the transcripts, and another reviewed the extracted codes. These researchers conducted a meeting to resolve eventual disagreements. Also, the Hawthorne effect (PARNAS, 2003) could affect our study. We clearly stated that study participation was entirely voluntary to support us in reducing this effect.

External validity. A threat arises from the small number of participants that may not be representative of a population. It did not allow us to perform more specific analysis, for example, if different practitioner roles who have different points of view on software projects, also have different perceptions of IDEA diagrams. We assumed this threat as the main limitation of this study. Another threat is related to the fact that study participants were chosen by convenience because we invited only practitioners from our network in the software industry. We decided to use this method because we had very little control over the availability of subjects, resulting in inviting only practitioners via existing contacts in software organizations. To mitigate this threat, we tried to ensure that our sample was reasonably representative and not strongly biased. For this, we tried to carefully select practitioners from distinct roles, with experience in their roles, and from different organizations.

8.3 ANSWERING RESEARCH QUESTION 4 - IDEA DIAGRAMS

The first and second studies provided positive evidence that the IDEA diagrams can be useful for supporting TD management activities. From the first study (TAM), 92% of the participants indicated that they could use the diagrams. Most of the participants also agreed that, by using them, they could see productivity gains in performance and effectiveness. Also, the diagrams were considered easy to learn and use. These results were confirmed through the second study (interviews). Most of the participants indicated that the diagrams are easy to read and follow, can influence decisions on how to manage debt items, and could be used to support their daily activities.

Software practitioners can use the IDEA diagrams to improve their ability to prevent, monitor, and pay off TD items. By identifying the actions and enabling practices, practitioners can define strategies for boosting their TD management activities, while having information on decision factors and impediments can support practitioners in defining strategies for reducing the internal and external factors that result in TD non-management. By analyzing the relationships between quadrants, diagrams can assist practitioners in defining these strategies.

The IDEA diagrams can be used by practitioners with or without experience managing TD items. For software teams who want to start managing TD, the ranked lists of practices and PARs organized in each of the IDEA diagrams provide valuable guidance on what to employ (practices) or curb (PARs) based on experience from other development teams. If a team already has experience managing TD, it can identify other commonly used practices or other PARs faced and can also identify enabling activities (enabling practices) that will improve the teams ability to manage TD. In other words, teams can create their own IDEA diagrams. This exercise is beneficial in and of itself but is also useful in comparing a team owned diagram to those of others, providing learning opportunities on TD management among teams or squads.

As a communication device, IDEA diagrams can be used in meetings to discuss TD items with all stakeholders, explaining the factors that lead to the non-management of TD and presenting workable solutions to minimize the effects of these factors. Additionally, software teams can use the diagrams to understand the risks associated with the accumulation of TD items and share them with all stakeholders with the aim of changing their mindset about the importance of TD management.

For researchers, our findings can guide new investigations on TD management, increasing the possibility of defining methods, tools, and artifacts closer to practitioners needs. Also, our findings can motivate new pieces of research in a problem-driven way, considering the IDEA diagrams as a starting point.

8.4 CONCLUDING REMARKS

This chapter presents the assessment of the IDEA diagrams, which provide support for TD prevention, monitoring, and payment activities. We conducted a study based on the TAM model with 72 students enrolled in a software engineering course and an interview-based study with 11 software practitioners. The results from both studies are very positive, complementary, and confirmatory, revealing that the data embedded into the IDEA diagrams and diagrams themselves can increase a teams capability to prevent, monitor, and pay off TD items.

From both studies, participants have pointed out some limitations (or difficulties) in using the IDEA diagrams:

- Some practices and PARs could not fit in all software project contexts. To minimize this limitation, development teams will be able to adapt the IDEA diagrams to specific contexts considering project variables such as process models, team size, and project age. This functionality is part of a plugin that we will develop to allow a dynamic interaction with the diagrams. These specializations can also reduce the number of practices and PARs in the diagrams.

- The manipulation of the diagrams is manual. To reduce this limitation, we aim to develop a plugin that will allow a dynamic interaction with the diagrams and, also, integrate them with well-known project management tools such as JIRA and Azure.

- The combined analysis of the quadrants is not self-explanatory. Training material, including videos, practical examples, and presentations, will be made available with the IDEA diagrams.

The following chapter discusses the final remarks of this dissertation.

Chapter

9

This chapter presents the concluding remarks of this Ph.D. dissertation, giving details on the work history, obtained findings, work limitations, and future work.

CONCLUDING REMARKS

This Ph.D. dissertation investigates the state of Technical debt (TD) prevention, monitoring, and payment practice, revealing the practices used to perform these activities and Practice avoidance reasons (PARs) that hinder their application. For this, we analyze answers given by software practitioners regarding these activities. These answers were collected by Brazilian, Chilean, Colombia, Costa Rican, Serbian, and the North American InsighTD replications. The dissertation also offers three artifacts: an updated version of the conceptual model for TD, resulting from the inclusion of the findings learned from the state of practice, a set of conceptual maps that organize the set of practices and PARs, and a set of Impediments, decision factors, enabling practices, and actions (IDEA) diagrams that allow the analysis of capabilities and issues for TD management. Empirical studies were conducted to assess the artifacts in terms of their support for TD management.

This chapter presents the concluding remarks of this dissertation. Section 9.1 presents the history of this work. Section 9.2 summarizes the main findings obtained in this work. Section 9.3 discusses the work limitations. Lastly, Section 9.4 offers perspectives for future work.

9.1 DISSERTATION HISTORY

Figure 9.1 presents the history of this work. Through it, one can identify the activities in time and the results obtained so far. Over four years, our work analyzed InsighTD data on TD prevention, monitoring, and payment activities. And in the last two years, we also have conducted complementary studies to assess the artifacts proposed in this dissertation.

The figure also presents complementary activities performed during the Ph.D. course, being characterized by the publication of papers on model smells, causes and effects of TD, and software requirements, by the collaboration in dissertations, thesis, or other scientific works conducted at the Federal University of Bahia (UFBA), Salvador University

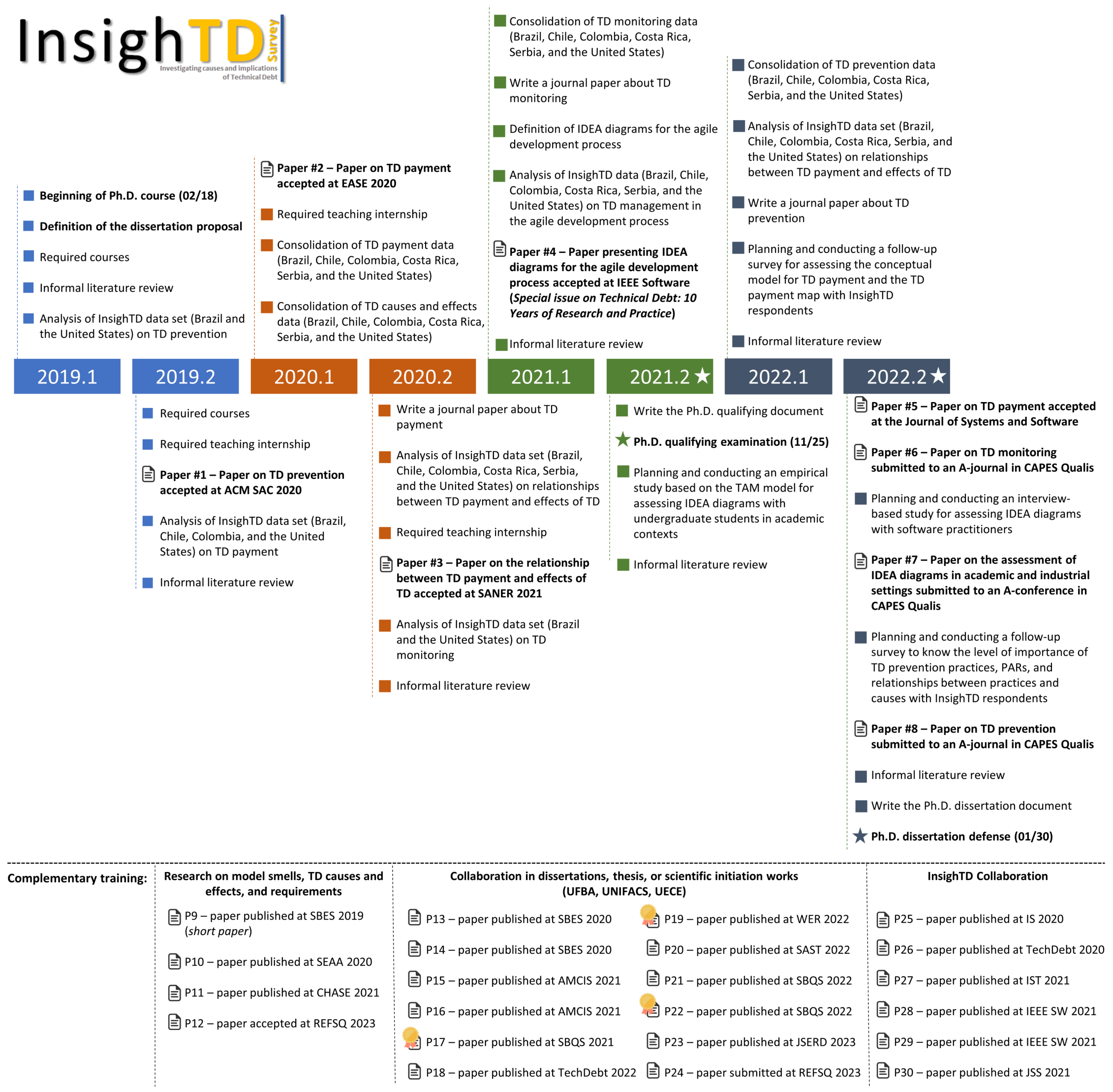

InsighTD Survey
Investigating causes and implications of Technical Debt
2019.1
Beginning of Ph.D. course (02/18)
Definition of the dissertation proposal
Required courses
Informal literature review
Analysis of InsighTD data set (Brazil and the United States) on TD prevention
2019.2
Required courses
Required teaching internship
Paper #1 – Paper on TD prevention accepted at ACM SAC 2020
Analysis of InsighTD data set (Brazil, Chile, Colombia, and the United States) on TD payment
Informal literature review
2020.1
Paper #2 – Paper on TD payment accepted at EASE 2020
Required teaching internship
Consolidation of TD payment data (Brazil, Chile, Colombia, Costa Rica, Serbia, and the United States)
Consolidation of TD causes and effects data (Brazil, Chile, Colombia, Costa Rica, Serbia, and the United States)
2020.2
Write a journal paper about TD payment
Analysis of InsighTD data set (Brazil, Chile, Colombia, Costa Rica, Serbia, and the United States) on relationships between TD payment and effects of TD
Required teaching internship
Paper #3 – Paper on the relationship between TD payment and effects of TD accepted at SANER 2021
Analysis of InsighTD data set (Brazil and the United States) on TD monitoring
Informal literature review
2021.1
Consolidation of TD monitoring data (Brazil, Chile, Colombia, Costa Rica, Serbia, and the United States)
Write a journal paper about TD monitoring
Definition of IDEA diagrams for the agile development process
Analysis of InsighTD data (Brazil, Chile, Colombia, Costa Rica, Serbia, and the United States) on TD management in the agile development process
Paper #4 – Paper presenting IDEA diagrams for the agile development process accepted at IEEE Software (Special issue on Technical Debt: 10 Years of Research and Practice)
Informal literature review
2021.2 ★
Write the Ph.D. qualifying document
★ Ph.D. qualifying examination (11/25)
Planning and conducting an empirical study based on the TAM model for assessing IDEA diagrams with undergraduate students in academic contexts
Informal literature review
2022.1
Consolidation of TD prevention data (Brazil, Chile, Colombia, Costa Rica, Serbia, and the United States)
Analysis of InsighTD data set (Brazil, Chile, Colombia, Costa Rica, Serbia, and the United States) on relationships between TD payment and effects of TD
Write a journal paper about TD prevention
Planning and conducting a follow-up survey for assessing the conceptual model for TD payment and the TD payment map with InsighTD respondents
Informal literature review
2022.2 ★
Paper #5 – Paper on TD payment accepted at the Journal of Systems and Software
Paper #6 – Paper on TD monitoring submitted to an A-journal in CAPES Qualis
Planning and conducting an interview-based study for assessing IDEA diagrams with software practitioners
Paper #7 – Paper on the assessment of IDEA diagrams in academic and industrial settings submitted to an A-conference in CAPES Qualis
Planning and conducting a follow-up survey to know the level of importance of TD prevention practices, PARs, and relationships between practices and causes with InsighTD respondents
Paper #8 – Paper on TD prevention submitted to an A-journal in CAPES Qualis
Informal literature review
Write the Ph.D. dissertation document
★ Ph.D. dissertation defense (01/30)
Complementary training:
Research on model smells, TD causes and effects, and requirements
P9 – paper published at SBES 2019 (short paper)
P10 – paper published at SEAA 2020
P11 – paper published at CHASE 2021
P12 – paper accepted at REFSQ 2023
Collaboration in dissertations, thesis, or scientific initiation works (UFBA, UNIFACS, UECE)
P13 – paper published at SBES 2020
P14 – paper published at SBES 2020
P15 – paper published at AMCIS 2021
P16 – paper published at AMCIS 2021
P17 – paper published at SBQS 2021
P18 – paper published at TechDebt 2022
P19 – paper published at WER 2022
P20 – paper published at SAST 2022
P21 – paper published at SBQS 2022
P22 – paper published at SBQS 2022
P23 – paper published at JSERD 2023
P24 – paper submitted at REFSQ 2023
InsighTD Collaboration
P25 – paper published at IS 2020
P26 – paper published at TechDebt 2020
P27 – paper published at IST 2021
P28 – paper published at IEEE SW 2021
P29 – paper published at IEEE SW 2021
P30 – paper published at JSS 2021

Figure 9.1 Ph.D. dissertation history

(UNIFACS) or State University of Ceará (UECE), and by the cooperation in the analysis and dissemination of different results obtained by the InsighTD replication teams.

Table 9.1 presents all publications achieved during the Ph.D. course, indicating the category, title, authorship, type, and reference. The category has the following values: primary (P), a publication that is in the scope of this dissertation; secondary (S), a publication that is not in the scope of this dissertation but is in the scope of the InsighTD project; and related (R), another kind of publication. The type of each publication can be conference-full paper (Cfp), conference-short paper (Csp), or journal (J).

Table 9.1: Publications achieved during the Ph.D. course

#	Cat.	Title	Authorship	Type	Reference
1	P	Actions and impediments for technical debt prevention: Results from a global family of industrial surveys	First author	Cfp	(FREIRE *et al.*, 2020c)
2	P	Surveying Software Practitioners on Technical Debt Payment Practices and Reasons for not Paying off Debt Items	First author	Cfp	(FREIRE *et al.*, 2020b)
3	P	How do Technical Debt Payment Practices Relate to the Effects of the Presence of Debt Items in Software Projects?	First author	Cfp	(FREIRE *et al.*, 2021a)
4	P	Pitfalls and Solutions for Technical Debt Management in Agile Software Projects	First author	J	(FREIRE *et al.*, 2021c)
5	P	Software Practitioners Point of View on Technical Debt Payment	First author	J	(FREIRE *et al.*, 2023)
6	P	Hearing the Voice of Software Practitioners on Technical Debt Monitoring: Understanding Monitoring Practices and the Practices' Avoidance Reasons	First author	J	(FREIRE *et al.*, 2022)
7	P	Assessing IDEA Diagrams for Supporting Analysis of Capabilities and Issues in Technical Debt Management	First author	Cfp	(FREIRE *et al.*, 2022)
8	P	A Comprehensive View on TD Prevention Practices and Reasons for not Prevent It	First author	J	(FREIRE *et al.*, 2023b)
9	S	On the Relationship Between Technical Debt Management and Process Models	Second author	J	(RIOS *et al.*, 2021)
10	S	Technical and Non-Technical Prioritization Schema for Technical Debt: Voice of TD-Experienced Practitioners	Fourth author	J	(MANDIĆ *et al.*, 2021)
11	S	Technical debt payment and prevention through the lenses of software architects	Fifth author	J	(PÉREZ *et al.*, 2021)
12	S	Prevalence, Common Causes and Effects of Technical Debt: Results from a Family of Surveys with the IT Industry	Fifth author	J	(RAMAČ *et al.*, 2022a)
13	S	What are the practices used by software practitioners on technical debt payment: Results from an international family of surveys	Fifth author	Cfp	(PÉREZ *et al.*, 2020)
14	S	Using Surveys to Build-up Empirical Evidence on Test-Related Technical Debt	Second author	Cfp	(SOUZA *et al.*, 2020)

(*table continues*)

Table 9.1: Publications achieved during the Ph.D. course (continued)

#	Cat.	Title	Authorship	Type	Reference
15	S	Technical Debt Payment Practices and Rationales Behind Payment Avoidance in the Serbian IT Industry	Third author	Cfp	(RAMAČ *et al.*, 2022b)
16	S	How Experience Impacts Practitioners Perception of Causes and Effects of Technical Debt	First author	Cfp	(FREIRE *et al.*, 2021b)
17	S	A Conceptual Framework to Support the Management of Technical Debt in Software Testing	Second author	Cfp	(ROCHA *et al.*, 2021)
18	S	Investigando a Relação entre a Prevenção da Dívida Técnica e as Atividades de Desenvolvimento de Software: Um Survey com Profissionais	Third author	Cfp	(BERENGUER *et al.*, 2021a)
19	S	Technical Debt is not Only about Code and We Need to be Aware about It	Third author	Cfp	(BERENGUER *et al.*, 2021b)
20	S	Organizing the TD Management Landscape for Requirements and Requirements Documentation Debt	Second author	Cfp	(BARBOSA *et al.*, 2022)
21	S	Evaluating a Conceptual Framework for Supporting Technical Debt Management in Testing Activities - A Feasibility Study	Second author	Cfp	(ROCHA *et al.*, 2022)
22	S	Investigating how Agile Software Practitioners Repay Technical Debt in Software Projects	Second author	Cfp	(SOARES *et al.*, 2022)
23	S	Investigating the Relationship between Technical Debt Management and Software Development Issues	Third author	J	(BERENGUER *et al.*, 2023)
24	R	Influence of Model Refactoring on Code Debt	First author	Csp	(FREIRE *et al.*, 2019)
25	R	On the Influence of UML Class Diagrams Refactoring on Code Debt: A Family of Replicated Empirical Studies	First author	Cfp	(FREIRE *et al.*, 2020a)
26	R	Using Stack Overflow to Assess Technical Debt Identification on Software Projects	Second author	Cfp	(GAMA *et al.*, 2020)
27	R	Requirements Engineering Issues Experienced by Software Practitioners: A Study on Stack Exchange	First author	Cfp	(FREIRE *et al.*, 2023a)
28	R	Investigating the point of view of project management practitioners on technical debt	Third author	Cfp	(GOMES *et al.*, 2022)
29	R	Technical Debt on Agile Projects: Managers point of view at Stack Exchange	Third author	Cfp	(SANTOS *et al.*, 2022)
30	R	Attributes of a Great Requirements Engineer	Second author	Cfp	(BARBOSA *et al.*, 2023)

Caption:

Cat - Category	P - Primary	S - Secondary	R - Related
Cfp - Conference (full paper)	Csp - Conference (short paper)		J - Journal

Looking at Table 9.1, one can notice that, until the writing of this dissertation, we published five papers that are direct results of this dissertation, being one in the IEEE Software (FREIRE *et al.*, 2021c), one in the Journal of Systems and Software (FREIRE *et al.*, 2023), and three in conferences in the area (FREIRE *et al.*, 2020c; FREIRE *et al.*, 2020b; FREIRE *et al.*, 2021a). In addition, we have three papers under review, one in a conference (FREIRE *et al.*, 2022) and two in a journal (FREIRE *et al.*, 2023b; FREIRE *et al.*, 2022).

Of the other twenty-two papers published during the period, fifteen are studies derived from this thesis. Of these, three papers were coordinated by the Serbian InsighTD replication team (RAMAČ *et al.*, 2022b; MANDIĆ *et al.*, 2021; RAMAČ *et al.*, 2022a) and two by the Colombian InsighTD replication team (PÉREZ *et al.*, 2020; PÉREZ *et al.*, 2021). The other seventeen papers are related to model smells, causes and effects of TD, or software requirements (lines 24, 25, and 27 at Table 9.1) or resulted from dissertations, theses, and scientific works related to software engineering TD (lines 9, 14, 16-23, 26, 28-30 at Table 9.1).

During this period, four awards were received:

- Distinguished Paper of the Brazilian Symposium on Software Quality (SBQS 2021), referring to Berenguer *et al.* (2021b).
- Best Paper of the Workshop on Requirements Engineering (WER 2022), referring to Barbosa *et al.* (2022).
- Distinguished Paper of the Brazilian Symposium on Software Quality (SBQS 2022), referring to Santos *et al.* (2022).
- Best Presentation in the Workshop de Estudantes de Pós-Graduação em Ciência da Computação do Programa de Pós-Graduação em Ciência da Computação (PGCOMP)-UFBA (WEPGCOMP 2022).

Lastly, the author of this dissertation acted as an informal co-adviser of the following former masters students:

- Lucinéia Batista de Souza, Computer Science. Master Thesis: *Organização de um conjunto de descobertas experimentais sobre dívida técnica relacionada ao teste.* Graduated at UFBA, 2021.
- Verusca Carin Barros Rocha, Computer Science. Master Thesis: *Evoluindo um mapa de apoio ao gerenciamento de itens de dívida técnica relacionada a teste de software.* Graduated at UFBA, 2022.
- Adriano Borges, Computer Science. Master Thesis: *Uma investigação sobre aspectos não técnicos relacionados à gestão da dívida técnica.* Graduated at UNIFACS, 2022.
- Gabriel Bahia Soares, Computer Science. Master Thesis: *Gerenciamento de dívida técnica em projetos de software ágeis.* Graduated at UNIFACS, 2022.

and has served as a reviewer for peer-review conferences (Industry Track of X Brazilian Conference on Software: Theory and Practice 2019; Encontro Nacional de Computação dos Institutos Federais - ENCompIF 2021, 2022) and journals (Software Quality Journal

- SQJ; Journal on Interactive Systems - JIS; International Journal of Agile Systems and Management - IJASM; *Revista Eletrônica de Iniciação Científica em Computação* - REIC).

9.2 SUMMARY OF FINDINGS OF THIS DISSERTATION

The main takeaways of this dissertation can be summarized in the following points:

- *Well-defined requirements*, *adoption of good programming practices*, and *better project management* were the most cited prevention-related practices for avoiding TD items in software projects.

- When practitioners talk about TD prevention, they report preventive actions as well as enabling TD prevention practices. Although enabling practices can be used to improve the capability of software teams in TD prevention, preventive actions are more cited by the surveys respondents.

- The prevention practices were organized into seven categories. The categories *planning and management* and *methodology* were the most cited. The first encompasses mainly enabling TD prevention practices, and the second primarily prevention actions.

- *Short deadlines* and *ineffective management* were the most cited PARs that could be used to explain the non-prevention of TD.

- The PARs explaining TD non-prevention can be a decision of the team or an impediment beyond the teams control. Both types (decision factor and impediment) were used by more than 50% of the projects.

- The PARs were organized into seven categories. The categories *planning and management*, *development issues*, and *methodology* were the most cited. While *planning and management* and *development issues* group decision factors and impediments, the methodology category only encompasses decision factors.

- TD is mainly monitored by tracking TD items in the backlog, by using specialized tools, or by discussing the TD item during team meetings.

- Direct monitoring actions, supported by enabling tools and practices, are the most common way of monitoring TD.

- Most TD monitoring practices represent dedicated management tasks or dedicated activities in an overall development process.

- The most pervasive TD monitoring practice is tracking TD items in a backlog. This practice can be used for monitoring 12 out of 14 TD types. The practices that follow are the use of specialized tools and code refactoring.

- Planning and management related practices are used for monitoring all types of debt except for requirements debt, versioning debt, and people debt.

- *Lack of interest*, *lack of time*, and the *focus on short term goals* are the main reasons why companies avoid monitoring TD items.

- TD is not monitored due to explicit decisions not to monitor the debt or due to impediments that obstruct teams or team members from monitoring.

- Most of the reasons why companies refuse to monitor TD items originate from management and the organization as an overall working context.

- Most TD types (13 out of 14) are not monitored for at least one of the following PARs: *lack of interest*, *lack of time*, or *focusing on short-term goals.*

- Only 40% of the TD items described in our data were paid off. Thus, paying off TD should not be the only focus of TD management.

- Not all TD payment activities result directly in the elimination of TD. Implementing practices for TD prevention, prioritization, and enabling TD payment is also necessary.

- Eliminating debt items cannot be solely a technical concern. Management practices are also necessary for implementing TD payment initiatives and making them part of the TD management strategy.

- Choosing practices from the categories *development issues*, *internal quality issues*, *methodology*, and *planning and management* can support software practitioners in addressing multiple types of debt.

- *Focusing on short-term goals* is not only a leading cause of incurring TD but also a primary reason for perpetuating it.

- In most cases, the PARs for TD non-payment have a managerial nature regardless of their type (decision factor or impediment), revealing that software practitioners must disseminate the importance of TD payment to their managers.

- Payment of all types of TD faces obstacles from the category *planning and management*, implying that involving project managers in TD payment initiatives is crucial.

- There is empirical evidence that identification, payment, prevention, and prioritization activities are related when software practitioners reduce TD items in their projects. The updated version of the conceptual model for TD highlights this relation.

- The conceptual model can help researchers characterize the context of the projects they are studying. Adding these new classes and relationships will help ensure that prevention, monitoring, and payment issues will be considered and described in detail in future studies.

- The conceptual maps can support identifying appropriate TD prevention, monitoring, and payment practices and the PARs for TD non-prevention, non-monitoring, and non-payment.

- IDEA diagrams can support software practitioners in defining strategies for increasing their capabilities and reducing their issues in managing debt items.

- Lastly, our findings can motivate new pieces of research in a problem-driven way, considering the artifacts (conceptual model, conceptual maps, and IDEA diagrams) as starting point. Developing new TD prevention, monitoring, and payment approaches could more closely reflect the practitioners needs.

9.3 DISSERTATION LIMITATIONS

As with any experimental work, our work has limitations derived from the need for more time and resources to execute more empirical studies and further analyze their data. Below we list some of those limitations:

- We used survey method to investigate the state of TD prevention, monitoring, and payment practice. To mitigate this methodological limitation, we performed other empirical studies to assess the artifacts proposed in this dissertation. However, it would be important to have more studies to validate the findings.

- This dissertation organizes the state of TD prevention, monitoring, and payment practice by considering the answers collected from the InsighTD project. Although this topic have been investigated in the technical literature, we did not include these findings in the artifacts proposed in this dissertation.

- Although the InsighTD project has twelve replication teams, this dissertation only uses data from six replications because they have been performed so far.

- There are other ways to analyze the InsighTD data set on TD prevention, monitoring, and payment, and this dissertation covered only some possibilities. At the time of writing this dissertation, there are submitted papers with new analyses derived from the InsighTD project. For example, Freire *et al.* (2023b) analyzed the relationship between TD prevention practices and effects of TD and the level of importance of these relationships, prevention practices, and PARs for TD prevention initiatives.

- Although initially assessed, the conceptual model updated by this dissertation must be thoroughly evaluated to contemplate TD prevention and monitoring concepts.

- This dissertation only empirically assessed the TD payment map. We still need to evaluate the TD prevention and monitoring maps.
- While we have assessed IDEA diagrams in academic and industrial settings, more industry case studies need to be conducted to enable software practitioners to use them in their daily activities.
- The manipulation of the IDEA diagrams is manual, requiring some effort to add or remove practices and PARs or update the percentages. We intend to automatize the diagrams to reduce this limitation.
- The IDEA diagrams can be specialized from distinct context variables, such as company size, process model, types of debt, level of experience of the team, and practitioner project role. Although InsighTD allows the identification of specified practices and PARs for these variables, this dissertation only specialized IDEA diagrams per process models and types of debt. The other specializations will be approached in future studies.

9.4 FUTURE WORK

From the discussion of the previous section, we can suggest the following prospective future work:

- **Conducting empirical studies to digest more the practices and PARs**. Some practices and PARs are in a high-level abstraction, limiting their application by software practitioners. Then, empirical studies can be conducted to collect more description about these practices and PARs, making them more feasible for use in practice. These studies can also reveal what practices or PARs are more critical for TD management.
- **Conducting systematic reviews to identify what practices were previously investigated**. By identifying the practices investigated in the technical literature, it is possible to verify how these practices were applied and what results were obtained, providing more information to support the choice of practices by software practitioners.
- **Correlations between practices and between PARs**. The InsighTD participants indicated, in some cases, more than one practice or PAR, allowing the investigation of practices or PARs commonly considered in combination. We have begun investigating this combination, considering TD payment practices and PARs for TD non-payment, as Freire *et al.* (2023) described. However, it is necessary a similar investigation for TD monitoring and payment.
- **Cooccurrence between causes of TD and TD prevention and monitoring, and between effects of TD and TD payment**. Investigating these cooccurrences can reveal how particular TD prevention or monitoring practices (or PARs

for TD non-prevention or non-monitoring) can be applied to particular causes of TD and how particular TD payment practices or PARs for TD non-payment can be employed to particular effects of TD. We have started investigating the cooccurrence of TD payment and effects (FREIRE *et al.*, 2021a) and TD prevention and causes (FREIRE *et al.*, 2023b). However, the cooccurrence of TD monitoring (practices and PARs) and causes of TD still need to be investigated.

- **Planning and performing empirical studies to assess the conceptual model for TD and conceptual maps**. The complete conceptual model for TD and TD prevention and monitoring maps can be assessed to capture the perception of software practitioners on the accuracy and completeness of the model and maps.
- **Specializations of IDEA diagrams**. The diagrams can be specialized considering different context variables. These specializations can support software practitioners in identifying practices and PARs that fit their context.
- **Planning and performing case studies to assess IDEA diagrams in industrial settings**. Software practitioners can use the IDEA diagrams in their daily project activities to support them in defining strategies to deal with TD items. Then, we can investigate where the diagrams would fit in the development process followed by the team and whether the diagrams can increase the software teams ability to prevent, monitor, and pay off TD items.
- **Automatization of the IDEA diagrams**. To develop a plugin that will allow dynamic interaction with the diagrams and also integrate them with well-known project management tools such as JIRA and Azure.
- **Definition of a strategy for TD management**. It is possible to investigate how conceptual maps and IDEA diagrams can comprise a strategy to support TD management, considering how software practitioners can identify the practices and PARs used/presented in their projects.

BIBLIOGRAPHY

ABAD, Z. S. H.; KARIMPOUR, R.; HO, J.; DIDAR-AL-ALAM, S.; RUHE, G.; TSE, E.; BARABASH, K.; HARGREAVES, I. Understanding the impact of technical debt in coding and testing: An exploratory case study. In: *2016 IEEE/ACM 3rd International Workshop on Software Engineering Research and Industrial Practice (SER&IP)*. [S.l.: s.n.], 2016. p. 25–31.

ALBUQUERQUE, D.; GUIMARAES, E. T.; TONIN, G. S.; PERKUSICH, M. B.; ALMEIDA, H.; PERKUSICH, A. Perceptions of technical debt and its management activities - a survey of software practitioners. In: *Proceedings of the XXXVI Brazilian Symposium on Software Engineering*. New York, NY, USA: Association for Computing Machinery, 2022. (SBES '22), p. 220229. ISBN 9781450397353. Available at: <https://doi.org/10.1145/3555228.3555237>.

ALVES, N. S.; MENDES, T. S.; MENDONÇA, M. G. de; SPINOLA, R. O.; SHULL, F.; SEAMAN, C. Identification and management of technical debt. *Inf. Softw. Technol.*, Butterworth-Heinemann, USA, v. 70, n. C, p. 100121, feb 2016. ISSN 0950-5849. Available at: <https://doi.org/10.1016/j.infsof.2015.10.008>.

ALVES, N. S. R. *Organização de um conjunto de descobertas experimentais sobre causas e efeitos da dívida técnica através de uma família de surveys globalmente distribuída*. Tese (Doutorado) — Universidade Federal da Bahia, 2020.

ALZAGHOUL, E.; BAHSOON, R. Evaluating technical debt in cloud-based architectures using real options. In: *2014 23rd Australian Software Engineering Conference*. [S.l.: s.n.], 2014. p. 1–10.

AMPATZOGLOU, A.; AMPATZOGLOU, A.; CHATZIGEORGIOU, A.; AVGERIOU, P. The financial aspect of managing technical debt. *Inf. Softw. Technol.*, Butterworth-Heinemann, USA, v. 64, n. C, p. 5273, aug 2015. ISSN 0950-5849. Available at: <https://doi.org/10.1016/j.infsof.2015.04.001>.

ANDRADE, C. Internal, external, and ecological validity in research design, conduct, and evaluation. *Indian J Psychol Med*, United States, v. 40, n. 5, p. 498–499, set. 2018.

APA, C.; JERONIMO, H.; NASCIMENTO, L. M.; VALLESPIR, D.; TRAVASSOS, G. H. The perception and management of technical debt in software startups. In: ______. *Fundamentals of Software Startups: Essential Engineering and Business Aspects*. Cham: Springer International Publishing, 2020. p. 61–78. ISBN 978-3-030-35983-6. Available at: <https://doi.org/10.1007/978-3-030-35983-6_4>.

APA, C.; SOLARI, M.; VALLESPIR, D.; TRAVASSOS, G. H. A taste of the software industry perception of technical debt and its management in uruguay: A survey in software industry. In: *Proceedings of the 14th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. New York, NY, USA: Association for Computing Machinery, 2020. (ESEM '20). ISBN 9781450375801. Available at: <https://doi.org/10.1145/3382494.3421463>.

ARAGÃO, B. S.; ANDRADE, R. M. C.; SANTOS, I. S.; CASTRO, R. N. S.; LELLI, V.; DARIN, T. G. R. Testdcat 3.0: Catalog of test debt subtypes and management activities. *Software Quality Journal*, Kluwer Academic Publishers, USA, v. 30, n. 1, p. 181225, mar 2022. ISSN 0963-9314. Available at: <https://doi.org/10.1007/s11219-020-09533-y>.

AVGERIOU, P.; KRUCHTEN, P.; OZKAYA, I.; SEAMAN, C. Managing Technical Debt in Software Engineering (Dagstuhl Seminar 16162). *Dagstuhl Reports*, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, v. 6, n. 4, p. 110–138, 2016. ISSN 2192-5283. Available at: <http://drops.dagstuhl.de/opus/volltexte/2016/6693>.

BARBOSA, L.; FREIRE, S.; P., R. S.; MENDONÇA, M.; KALINOWSKI, M.; CODABUX, Z.; SPÍNOLA, R. Attributes of a great requirements engineer. In: *The 29th International Working Conference on Requirement Engineering: Foundation for Software Quality (REFSQ*. [S.l.: s.n.], 2023. Submitted.

BARBOSA, L.; FREIRE, S.; RIOS, N.; RAMAČ, R.; TAUŠAN, N.; PÉREZ, B.; CASTELLANOS, C.; CORREAL, D.; PACHECO, A.; LÓPEZ, G.; MANDIĆ, V.; MACIEL, R. S. P.; MENDONÇA, M.; FALESSI, D.; IZURIETA, C.; SEAMAN, C.; SPÍNOLA, R. Organizing the td management landscape for requirements and requirements documentation debt. In: *25th Workshop on Requirements Engineering (WER)*. [s.n.], 2022. Available at: <http://wer.inf.puc-rio.br/WERpapers/artigos/artigos_WER22/WER_2022_Camera_ready_paper_28.pdf>.

BASILI, V. R. The experimental paradigm in software engineering. In: ROMBACH, H. D.; BASILI, V. R.; SELBY, R. W. (Ed.). *Experimental Software Engineering Issues: Critical Assessment and Future Directions*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1993. p. 1–12. ISBN 978-3-540-47903-1.

BASILI, V. R.; SELBY, R. W.; HUTCHENS, D. H. Experimentation in software engineering. *IEEE Trans. Softw. Eng.*, IEEE Press, v. 12, n. 7, p. 733743, jul 1986. ISSN 0098-5589.

BEHUTIYE, W. N.; RODRÍGUEZ, P.; OIVO, M.; MISIRLI, A. T. Analyzing the concept of technical debt in the context of agile software development: A systematic literature review. *Inf. Softw. Technol.*, v. 82, p. 139–158, 2017.

BERENGUER, C.; BORGES, A.; FREIRE, S.; RIOS, N.; SPINOLA, R. O. Investigando a relação entre a prevenção da dívida técnica e as atividades de desenvolvimento de software: Um survey com profissionais. In: *27th annual Americas Conference on Information*

Systems (AMCIS). [s.n.], 2021. Available at: <https://aisel.aisnet.org/amcis2021/lacais/lacais/4>.

BERENGUER, C.; BORGES, A.; FREIRE, S.; RIOS, N.; TAUŠAN, N.; RAMAČ, R.; PÉREZ, B.; CASTELLANOS, C.; CORREAL, D.; PACHECO, A.; LÓPEZ, G.; FALESSI, D.; SEAMAN, C.; MANDIĆ, V.; IZURIETA, C.; SPÍNOLA, R. Technical debt is not only about code and we need to be aware about it. In: *XX Brazilian Symposium on Software Quality*. New York, NY, USA: Association for Computing Machinery, 2021. (SBQS '21). ISBN 9781450395533. Available at: <https://doi.org/10.1145/3493244.3493285>.

BERENGUER, C.; BORGES, A.; FREIRE, S.; RIOS, N.; RAMAČ, R.; TAUŠAN, N.; PÉREZ, B.; CASTELLANOS, C.; CORREAL, D.; PACHECO, A.; LÓPEZ, G.; MENDONÇA, M.; FALESSI, D.; SEAMAN, C.; MANDIĆ, V.; IZURIETA, C.; SPÍNOLA, R. Investigating the relationship between technical debt management and software development issues. *Journal of Software Engineering Research and Development*, v. 11, n. 1, p. 3:1 3:21, Feb. 2023. Available at: <https://sol.sbc.org.br/journals/index.php/jserd/article/view/2581>.

BOGNER, J.; VERDECCHIA, R.; GEROSTATHOPOULOS, I. Characterizing technical debt and antipatterns in ai-based systems: A systematic mapping study. In: *2021 IEEE/ACM International Conference on Technical Debt (TechDebt)*. [S.l.: s.n.], 2021. p. 64–73.

BOMFIM, M. M.; SANTOS, V. A. Strategies for reducing technical debt in agile teams. In: SILVA, T. Silva da; ESTÁCIO, B.; KROLL, J.; FONTANA, R. M. (Ed.). *Agile Methods*. Cham: Springer International Publishing, 2017. p. 60–71. ISBN 978-3-319-55907-0.

BONFIM, V. D.; BENITTI, F. B. V. Requirements debt: causes, consequences, and mitigating practices. In: PENG, R.; PANTOJA, C. E.; KAMTHAN, P. (Ed.). *The 34th International Conference on Software Engineering and Knowledge Engineering, SEKE 2022, KSIR Virtual Conference Center, USA, July 1 - July 10, 2022*. KSI Research Inc., 2022. p. 13–18. Available at: <https://doi.org/10.18293/SEKE2022-114>.

BROWN, N.; CAI, Y.; GUO, Y.; KAZMAN, R.; KIM, M.; KRUCHTEN, P.; LIM, E.; MACCORMACK, A.; NORD, R.; OZKAYA, I.; SANGWAN, R.; SEAMAN, C.; SULLIVAN, K.; ZAZWORKA, N. Managing technical debt in software-reliant systems. In: *Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research*. New York, NY, USA: Association for Computing Machinery, 2010. (FoSER '10), p. 4752. ISBN 9781450304276. Available at: <https://doi.org/10.1145/1882362.1882373>.

CARTAXO, B.; PINTO, G.; VIEIRA, E.; SOARES, S. Evidence briefings: Towards a medium to transfer knowledge from systematic reviews to practitioners. In: *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. New York, NY, USA: Association for Computing Machinery, 2016. (ESEM '16). ISBN 9781450344272. Available at: <https://doi.org/10.1145/2961111.2962603>.

CHARALAMPIDOU, S.; AMPATZOGLOU, A.; CHATZIGEORGIOU, A.; TSIRIDIS, N. Integrating traceability within the ide to prevent requirements documentation debt. In: *2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. [S.l.: s.n.], 2018. p. 421–428.

CODABUX, Z.; WILLIAMS, B. J.; NIU, N. A quality assurance approach to technical debt. In: *In Proceedings of the International Conference on Software Engineering Research and Practice (SERP)*. [S.l.]: The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2014.

CUNNINGHAM, W. The wycash portfolio management system. In: *Addendum to the Proceedings on Object-Oriented Programming Systems, Languages, and Applications (Addendum)*. New York, NY, USA: Association for Computing Machinery, 1992. (OOPSLA '92), p. 2930. ISBN 0897916107. Available at: <https://doi.org/10.1145/157709.157715>.

DAS, D.; MARUF, A. A.; ISLAM, R.; LAMBARIA, N.; KIM, S.; ABDELFATTAH, A. S.; CERNY, T.; FRAJTAK, K.; BURES, M.; TISNOVSKY, P. Technical debt resulting from architectural degradation and code smells: A systematic mapping study. *SIGAPP Appl. Comput. Rev.*, Association for Computing Machinery, New York, NY, USA, v. 21, n. 4, p. 2036, jan 2022. ISSN 1559-6915. Available at: <https://doi.org/10.1145/3512753.3512755>.

DAVIS, F. D. Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS Quarterly*, Management Information Systems Research Center, University of Minnesota, v. 13, n. 3, p. 319–340, 1989. ISSN 02767783. Available at: <http://www.jstor.org/stable/249008>.

DAVIS, N. Driving quality improvement and reducing technical debt with the definition of done. In: *2013 Agile Conference*. [S.l.: s.n.], 2013. p. 164–168.

de Toledo, S. S.; MARTINI, A.; SJøBERG, D. I. Identifying architectural technical debt, principal, and interest in microservices: A multiple-case study. *Journal of Systems and Software*, v. 177, p. 110968, 2021. ISSN 0164-1212. Available at: <https://www.sciencedirect.com/science/article/pii/S0164121221000650>.

ERNST, N.; DELANGE, J.; KAZMAN, R. *Technical debt in practice how to find it and fix it*. [S.l.]: The MIT Press, 2021.

ERNST, N. A.; BELLOMO, S.; OZKAYA, I.; NORD, R. L.; GORTON, I. Measure it? manage it? ignore it? software practitioners and technical debt. In: *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*. New York, NY, USA: Association for Computing Machinery, 2015. (ESEC/FSE 2015), p. 5060. ISBN 9781450336758. Available at: <https://doi.org/10.1145/2786805.2786848>.

FALESSI, D.; JURISTO, N.; WOHLIN, C.; TURHAN, B.; MÜNCH, J.; JEDLITSCHKA, A.; OIVO, M. Empirical software engineering experts on the use of students and professionals in experiments. *Empirical Software Engineering*, v. 23, n. 1, p. 452–489, Feb 2018. ISSN 1573-7616. Available at: <https://doi.org/10.1007/s10664-017-9523-3>.

FALESSI, D.; KAZMAN, R. Worst smells and their worst reasons. In: *2021 IEEE/ACM International Conference on Technical Debt (TechDebt)*. [S.l.: s.n.], 2021. p. 45–54.

FERNÁNDEZ, D. M.; WAGNER, S.; KALINOWSKI, M.; FELDERER, M.; MAFRA, P.; VETRÒ, A.; CONTE, T.; CHRISTIANSSON, M.-T.; GREER, D.; LASSENIUS, C.; MÄNNISTÖ, T.; NAYABI, M.; OIVO, M.; PENZENSTADLER, B.; PFAHL, D.; PRIKLADNICKI, R.; RUHE, G.; SCHEKELMANN, A.; SEN, S.; SPINOLA, R.; TUZCU, A.; VARA, J. L. de la; WIERINGA, R. Naming the pain in requirements engineering. *Empirical Software Engineering*, v. 22, n. 5, p. 2298–2338, Oct 2017. ISSN 1573-7616. Available at: <https://doi.org/10.1007/s10664-016-9451-7>.

FREIRE, E. S. S.; PASSOS, A. F. de O.; SANT'ANNA, C.; SPÍNOLA, R. O.; NETO, M. G. de M. Influence of model refactoring on code debt: A replicated study. In: *Proceedings of the XXXIII Brazilian Symposium on Software Engineering*. New York, NY, USA: Association for Computing Machinery, 2019. (SBES '19), p. 452456. ISBN 9781450376518. Available at: <https://doi.org/10.1145/3350768.3350793>.

FREIRE, S.; GOMES, F.; BARBOSA, L.; MENDES, T.; REGES, G.; MACIEL, R. S. P.; MENDONÇA, M.; SPÍNOLA, R. Requirements engineering issues experienced by software practitioners: A study on stack exchange. In: *The 29th International Working Conference on Requirement Engineering: Foundation for Software Quality (REFSQ*. [S.l.: s.n.], 2023.

FREIRE, S.; PACHECO, A.; RIOS, N.; PÉREZ, B.; CASTELLANOS, C.; CORREAL, D.; RAMAČ, R.; MANDIĆ, V.; TAUŠAN, N.; LÓPEZ, G.; MENDONÇA, M.; FALESSI, D.; IZURIETA, C.; SEAMAN, C.; SPINOLA, R. A comprehensive view on td prevention practices and reasons for not prevent it. *submitted*, 2023.

FREIRE, S.; PASSOS, A.; MENDONÇA, M.; SANTANNA, C.; SPÍNOLA, R. O. On the influence of uml class diagrams refactoring on code debt: A family of replicated empirical studies. In: *2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. [S.l.: s.n.], 2020. p. 346–353.

FREIRE, S.; RIOS, N.; GUTIERREZ, B.; TORRES, D.; MENDONÇA, M.; IZURIETA, C.; SEAMAN, C.; SPÍNOLA, R. O. 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*. New York, NY, USA: Association for Computing Machinery, 2020. (EASE '20), p. 210219. ISBN 9781450377317. Available at: <https://doi.org/10.1145/3383219.3383241>.

FREIRE, S.; RIOS, N.; MENDONÇA, M.; FALESSI, D.; SEAMAN, C.; IZURIETA, C.; SPÍNOLA, R. O. Actions and impediments for technical debt prevention: Results from a global family of industrial surveys. In: *Proceedings of the 35th Annual ACM Symposium on Applied Computing*. New York, NY, USA: Association for Computing

Machinery, 2020. (SAC '20), p. 15481555. ISBN 9781450368667. Available at: <https://doi.org/10.1145/3341105.3373912>.

FREIRE, S.; RIOS, N.; PÉREZ, B.; TORRES, D.; MENDONÇA, M.; IZURIETA, C.; SEAMAN, C.; SPÍNOLA, R. How do technical debt payment practices relate to the effects of the presence of debt items in software projects? In: *2021 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. [S.l.: s.n.], 2021. p. 605–609.

FREIRE, S.; RIOS, N.; PéREZ, B.; CASTELLANOS, C.; CORREAL, D.; RAMAČ, R.; MANDIĆ, V.; TAUŠAN, N.; LÓPEZ, G.; PACHECO, A.; FALESSI, D.; MENDONÇA, M.; IZURIETA, C.; SEAMAN, C.; SPÍNOLA, R. 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)*. [S.l.: s.n.], 2021. p. 21–30. ISSN 2574-1837.

FREIRE, S.; RIOS, N.; PÉREZ, B.; CASTELLANOS, C.; CORREAL, D.; RAMAČ, R.; MANDIĆ, V.; TAUŠAN, N.; PACHECO, A.; LÓPEZ, G.; MENDONÇA, M.; IZURIETA, C.; FALESSI, D.; SEAMAN, C.; SPÍNOLA, R. Pitfalls and solutions for technical debt management in agile software projects. *IEEE Software*, v. 38, n. 6, p. 42–49, 2021.

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.; SPINOLA, R. Hearing the voice of software practitioners on technical debt monitoring: Understanding monitoring practices and the practices' avoidance reasons. *submitted*, 2022.

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. Software practitioners point of view on technical debt payment. *Journal of Systems and Software*, v. 196, p. 111554, 2023. ISSN 0164-1212. Available at: <https://www.sciencedirect.com/science/article/pii/S0164121222002308>.

FREIRE, S.; ROCHA, V.; MENDONÇA, M.; IZURIETA, C.; SEAMAN, C.; SPINOLA, R. Assessing idea diagrams for supporting analysis of capabilities and issues in technical debt management. In: *submitted*. [S.l.: s.n.], 2022.

FU, L.; LIANG, P.; RASHEED, Z.; LI, Z.; TAHIR, A.; HAN, X. Potential technical debt and its resolution in code reviews: An exploratory study of the openstack and qt communities. In: *Proceedings of the 16th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement*. New York, NY, USA: Association for Computing Machinery, 2022. (ESEM '22), p. 216226. ISBN 9781450394277. Available at: <https://doi.org/10.1145/3544902.3546253>.

GAMA, E.; FREIRE, S.; MENDONÇA, M.; SPÍNOLA, R. O.; PAIXAO, M.; CORTÉS, M. I. Using stack overflow to assess technical debt identification on software projects.

In: *Proceedings of the XXXIV Brazilian Symposium on Software Engineering*. New York, NY, USA: Association for Computing Machinery, 2020. (SBES '20), p. 730739. ISBN 9781450387538. Available at: <https://doi.org/10.1145/3422392.3422429>.

GOMES, F.; SANTOS, E. P. dos; FREIRE, S.; MENDONÇA, M.; MENDES, T. S.; SPÍNOLA, R. Investigating the point of view of project management practitioners on technical debt - a preliminary study on stack exchange. In: *2022 IEEE/ACM International Conference on Technical Debt (TechDebt)*. [S.l.: s.n.], 2022. p. 31–40.

GUO, Y.; SEAMAN, C. A portfolio approach to technical debt management. In: *Proceedings of the 2nd Workshop on Managing Technical Debt*. New York, NY, USA: Association for Computing Machinery, 2011. (MTD '11), p. 3134. ISBN 9781450305860. Available at: <https://doi.org/10.1145/1985362.1985370>.

GUO, Y.; SPINOLA, R. O.; SEAMAN, C. Exploring the costs of technical debt management — a case study. *Empirical Softw. Engg.*, Kluwer Academic Publishers, USA, v. 21, n. 1, p. 159182, feb 2016. ISSN 1382-3256. Available at: <https://doi.org/10.1007/s10664-014-9351-7>.

GUPTA, R. K.; MANIKREDDY, P.; NAIK, S.; ARYA, K. Pragmatic approach for managing technical debt in legacy software project. In: *Proceedings of the 9th India Software Engineering Conference*. New York, NY, USA: Association for Computing Machinery, 2016. (ISEC '16), p. 170176. ISBN 9781450340182. Available at: <https://doi.org/10.1145/2856636.2856655>.

IZURIETA, C.; OZKAYA, I.; SEAMAN, C.; KRUCHTEN, P.; NORD, R.; SNIPES, W.; AVGERIOU, P. Perspectives on managing technical debt: A transition point and roadmap from dagstuhl. In: *CEUR Workshop Proceedings*. [S.l.: s.n.], 2016. (CEUR Workshop Proceedings, v. 1771), p. 84–87. Joint of the 4th International Workshop on Quantitative Approaches to Software Quality, QuASoQ 2016 and 1st International Workshop on Technical Debt Analytics, TDA 2016 ; Conference date: 06-12-2016.

IZURIETA, C.; VETRÒ, A.; ZAZWORKA, N.; CAI, Y.; SEAMAN, C.; SHULL, F. Organizing the technical debt landscape. In: *2012 Third International Workshop on Managing Technical Debt (MTD)*. [S.l.: s.n.], 2012. p. 23–26.

KROGSTIE, J. Quality of conceptual models in model driven software engineering. In: _____. *Conceptual Modeling Perspectives*. Cham: Springer International Publishing, 2017. p. 185–198. ISBN 978-3-319-67271-7. Available at: <https://doi.org/10.1007/978-3-319-67271-7_13>.

KRUCHTEN, P.; NORD, R. L.; OZKAYA, I. Technical debt: From metaphor to theory and practice. *IEEE Software*, v. 29, n. 6, p. 18–21, 2012.

LEHMAN, M. M.; BELADY, L. A. *Program Evolution: Processes of Software Change*. USA: Academic Press Professional, Inc., 1985. ISBN 0124424406.

LENARDUZZI, V.; FUCCI, D. Towards a holistic definition of requirements debt. In: *2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. Los Alamitos, CA, USA: IEEE Computer Society, 2019. p. 1–5. Available at: <https://doi.ieeecomputersociety.org/10.1109/ESEM.2019.8870159>.

LENARDUZZI, V.; ORAVA, T.; SAARIMäKI, N.; SYSTA, K.; TAIBI, D. An empirical study on technical debt in a finnish sme. In: *2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. [S.l.: s.n.], 2019. p. 1–6.

LI, Z.; AVGERIOU, P.; LIANG, P. A systematic mapping study on technical debt and its management. *J. Syst. Softw.*, Elsevier Science Inc., USA, v. 101, n. C, p. 193220, mar 2015. ISSN 0164-1212. Available at: <https://doi.org/10.1016/j.jss.2014.12.027>.

LIENTZ, B. P.; SWANSON, E. B.; TOMPKINS, G. E. Characteristics of application software maintenance. *Commun. ACM*, Association for Computing Machinery, New York, NY, USA, v. 21, n. 6, p. 466471, jun 1978. ISSN 0001-0782. Available at: <https://doi.org/10.1145/359511.359522>.

MANDIĆ, V.; TAUŠAN, N.; RAMAČ, R.; FREIRE, S.; RIOS, N.; PÉREZ, B.; CASTELLANOS, C.; CORREAL, D.; PACHECO, A.; LÓPEZ, G.; IZURIETA, C.; FALESSI, D.; SEAMAN, C.; SPÍNOLA, R. Technical and nontechnical prioritization schema for technical debt: Voice of td-experienced practitioners. *IEEE Software*, v. 38, n. 6, p. 50–58, Nov 2021. ISSN 1937-4194.

MARTINI, A. Anacondebt: A tool to assess and track technical debt. In: *Proceedings of the 2018 International Conference on Technical Debt*. New York, NY, USA: Association for Computing Machinery, 2018. (TechDebt '18), p. 5556. ISBN 9781450357135. Available at: <https://doi.org/10.1145/3194164.3194185>.

MARTINI, A.; BESKER, T.; BOSCH, J. Technical debt tracking: Current state of practice: A survey and multiple case study in 15 large organizations. *Science of Computer Programming*, v. 163, p. 42–61, 2018. ISSN 0167-6423. Available at: <https://www.sciencedirect.com/science/article/pii/S0167642318301035>.

MCCONNELL, S. *Technical debt, 10x Software Development Blog. Construx Conversations.* 2007. Available at: <http://blogs.construx.com/blogs/stevemcc/archive/2007/11/01/technical-debt-2.aspx>.

MENDES, T. S.; GOMES, F. G. S.; GONÇALVES, D. P.; MENDONÇA, M. G.; NOVAIS, R. L.; SPÍNOLA, R. O. Visminertd: a tool for automatic identification and interactive monitoring of the evolution of technical debt items. *Journal of the Brazilian Computer Society*, v. 25, n. 1, p. 2, Jan 2019. ISSN 1678-4804. Available at: <https://doi.org/10.1186/s13173-018-0083-1>.

OLIVEIRA, F.; GOLDMAN, A.; SANTOS, V. Managing technical debt in software projects using scrum: An action research. In: *2015 Agile Conference (AGILE)*. Los Alamitos, CA, USA: IEEE Computer Society, 2015. p. 50–59. Available at: <https://doi.ieeecomputersociety.org/10.1109/Agile.2015.7>.

PARNAS, D. L. Software aging. In: *Proceedings of the 16th International Conference on Software Engineering*. Washington, DC, USA: IEEE Computer Society Press, 1994. (ICSE '94), p. 279287. ISBN 081865855X.

PARNAS, D. L. The limits of empirical studies of software engineering. In: *Proceedings of the 2003 International Symposium on Empirical Software Engineering*. USA: IEEE Computer Society, 2003. (ISESE '03), p. 2. ISBN 0769520022.

PÉREZ, B.; CASTELLANOS, C.; CORREAL, D.; RIOS, N.; FREIRE, S.; SPÍNOLA, R.; SEAMAN, C. What are the practices used by software practitioners on technical debt payment: Results from an international family of surveys. In: *Proceedings of the 3rd International Conference on Technical Debt*. New York, NY, USA: Association for Computing Machinery, 2020. (TechDebt '20), p. 103112. ISBN 9781450379601. Available at: <https://doi.org/10.1145/3387906.3388632>.

PÉREZ, B.; CASTELLANOS, C.; CORREAL, D.; RIOS, N.; FREIRE, S.; SPÍNOLA, R.; SEAMAN, C.; IZURIETA, C. Technical debt payment and prevention through the lenses of software architects. *Inf. Softw. Technol.*, Butterworth-Heinemann, USA, v. 140, n. C, dec 2021. ISSN 0950-5849. Available at: <https://doi.org/10.1016/j.infsof.2021.106692>.

PMI. *A guide to the Project Management Body of Knowledge: (PMBOKő Guide)*. [S.l.]: Project Management Institute, 2017.

RAMAČ, R.; MANDIĆ, V.; TAUŠAN, N.; RIOS, N.; FREIRE, S.; PÉREZ, B.; CASTELLANOS, C.; CORREAL, D.; PACHECO, A.; LÓPEZ, G.; IZURIETA, C.; SEAMAN, C.; SPINOLA, R. Prevalence, common causes and effects of technical debt: Results from a family of surveys with the it industry. *Journal of Systems and Software*, v. 184, p. 111114, 2022. ISSN 0164-1212. Available at: <https://www.sciencedirect.com/science/article/pii/S0164121221002119>.

RAMAČ, R.; TAUŠAN, N.; FREIRE, S.; RIOS, N.; NETO, M. G. de M.; SPÍNOLA, R. O.; MANDIĆ, V. Technical debt payment practices and rationales behind payment avoidance in the serbian it industry. In: LALIC, B.; GRACANIN, D.; TASIC, N.; SIMEUNOVIĆ, N. (Ed.). *Proceedings on 18th International Conference on Industrial Systems – IS'20*. Cham: Springer International Publishing, 2022. p. 103–110. ISBN 978-3-030-97947-8.

RIBEIRO, L. F.; FARIAS, M. A. d. F.; MENDONÇA, M.; SPÍNOLA, R. O. Decision criteria for the payment of technical debt in software projects: A systematic mapping study. In: *Proceedings of the 18th International Conference on Enterprise Information Systems*. Setubal, PRT: SCITEPRESS - Science and Technology Publications, Lda, 2016. (ICEIS 2016), p. 572579. ISBN 9789897581878. Available at: <https://doi.org/10.5220/0005914605720579>.

RIOS, N.; FREIRE, S.; PÉREZ, B.; CASTELLANOS, C.; CORREAL, D.; MENDONCA, M.; FALESSI, D.; IZURIETA, C.; SEAMAN, C. B.; SPINOLA, R. O. On the relationship between technical debt management and process models. *IEEE Software*, v. 38, n. 5, p. 56–64, Sep. 2021. ISSN 1937-4194.

RIOS, N.; MENDES, L.; CERDEIRAL, C.; MAGALHÃES, A. P. F.; PÉREZ, B.; CORREAL, D.; ASTUDILLO, H.; SEAMAN, C.; IZURIETA, C.; SANTOS, G.; SPÍNOLA, R. O. Hearing the voice of software practitioners on causes, effects, and practices to deal with documentation debt. In: *Requirements Engineering: Foundation for Software Quality: 26th International Working Conference, REFSQ 2020, Pisa, Italy, March 2427, 2020, Proceedings*. Berlin, Heidelberg: Springer-Verlag, 2020. p. 5570. ISBN 978-3-030-44428-0. Available at: <https://doi.org/10.1007/978-3-030-44429-7_4>.

RIOS, N.; MENDONÇA, M. G. de; SPINOLA, R. O. A tertiary study on technical debt: Types, management strategies, research trends, and base information for practitioners. *Information and Software Technology*, v. 102, p. 117–145, 2018. ISSN 0950-5849. Available at: <https://www.sciencedirect.com/science/article/pii/S0950584918300946>.

RIOS, N.; SPINOLA, R. O.; MENDONÇA, M.; SEAMAN, C. The most common causes and effects of technical debt: First results from a global family of industrial surveys. In: *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. New York, NY, USA: Association for Computing Machinery, 2018. (ESEM '18). ISBN 9781450358231. Available at: <https://doi.org/10.1145/3239235.3268917>.

RIOS, N.; SPINOLA, R. O.; MENDONÇA, M.; SEAMAN, C. 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)*. [S.l.: s.n.], 2019. p. 3–12.

RIOS, N.; SPÍNOLA, R. O.; MENDONÇA, M.; SEAMAN, C. The practitioners point of view on the concept of technical debt and its causes and consequences: A design for a global family of industrial surveys and its first results from brazil. *Empirical Softw. Engg.*, Kluwer Academic Publishers, USA, v. 25, n. 5, p. 32163287, sep 2020. ISSN 1382-3256. Available at: <https://doi.org/10.1007/s10664-020-09832-9>.

ROCHA, V.; FREIRE, S.; MENDONÇA, M.; SPÍNOLA, R. Evaluating a conceptual framework for supporting technical debt management in testing activities - a feasibility study. In: *Proceedings of the 7th Brazilian Symposium on Systematic and Automated Software Testing*. New York, NY, USA: Association for Computing Machinery, 2022. (SAST '22), p. 6978. ISBN 9781450397537. Available at: <https://doi.org/10.1145/3559744.3559753>.

ROCHA, V.; FREIRE, S.; RIOS, N.; LIMA, C.; RIBEIRO, L.; PéREZ, B.; NETO, A. D.; MOURA, H.; CORREAL, D.; MENDONÇA, M.; SPINOLA, R. A conceptual framework to support the management of technical debt in software testing. In: *27th annual Americas Conference on Information Systems (AMCIS)*. [s.n.], 2021. Available at: <https://aisel.aisnet.org/amcis2021/sig_sand/sig_sand/6>.

SAMARTHYAM, G.; MURALIDHARAN, M.; ANNA, R. K. Understanding test debt. In: _____. *Trends in Software Testing*. Singapore: Springer Singapore, 2017. p. 1–17. ISBN 978-981-10-1415-4. Available at: <https://doi.org/10.1007/978-981-10-1415-4_1>.

SANTOS, E.; GOMES, F.; FREIRE, S.; MENDONÇA, M.; MENDES, T.; SPÍNOLA, R. Technical debt on agile projects: Managers' point of view at stack exchange. In: *XXI Brazilian Symposium on Software Quality*. [S.l.: s.n.], 2022. (SBQS '22).

SEAMAN, C. Qualitative methods in empirical studies of software engineering. *IEEE Transactions on Software Engineering*, v. 25, n. 4, p. 557–572, 1999.

SEAMAN, C.; GUO, Y. Chapter 2 - measuring and monitoring technical debt. In: ZELKOWITZ, M. V. (Ed.). Elsevier, 2011, (Advances in Computers, v. 82). p. 25–46. Available at: <https://www.sciencedirect.com/science/article/pii/B9780123855121000025>.

SEAMAN, C.; GUO, Y.; ZAZWORKA, N.; SHULL, F.; IZURIETA, C.; CAI, Y.; VETRÒ, A. Using technical debt data in decision making: Potential decision approaches. In: *2012 Third International Workshop on Managing Technical Debt (MTD)*. [S.l.: s.n.], 2012. p. 45–48.

SHAHIR, H. Y.; DANESHPAJOUH, S.; RAMSIN, R. Improvement strategies for agile processes: A swot analysis approach. In: *2008 Sixth International Conference on Software Engineering Research, Management and Applications*. [S.l.: s.n.], 2008. p. 221–228.

SILVA, V. M.; JUNIOR, H. J.; TRAVASSOS, G. H. A taste of the software industry perception of technical debt and its management in brazil. *Journal of Software Engineering Research and Development*, v. 7, p. 1:1 1:16, Jul. 2019. Available at: <https://sol.sbc.org.br/journals/index.php/jserd/article/view/19>.

SOARES, G.; FREIRE, S.; RIOS, N.; PÉREZ, B.; MENDONÇA, M.; IZURIETA, C.; SEAMAN, C.; SPÍNOLA, R. Investigating how agile software practitioners repay technical debt in software projects. In: *XXI Brazilian Symposium on Software Quality*. [S.l.: s.n.], 2022. (SBQS '22).

SOUZA, L.; FREIRE, S.; ROCHA, V.; RIOS, N.; SPÍNOLA, R. O.; MENDONÇA, M. Using surveys to build-up empirical evidence on test-related technical debt. In: *Proceedings of the XXXIV Brazilian Symposium on Software Engineering*. New York, NY, USA: Association for Computing Machinery, 2020. (SBES '20), p. 750759. ISBN 9781450387538. Available at: <https://doi.org/10.1145/3422392.3422430>.

SPINOLA, R. O.; ZAZWORKA, N.; VETRÒ, A.; SEAMAN, C.; SHULL, F. Investigating technical debt folklore: Shedding some light on technical debt opinion. In: *Proceedings of the 4th International Workshop on Managing Technical Debt*. [S.l.]: IEEE Press, 2013. (MTD '13), p. 17. ISBN 9781467364430.

SPINOLA, R. O.; ZAZWORKA, N.; VETRÒ, A.; SHULL, F.; SEAMAN, C. Understanding automated and human-based technical debt identification approaches-a two-phase study. *Journal of the Brazilian Computer Society*, v. 25, n. 1, p. 5, Jun 2019. ISSN 1678-4804. Available at: <https://doi.org/10.1186/s13173-019-0087-5>.

STRAUSS, A. L.; CORBIN, J. M. *Basics of qualitative research: techniques and procedures for developing grounded theory.* [S.l.]: Sage Publications, Thousand Oaks, Calif, 1998. XIII, 312 s p.

TOLEDO, S. S. de; MARTINI, A.; PRZYBYSZEWSKA, A.; SJøBERG, D. I. Architectural technical debt in microservices: A case study in a large company. In: *2019 IEEE/ACM International Conference on Technical Debt (TechDebt).* [S.l.: s.n.], 2019. p. 78–87.

WIESE, M.; RIEBISCH, M.; SCHWARZE, J. Preventing technical debt by technical debt aware project management. In: *2021 IEEE/ACM International Conference on Technical Debt (TechDebt).* [S.l.: s.n.], 2021. p. 84–93.

WOHLIN, C.; RUNESON, P.; HST, M.; OHLSSON, M. C.; REGNELL, B.; WESSLN, A. *Experimentation in Software Engineering.* [S.l.]: Springer Publishing Company, Incorporated, 2012. ISBN 3642290434.

YLI-HUUMO, J.; MAGLYAS, A.; SMOLANDER, K. The sources and approaches to management of technical debt: A case study of two product lines in a middle-size finnish software company. In: JEDLITSCHKA, A.; KUVAJA, P.; KUHRMANN, M.; MÄNNISTÖ, T.; MÜNCH, J.; RAATIKAINEN, M. (Ed.). *Product-Focused Software Process Improvement.* Cham: Springer International Publishing, 2014. p. 93–107. ISBN 978-3-319-13835-0.

YLI-HUUMO, J.; MAGLYAS, A.; SMOLANDER, K. How do software development teams manage technical debt? - an empirical study. *J. Syst. Softw.*, Elsevier Science Inc., USA, v. 120, n. C, p. 195218, oct 2016. ISSN 0164-1212. Available at: <https://doi.org/10.1016/j.jss.2016.05.018>.

ZAZWORKA, N.; SPÍNOLA, R. O.; VETRÒ, A.; SHULL, F.; SEAMAN, C. A case study on effectively identifying technical debt. In: *Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering.* New York, NY, USA: Association for Computing Machinery, 2013. (EASE '13), p. 4247. ISBN 9781450318488. Available at: <https://doi.org/10.1145/2460999.2461005>.

Appendix A

INSIGHTD QUESTIONNAIRE

Investigating Causes and Effects of Technical Debt on Software Projects

Dear survey participant,

Thank you very much for taking 10-20 minutes of your valuable time by filling out this questionnaire!

InsighTD is a family of surveys conducted by a worldwide consortium of researchers (www.td-survey.com). Currently, researchers from 7 countries are replicating the study: Brazil, Costa Rica, Colombia, Finland, Netherlands, Serbia, and United States.

Our goal is to help both practitioners and researchers to better understand possible causes and effects of TD presence in software projects. The collected information will lay the foundation for identifying practically relevant problems in the TD area.

At the end of the survey, you will be asked to enter your email address (OPTIONAL). If you agree to provide your e-mail address, we will provide you with an overview of the survey results after an initial analysis. In any case, please be assured that the survey follows a high academic standard and is conducted anonymously. We will not associate your email address with your answers and exclusively use the addresses for the purpose of providing you with the survey results!

Thank you!

* Required

InsighTD Survey
Investigating causes and implications of Technical Debt

Participant Characterization

Please, in this section, provide us with some information about your organization. Besides, think of a specific project that you have participated in, during the last say 3 years, that you would like to tell us more about later. We will ask you to describe the demographics of that project, the role you assumed on it and your experience with software development.

1. **1. What is the size of your company (software and other areas)?** *
   Mark only one oval.

   ◯ 1-10 employees
   ◯ 11-50 employees
   ◯ 51-250 employees
   ◯ 251-500 employees
   ◯ 501-1000 employees
   ◯ 1001-2000 employees
   ◯ more than 2000 employees

2. **2. In which country you are currently working?** *

Mark only one oval.

- Afghanistan
- Albania
- Algeria
- Andorra
- Angola
- Antigua and Barbuda
- Argentina
- Armenia
- Australia
- Austria
- Azerbaijan
- Bahamas
- Bahrain
- Bangladesh
- Barbados
- Belarus
- Belgium
- Belize
- Benin
- Bhutan
- Bolivia
- Bosnia and Herzegovina
- Botswana
- Brazil
- Brunei
- Bulgaria
- Burkina Faso
- Burundi
- Cabo Verde
- Cambodia
- Cameroon
- Canada
- Central African Republic (CAR)
- Chad
- Chile
- China
- Colombia
- Comoros
- Democratic Republic of the Congo
- Republic of the Congo
- Costa Rica

- Cote d'Ivoire
- Croatia
- Cuba
- Cyprus
- Czech Republic
- Denmark
- Djibouti
- Dominica
- Dominican Republic
- Ecuador
- Egypt
- El Salvador
- Equatorial Guinea
- Eritrea
- Estonia
- Ethiopia
- Fiji
- Finland
- France
- Gabon
- Gambia
- Georgia
- Germany
- Ghana
- Greece
- Grenada
- Guatemala
- Guinea
- Guinea-Bissau
- Guyana
- Haiti
- Honduras
- Hungary
- Iceland
- India
- Indonesia
- Iran
- Iraq
- Ireland
- Israel
- Italy
- Jamaica

- Japan
- Jordan
- Kazakhstan
- Kenya
- Kiribati
- Kosovo
- Kuwait
- Kyrgyzstan
- Laos
- Latvia
- Lebanon
- Lesotho
- Liberia
- Libya
- Liechtenstein
- Lithuania
- Luxembourg
- Macedonia (FYROM)
- Madagascar
- Malawi
- Malaysia
- Maldives
- Mali
- Malta
- Marshall Islands
- Mauritania
- Mauritius
- Mexico
- Micronesia
- Moldova
- Monaco
- Mongolia
- Montenegro
- Morocco
- Mozambique
- Myanmar (Burma)
- Namibia
- Nauru
- Nepal
- Netherlands
- New Zealand
- Nicaragua

- Niger
- Nigeria
- North Korea
- Norway
- Oman
- Pakistan
- Palau
- Palestine
- Panama
- Papua New Guinea
- Paraguay
- Peru
- Philippines
- Poland
- Portugal
- Qatar
- Romania
- Russia
- Rwanda
- Saint Kitts and Nevis
- Saint Lucia
- Saint Vincent and the Grenadines
- Samoa
- San Marino
- Sao Tome and Principe
- Saudi Arabia
- Senegal
- Serbia
- Seychelles
- Sierra Leone
- Singapore
- Slovakia
- Slovenia
- Solomon Islands
- Somalia
- South Africa
- South Korea
- South Sudan
- Spain
- Sri Lanka
- Sudan
- Suriname

- Swaziland
- Sweden
- Switzerland
- Syria
- Taiwan
- Tajikistan
- Tanzania
- Thailand
- Timor-Leste
- Togo
- Tonga
- Trinidad and Tobago
- Tunisia
- Turkey
- Turkmenistan
- Tuvalu
- Uganda
- Ukraine
- United Arab Emirates (UAE)
- United Kingdom (UK)
- United States of America (USA)
- Uruguay
- Uzbekistan
- Vanuatu
- Vatican City
- Venezuela
- Vietnam
- Yemen
- Zambia
- Zimbabwe

3. **3. Later, we will be asking you some questions about a particular project and so right now we'd like you to choose what project you will be telling us about. This could be any software development project (current or within the past 3 years) that you had a development role in for a significant amount of time. Once you've chosen what project you will focus on, tell us what is the size of the system being developed in that project? (LOC = lines of code)** *
Mark only one oval.

- less than 10KLOC
- 10–100KLOC
- 100KLOC–1MLOC
- 1–10MLOC
- 10+ MLOC

4. **4. What is the total number of people of this project (include technical staff and business staff)?** *
Mark only one oval.

- Less than 5 people
- 5-9 people
- 10-20 people
- 21-30 people
- More than 30 people

5. **5. What is the age of this system, beginning from initial design and planning, up to now or to when your involvement ended?** *
Mark only one oval.

- Less than 1 year
- 1-2 years
- 2-5 years
- 5-10 years
- More than 10 years

6. **6. To which project role are you assigned in this project?** *
Mark only one oval.

- Business Analyst
- DBA / Data Analyst
- Developer
- Process Analyst
- Project Leader / Project Manager
- Requirements Analyst
- Software Architect
- Test Manager / Tester
- Other: ______________________

7. **7. How do you rate your experience in this role (at the time)?** *
Mark only one oval.

- Novice (Minimal or “textbook" knowledge without connecting it to practice)
- Beginner (Working knowledge of key aspects of practice)
- Competent (Good working and background knowledge of area of practice)
- Proficient (Depth of understanding of discipline and area of practice)
- Expert (Authoritative knowledge of discipline and deep tacit understanding across area of practice)

8. **8. Which of the following most closely describes the development process model you follow on this project?** *

Mark only one oval.

- ◯ Agile (a lightweight process that promotes iterative development, close collaboration between the development team and business side, constant communication, and tightly-knit teams)
- ◯ Hybrid (is the combination of agile methods with other non-agile techniques. For example, a detailed requirements effort, followed by sprints of incremental delivery)
- ◯ Traditional (conventional document-driven software development methods that can be characterized as extensive planning, standardization of development stages, formalized communication, significant documentation and design up front)

TD Concept

In this section, we will collect some information about your familiarity with the concept of Technical Debt.

9. **9. How familiar you are with the concept of Technical Debt?** *

Mark only one oval.

- ◯ Never heard of it
- ◯ I have read about it in books / articles
- ◯ I have been on projects where I recognized TD but the project did not explicitly manage it
- ◯ I have been on projects where we attempted to actively manage TD

10. **10. In your words, how would you define TD?**

TD Concept

Now, please, considering the following definition of TD:

Technical debt contextualizes the problem of outstanding software development tasks (for example, tests planned but not executed, pending code refactoring, pending documentation update, use of bad design practices, code that does not exhibit good coding practices) as a kind of debt that brings a short-term benefit to the project (normally in terms of higher productivity or shorter release time of software versions), that may have to be paid later in the development process with interest (for example, a poorly designed class tends to be more difficult and costly to maintain than if it had been implemented good object-oriented practices).

This definition is based on the work of S. McConnell.
"Technical debt," 10x Software Development Blog,(Nov 2007). Construx Conversations. URL= http://www.construx.com/10x_Software_Development/Technical_Debt/, 2007.

11. **11. How close to the above TD definition is your understanding about TD?** *

Mark only one oval.

- Very close
- Close
- Had no prior knowledge of TD
- Far
- Very far

12. **12. Are there any parts of the definition above from McConnell that you disagree with? Are there some things that you think are TD that are not included in this definition? Does it include things that you think are not TD?**

TD Causes

In this section, we intend to investigate the causes that usually lead development teams to incur TD on their project, as well as what are the most critical causes.

13. **13. Please give an example of TD (that conforms to the definition cited earlier from McConnell) that had a significant impact on the project that you have chosen (back in question 3) to tell us about (this example will be used to answer other questions of this questionnaire):** *

14. **14. Why did you select this example?** *

15. **15. About this example, how representative it is?** *

Mark only one oval.

- It was a unique instance
- It is the type of thing that happens from time to time in the project
- It is the type of thing that happens very often in the project

16. **16. What was the immediate, or precipitating, cause of the example of TD you just described?** *

17. **17. What other cause or factor contributed to the immediate cause you described above?**

18. **18. What other motives or reasons or causes contributed either directly or indirectly to the occurrence of the TD example?**

TD Causes (cont.)

19. **19. Considering all the cases of TD you've encountered in different projects, and the causes of those TD cases, which causes would you say are the most likely to lead to TD (ordered by likelihood of causing TD, with most likely listed first)? Please list up to 5 causes.** *

TD Effects

We are almost finishing the questionnaire!!

In this section, we intend to investigate the effects that TD items have on software projects, as well as what are the most critical effects.

20. **20. Considering the TD item you described in question 13, what were the impacts felt in the project?** *

21. **21. Considering all the cases of TD you've encountered in different projects and the effects of that TD that you have personally experienced, which 5 effects would you classify as the effects that have a bigger impact (ordered by their level of impact, with bigger impact listed first).** *

TD Management

This is the last step of the questionnaire :). Thanks for getting here!!!

For the questions below, please consider the real example of TD that you described in question 13.

22. **22. Do you think it would be possible to prevent the type of debt you described in question 13?** *

Mark only one oval.

- Yes
- No

23. **23. If yes, how? If not, why?**

24. **24. Once identified, was the debt item monitored?** *

Mark only one oval.

- Yes
- No

25. **25. If yes, how? If not, why?**

26. **26. Has the debt item been paid off (eliminated) from the project?** *

Mark only one oval.

- ◯ Yes
- ◯ No

27. **27. If yes, how? If not, why?**

28. **28. Considering your personal experience with TD management, what actions have you performed to prevent the occurrence of debt?**

Finished! Thank you so much for getting here!

29. **If you would like to be notified about the results, please inform us your e-mail address.**

Appendix

B

TECHNICAL DEBT PREVENTION - COMPLEMENTARY MATERIAL

Table B.1: TD prevention-related practices and their examples of citation

TD prevention-related practice	**Quotes from participants**
Adequate technical management	"Better technical leadership could have helped." "Have a technical leader who visualizes the possible impacts of each proposed solution."
Adoption of good programming practices	"Don't let a developer write one thing in a different language than the rest of your project." "Better code quality following design principles and patterns."
Allocation of qualified professionals	"By having at least one experienced developer on the team, who has done similar and knows best practices." "Have highly trained teams."
Appropriate reusing of code	"Take a third party's product long-term viability and direction into account when making a selection." "Always keeping in mind to create reusable code, adapted to best practices that allows modifications, improvements without having to restructure all the code."
Appropriate tasks allocation	"I believe that it is possible to minimize this type of debt with better distribution of activities." "Allocation of a person to carry out system documentation."
Appropriate test coverage	"Define criteria in the process that increase test coverage, increasing code quality." "Perform good quality tests that take into account both functionality and system performance."
Architecture review	"Ensuring times in the estimation of the sprints for the permanent verification of the architecture."
Awareness of the impact of business decisions on technology	"That those responsible for the project are technically advised before making decisions."
Being committed	"Have more commitment from the team throughout the development process, so as not to have future problems." "With competent, committed and truly professional people, and with managers aware of the reality of development teams."
Better project management	"Better project management." "Better organization by management team."
Better project planning	"Planning instead of imagining . . . " "Better planning and estimates."
Better understanding of development process by businesses	"Because businesses need to understand the application development process better across the whole pyramid of management."

(*table continues*)

Table B.1: TD prevention-related practices and their examples of citation (continued)

TD prevention-related practice	**Quotes from participants**
Bug tracking	"Having greater and better control of bugs and versions of deliveries."
Business experts	"Have a person who has the greatest possible knowledge of the business."
Changing team priorities slowly	"Slowly shift team priorities instead of changing pace constantly."
Code review	"Better code review during the development stage." "Conduct ongoing code reviews to ensure project growth is consistent with established quality."
Considering technical constraints	"Having the project managers know when to say no to customers based on what our base application can and cannot do." "Make clear to the customer the technical restrictions imposed and their possible implications."
Continuous integration	"Use of tools for continuous integration." "Early implementation of continuous integration and continuous delivery strategies."
Contracting a domain expert to architect the project	"Defining an architect, who coordinates different teams towards the same goal."
Cost Benefit analysis	"Don't try to make largest margin of profits by any means. I've never seen that do well in the long run."
Creating tests	"Do more testing along the project and write unit tests." "Gradually implement automated tests."
Cultural change	"Promoting a culture change. Improvement of standards. Aiming for excellence."
Deep analysis of functionality	"With a deep analysis of the functionality and application of language techniques to reduce the response time of the method."
Design review	"Design review is performed."
Discipline	"Just performing the appropriate activities." "Greater discipline in engineering and product ownership."
Do it right in the first time	"Get the difficult problems out of the way first, and make sure something is done right the first time."
Documentation update	"That the documentation is being done in line with the creation of the code."
Fair rewarding system	"By rewarding employees after successful release and by giving them bonus they have earned instead of taking all for yourself."
Flexibility in deadlines	"More flexible timelines." "With time flexibility." "Replanning of schedules so that the delivery is more detailed."
Focusing on agile delivery	"More focus on agile delivery frequently." "Applying an agile project management model that better manages time versus scope."
Focusing on long-term goals	"Focus on long-term goals instead of short-term gains." "Decrease in Emphasis on Short-Term Bottom Line and increase on Quality to improve Long-Term Bottom line."
Following the project planning	"Following the plan that has been defined. . . "
Following well-defined project process	"Compliance with the correct software cycle without altering processes and methodologies."
Good allocation of resources	"Carrying out a better analysis at the beginning of the project, seeing its feasibility based on current and obtainable resources in the short term prior to the start." "If a company requires a development, have all the material that is needed on hand."
Good communication between stakeholders	"Communicate with the marketing and sales team regularly and even with customers regularly." "It is not the solution and no one is exempt from technical debt, but maintaining a fluid, concrete community and constant communication with the client mitigates this risk."
Good communication on team	". . . greater team communication." "Working in a unified way in the development of the aforementioned impacts to reduce their effects as much as possible."
Having an emotional stability team	"Better emotional stability on the part of the developer (me), along with a reasonable schedule, tends to make the software better structured, more stable, and cleaner to maintain."

(*table continues*)

Table B.1: TD prevention-related practices and their examples of citation (continued)

TD prevention-related practice	**Quotes from participants**
Having expert consultation on design choices	"For bad design and sub-optimal integration of Spring MVC in existing other frontend framework - solution was to involve more skilled person to help with solution design." "Development plan made by a senior developer/architect."
Historical knowledge on TD	"When the development team knows the reasons that generate technical debt, it can prevent it from happening again." "Using previous TD experience to prevent."
Implementation of a TD identification strategy	"Early recognition. The first time someone copied the code, it should have been a warning sign to get cleaned up." "With the help of tools, you can detect TD early and act immediately on development with best practices."
Implementation of a TD management strategy	"Empower all team members to keep track of the TD and prioritize eliminating it." "Being aware of TD, and be willing to address it in small chunks."
Implementation of a TD payment strategy	"You pay less interest if you have less debt." "But the revenue that the app brings in may not warrant the investment it would take to keep it debt free."
Improve documentation	"Keeping thorough documentation and making it readily available to everyone." "Spend more time on documenting."
Improve requirements elicitation	"Increasing initial time to business understanding and improving initial solutions based on software design." "With the level of experience of senior management and the knowledge of the systems they have, they must perform a better requirement gathering."
Improve schedule to include tests and bug fixing	"Include in the planning the time for testing and correcting errors."
Improve the understanding of TD concept by PMs	"Management should understand consequences and support development team." "Show the business team the short- and medium-term impact of not resolving the TD."
Improving software development process	"Better collaborative processes that also incrementally address quality. . . " "Use a shorter and iterative development cycle where milestones are clearly laid out and are achievable in a viable time frame."
Improving tests	"Improving unit tests." "Running the types of tests that add the most value."
Improving the maintainability of the project	"Establish a checklist that includes necessary and sufficient elements to ensure that solutions evolve naturally and do not involve large and frequent rework."
Increase time for analysis and design	"By taking more time to design and implement a better UI." "Improve analysis times."
IT Governance	"A technological strategic plan and a technological governance plan." "Keep the organization aligned by implementing IT governance and generating a new technological culture within them."
Mirror environment for testing	"We need to implement integration and end-to-end environments where we can run regression tests and where stakeholders can see features before they are released so we can get feedback sooner."
Negotiate with line of business	"By having at least one experienced developer on the team, who has done similar and knows best practices and who can push back on management and tight deadlines."
Organizational support	"Institutional support." "There were ideas for modifications, code rewrites, etc. . . but without administrative support the ideas die."
Organized team	"Having a more measured pace of development, and a better team structure." "With a good team structure, advancing in the knowledge of each area to carry out a project that has no flaws."
Organizing code repository	"Definition of a basic flow for organizing the code repository."
Plan resources for investigating alternative solutions	"Allowing bigger time budget for project launch would provide enough resources for testing the alternate solution which would be better in the long run."

(*table continues*)

Table B.1: TD prevention-related practices and their examples of citation (continued)

TD prevention-related practice	**Quotes from participants**
Planning code structure	"More concrete code planning."
Prioritization of TD payment	"Empower all team members to keep track of the TD and prioritize eliminating it."
Prioritization of test and documentation	"Better planning, estimation and prioritization of tasks such as tests and documentation."
Providing design and requirements sooner	"... providing design and requirements sooner." "Doing sufficient analysis and design before the implementation of the requirements."
Quality assurance	"With quality control well applied to the software development process, it is possible to better identify the emergence of technical debts and resolve them before they become a bigger problem in the future." "Incrementally / iteratively addressing quality with better continuous controls (TDD, BDD, continually addressing design)."
Refactoring	"Doing the refactorings before they become necessary due to breakdowns." "Better process to periodically revisit and refactor projects."
Release only test-approved components	"Do not release to production software that does not comply with the software development life cycle (testing)."
Removing low-priority tasks	"Elimination of low-value or unnecessary tasks."
Requirement validation	"Double Check between requirements analyst, business analyst and end user". "Requirements validation is performed."
Requirements changes tracking	"With monitoring of requirements changes."
Risk and impact analysis	"Identify potential risks before they occur." "Take into account the risks of change, looking not only at what is immediate but also at all its possible consequences."
Self-confidence	"Avoid being influenced by a client's scope and rely more on what experience and best practices dictate."
Standardization in carrying out activities	"Rules and separation of functions."
Team open to changes	"The change is technical and cultural, it would begin with awareness of the importance of activities that, if not carried out, increase the technical debt."
Technical checking of proposals	"Be clearer with the client and not fall into compliance with all their demands."
Technical knowledge	"Understanding of the technology in use." "With a more macro vision of the system and a broader knowledge of the available technologies."
Technical support	"Increased technical support for inexperienced teams." "To have everything clearer, that people have a better disposition to work and help to clarify doubts."
Training	"Keeping the team updated on good development practices." "Considering the study of the technologies used in the project."
Understand team capabilities	"Be aware of your dev. teams capabilities."
Use of diverse test strategies	"Integrate unit tests in development, where in part of the Sprint hours are dedicated to integration tests."
Use the most appropriate version of the technology	"Performing a framework update." "Some of the points could have been remedied using a little more mature technology."
Using agile practices	"With agile methodology it can be prevented." "... agility-oriented planning (with partial delivery approaches)."
Using good design practices	"By spending more time on design and creating specification." "Good analysis and design must be carried out."
Utilizing lessons learned	"We learned something so now we know how to follow it."
Version control	"Having greater and better control of bugs and versions of deliveries." "Use of version control and revised change."
Well planned deadlines	"Adequate time given to developers to READ and UNDERSTAND said requirements." "more time to complete task proper way."

(*table continues*)

Table B.1: TD prevention-related practices and their examples of citation (continued)

TD prevention-related practice	**Quotes from participants**
Well-defined architecture	"If technical team make well architecture that can reduce lots of maintenance work." "It all comes down to the architecture and time spent designing this architecture."
Well-defined effort estimation methods	"by taking into account the time for refactoring and code improvements during the project planning (e.g., add it into estimated time for new features development)." "Better planning and estimates."
Well-defined ER model	"A well-defined documentation and ER model of the solution."
Well-defined metrics	"Thinking of an appropriate solution for the situation, using metrics and best practices."
Well-defined requirements	"More structured requirements . . . " "Better product functional needs forecasting."
Well-defined sales process	"Either train sales people better, or include devs in early stages of project negotiations. Please make sure that sales people are aware of elementary laws of physics and technological limitations."

Table B.2: TD prevention-related practices per type

Type	TD prevention-related practice	#CPRP	%PRPP
Enabling practice	Training	36	7%
	Allocation of qualified professionals	23	4%
	Implementation of a TD identification strategy	23	4%
	Good communication between stakeholders	19	3%
	Implementation of a TD management strategy	19	3%
	Good allocation of resources	16	3%
	Being committed	14	3%
	Using agile practices	14	3%
	Risk and impact analysis	12	2%
	Appropriate tasks allocation	10	2%
	Prioritization of TD payment	10	2%
	Organized team	9	2%
	Technical knowledge	9	2%
	Adequate technical management	8	1%
	Discipline	6	1%
	Good communication on team	6	1%
	Implementation of a TD payment strategy	6	1%
	Well-defined effort estimation methods	6	1%
	Focusing on agile delivery	5	1%
	Version control	5	1%
	Better understanding of development process by businesses	3	1%
	Changing team priorities slowly	3	1%
	Contracting a domain expert to architect the project	3	1%
	Historical knowledge on TD	3	1%
	Plan resources for investigating alternative solutions	3	1%
	Technical support	3	1%
	Awareness of the impact of business decisions on technology	2	0.4%
	Fair rewarding system	2	0.4%
	Having an emotional stability team	2	0.4%
	Improve the understanding of TD concept by PMs	2	0.4%
	IT Governance	2	0.4%
	Mirror environment for testing	2	0.4%
	Organizational support	2	0.4%
	Team open to changes	2	0.4%
	Well-defined metrics	2	0.4%
	Business experts	1	0.2%
	Cost Benefit analysis	1	0.2%
	Cultural change	1	0.2%
	Improve schedule to include tests and bug fixing	1	0.2%
	Negotiate with Line of business	1	0.2%
	Organizing code repository	1	0.2%
	Removing low-priority tasks	1	0.2%
	Self-confidence	1	0.2%
	Understand team capabilities	1	0.2%
	Utilizing lessons learned	1	0.2%
	Well-defined sales process	1	0.2%
Prevention action	Well-defined requirements	57	10%
	Adoption of good programming practices	49	9%
	Better Project Management	43	8%
	Following the project planning	34	6%
	Improving software development process	33	6%
	Improve documentation	26	5%
	Using good design practices	26	5%
	Well planned deadlines	26	5%
	Better project planning	24	4%
	Creating tests	24	4%
	Well-defined architecture	22	4%
	Following well-defined project process	17	3%
	Quality assurance	12	2%
	Refactoring	12	2%
	Code review	10	2%
	Architecture review	9	2%

(*table continues*)

Table B.2: TD prevention-related practices per type (continued)

Type	TD prevention-related practice	#CPRP	%PRPP
	Continuous integration	8	1%
	Increase time for analysis and design	7	1%
	Use the most appropriate version of the technology	7	1%
	Appropriate reusing of code	6	1%
	Appropriate test coverage	6	1%
	Use of diverse test strategies	6	1%
	Having expert consultation on design choices	5	1%
	Improve requirements elicitation	5	1%
	Considering technical constraints	4	1%
	Flexibility in deadlines	4	1%
	Focusing on long-term goals	4	1%
	Improving tests	4	1%
	Improving the maintainability of the project	4	1%
	Providing design and requirements sooner	3	1%
	Requirement validation	3	1%
	Deep analysis of functionality	2	0.4%
	Documentation update	2	0.4%
	Prioritization of test and documentation	2	0.4%
	Requirements changes tracking	2	0.4%
	Technical checkings of proposals	2	0.4%
	Bug tracking	1	0.2%
	Design review	1	0.2%
	Do it right in the first time	1	0.2%
	Planning code structure	1	0.2%
	Release only test-approved components	1	0.2%
	Standardization in carrying out activities	1	0.2%
	Well-defined ER model	1	0.2%

Caption:
#CPRP - Count of TD prevention-related practices.
%PRPP - Percentage of #CPRP in relation to the total of all projects (546)

Table B.3: TD prevention-related practices per nature

Nature	TD prevention-related practice	#CPRP	%PRPP
Managerial	Better Project Management	43	8%
	Following the project planning	34	6%
	Improving software development process	33	6%
	Well planned deadlines	26	5%
	Better project planning	24	4%
	Allocation of qualified professionals	23	4%
	Implementation of a TD identification strategy	23	4%
	Good communication between stakeholders	19	3%
	Implementation of a TD management strategy	19	3%
	Following well-defined project process	17	3%
	Good allocation of resources	16	3%
	Being committed	14	3%
	Using agile practices	14	3%
	Risk and impact analysis	12	2%
	Appropriate tasks allocation	10	2%
	Prioritization of TD payment	10	2%
	Organized team	9	2%
	Adequate technical management	8	1%
	Increase time for analysis and design	7	1%
	Good communication on team	6	1%
	Implementation of a TD payment strategy	6	1%
	Well-defined effort estimation methods	6	1%
	Focusing on agile delivery	5	1%
	Flexibility in deadlines	4	1%
	Focusing on long-term goals	4	1%
	Better understanding of development process by businesses	3	1%
	Changing team priorities slowly	3	1%
	Historical knowledge on TD	3	1%
	Plan resources for investigating alternative solutions	3	1%
	Awareness of the impact of business decisions on technology	2	0.4%
	Fair rewarding system	2	0.4%
	Having an emotional stability team	2	0.4%
	Improve the understanding of TD concept by PMs	2	0.4%
	Organizational support	2	0.4%
	Cost Benefit analysis	1	0.2%
	Cultural change	1	0.2%
	Improve schedule to include tests and bug fixing	1	0.2%
	Negotiate with Line of business	1	0.2%
	Removing low-priority tasks	1	0.2%
	Self-confidence	1	0.2%
	Standardization in carrying out activities	1	0.2%
	Understand team capabilities	1	0.2%
	Utilizing lessons learned	1	0.2%
	Well-defined sales process	1	0.2%
Technical	Well-defined requirements	57	10%
	Adoption of good programming practices	49	9%
	Training	36	7%
	Improve documentation	26	5%
	Using good design practices	26	5%
	Creating tests	24	4%
	Well-defined architecture	22	4%
	Quality assurance	12	2%
	Refactoring	12	2%
	Code review	10	2%
	Architecture review	9	2%
	Technical knowledge	9	2%
	Continuous integration	8	1%
	Use the most appropriate version of the technology	7	1%
	Appropriate reusing of code	6	1%
	Appropriate test coverage	6	1%
	Discipline	6	1%
	Use of diverse test strategies	6	1%

(*table continues*)

Table B.3: TD prevention-related practices per nature (continued)

Nature	TD prevention-related practice	#CPRP	%PRPP
	Having expert consultation on design choices	5	1%
	Improve requirements elicitation	5	1%
	Version control	5	1%
	Considering technical constraints	4	1%
	Improving tests	4	1%
	Improving the maintainability of the project	4	1%
	Contracting a domain expert to architect the project	3	1%
	Providing design and requirements sooner	3	1%
	Requirement validation	3	1%
	Technical support	3	1%
	Deep analysis of functionality	2	0.4%
	Documentation update	2	0.4%
	IT Governance	2	0.4%
	Mirror environment for testing	2	0.4%
	Prioritization of test and documentation	2	0.4%
	Requirements changes tracking	2	0.4%
	Team open to changes	2	0.4%
	Technical checkings of proposals	2	0.4%
	Well-defined metrics	2	0.4%
	Bug tracking	1	0.2%
	Business experts	1	0.2%
	Design review	1	0.2%
	Do it right in the first time	1	0.2%
	Organizing code repository	1	0.2%
	Planning code structure	1	0.2%
	Release only test-approved components	1	0.2%
	Well-defined ER model	1	0.2%

Caption:
#CPRP - Count of TD prevention-related practices.
%PRPP - Percentage of #CPRP in relation to the total of all projects (546)

Table B.4: TD prevention-related practices per category

Category	TD prevention-related practice	#CPRP	%PRPP
Development issues	Adoption of good programming practices	49	9%
	Quality assurance	12	2%
	Appropriate reusing of code	6	1%
	Having expert consultation on design choices	5	1%
	Version control	5	1%
	Considering technical constraints	4	1%
	Plan resources for investigating alternative solutions	3	1%
	Documentation update	2	0.4%
	Planning code structure	1	0.2%
Infrastructure	Use the most appropriate version of the technology	7	1%
	Organizing code repository	1	0.2%
Internal quality issues	Well-defined requirements	57	10%
	Using good design practices	26	5%
	Well-defined architecture	22	4%
	Refactoring	12	2%
	Improving the maintainability of the project	4	1%
	Well-defined ER model	1	0.2%
Methodology	Improving software development process	33	6%
	Improve documentation	26	5%
	Creating tests	24	4%
	Good communication between stakeholders	19	3%
	Following well-defined project process	17	3%
	Using agile practices	14	3%
	Code review	10	2%
	Architecture review	9	2%
	Continuous integration	8	1%
	Appropriate test coverage	6	1%
	Good communication on team	6	1%
	Use of diverse test strategies	6	1%
	Focusing on agile delivery	5	1%
	Improve requirements elicitation	5	1%
	Improving tests	4	1%
	Providing design and requirements sooner	3	1%
	Requirement validation	3	1%
	Deep analysis of functionality	2	0.4%
	Mirror environment for testing	2	0.4%
	Prioritization of test and documentation	2	0.4%
	Requirements changes tracking	2	0.4%
	Technical checkings of proposals	2	0.4%
	Bug tracking	1	0.2%
	Design review	1	0.2%
	Standardization in carrying out activities	1	0.2%
	Understand team capabilities	1	0.2%
	Utilizing lessons learned	1	0.2%
	Release only test-approved components	1	0.2%
Organizational	Training	36	7%
	Better understanding of development process by businesses	3	1%
	Historical knowledge on TD	3	1%
	Contracting a domain expert to architect the project	3	1%
	Technical support	3	1%
	Fair rewarding system	2	0.4%
	Organizational support	2	0.4%
	IT Governance	2	0.4%
	Cultural change	1	0.2%
	Business experts	1	0.2%
People	Being committed	14	3%
	Technical knowledge	9	2%
	Organized team	9	2%
	Discipline	6	1%
	Having an emotional stability team	2	0.4%
	Improve the understanding of TD concept by PMs	2	0.4%
	Team open to changes	2	0.4%

(*table continues*)

Table B.4: TD prevention-related practices per category (continued)

Category	TD prevention-related practice	#CPRP	%PRPP
	Self-confidence	1	0.2%
	Do it right in the first time	1	0.2%
Planning and management	Better Project Management	43	8%
	Following the project planning	34	6%
	Well planned deadlines	26	5%
	Better project planning	24	4%
	Allocation of qualified professionals	23	4%
	Implementation of a TD identification strategy	23	4%
	Implementation of a TD management strategy	19	3%
	Good allocation of resources	16	3%
	Risk and impact analysis	12	2%
	Appropriate tasks allocation	10	2%
	Prioritization of TD payment	10	2%
	Adequate technical management	8	1%
	Increase time for analysis and design	7	1%
	Implementation of a TD payment strategy	6	1%
	Well-defined effort estimation methods	6	1%
	Flexibility in deadlines	4	1%
	Focusing on long-term goals	4	1%
	Changing team priorities slowly	3	1%
	Awareness of the impact of business decisions on technology	2	0.4%
	Well-defined metrics	2	0.4%
	Cost Benefit analysis	1	0.2%
	Improve schedule to include tests and bug fixing	1	0.2%
	Negotiate with Line of business	1	0.2%
	Removing low-priority tasks	1	0.2%
	Well-defined sales process	1	0.2%

Caption:
#CPRP - Count of TD prevention-related practices.
%PRPP - Percentage of #CPRP in relation to the total of all projects (546)

Table B.5: PARs for TD non-prevention and their examples of citation

PAR for TD non-prevention	**Quotes from participants**
Architectural evolution	"Due to the constant evolution of architecture."
Continuous change of coding standards	"Standards will always change with new technologies and personnel; the debt could be minimized by being more proactive with refactoring but not eliminated entirely."
Debt close to the project end	"The technical debt only existed at a later time."
Dev teams interdependency	"Dependencies with other teams and inability to prevent all possible situations of difficulty."
Differences among stakeholders	"There are always discrepancies with different stakeholders."
Documentation issues (lack of or non-updated)	"When youre building on a platform that does not have sufficient documentation, you may not know how your code will work with the limits until a prototype is tested." "Documentation for the customer must always be up to date."
Ineffective management	"Management and process were not mature enough to prevent TD." "There are demands that arise and must be resolved promptly."
Lack of concern about maintainability	"Maintainability is seldom a developer's concern."
Lack of experience	"It is natural for developers to have incomplete knowledge or lack of experience when developing."
Lack of financial resources	"Money. Time is a resource that costs money."
Lack of good technical solutions	"At the time the decision was made, we investigated all possible solutions and came up with the least bad solution."
Lack of information	"It is natural for developers to have incomplete knowledge or lack of experience when developing."
Lack of predictability in the software development	"No large project ever has the foresight to account for every scenario." "It's difficult to see into the future to know what it will take to develop software. Sometimes newly added requirements or specs can change what is expected out of a class which outdates its ability to function properly."
Lack of process maturity	"Management and process were not mature enough to prevent TD."
Lack of qualified professionals	"Not all developers are or can be extremely excellent (many are, of course)."
Lack of technical knowledge	"The solution adopted was one of the best considering the restrictions of available infrastructure, deadline and technical knowledge." "I believe that it would not be possible to avoid this TD, as the main cause is the lack of knowledge of ReactJS. Also, the type of knowledge needed is very specific and not something you can get through documentation or courses. It is necessary to use ReactJS in practice to solve real problems to understand what are the limitations/difficulties of native state control, and only then look for alternatives."
Legacy system difficult to heal	"New features, new markets and new systems will be built on top of existing systems.Like skyscrapers built on sandcastles the old systems will be eventually replaced by newer better systems redesigned from scratch."
Not sure if client will accept it	"the functionality was experimental, there was no certainty that the user would accept it, it was not worth the effort."
People issues	"I believe this kind of situation will always occur as it is not just one reason that causes it but it can come from many causes, code is made by people and we are all different."
Pressure for results	"No, because the pressure of business and priorities simply disrupts the rigorous technical work." "The pressure for results and compliance schedules (which are always required in terms of monetary benefit), mean that there is technical debt practically from day 1 of the project."
Requirements change	"Requirements are always going to change during development . . . " "Because when the client asks for features abruptly, no matter how generalized the architecture is towards the problem, with an outlier there may be, that can mean a refactor of the code, and that could dirty the code, reducing its maintainability."
Restrictions on available infrastructure	"The solution adopted was one of the best considering the restrictions of available infrastructure, deadline and technical knowledge."

(*table continues*)

Table B.5: PARs for TD non-prevention and their examples of citation (continued)

PAR for TD non-prevention	**Quotes from participants**
Short deadline	“Because we always need to make trade-offs due to time pressure.” “A lot of the TD Ive encountered will not be solved because it’s a symptom of market forces. There will always be a rush to get a product out.”

Table B.6: PARs for TD non-prevention per type

Type	**PAR for TD non-prevention**	**#CPAR**	**%PARP**
Decision factor	Ineffective management	7	14%
	Lack of predictability in the software development	5	10%
	Requirements change	5	10%
	Documentation issues (lack of or non-updated)	2	4%
	Lack of concern about maintainability	2	4%
	lack of process maturity	2	4%
	Architectural evolution	1	2%
	Continuous change of coding standards	1	2%
	Dev teams interdependency	1	2%
	People issues	1	2%
Impediment	Short deadline	14	29%
	Pressure for results	4	8%
	Lack of technical knowledge	3	6%
	Lack of good technical solutions	2	4%
	Lack of qualified professionals	2	4%
	Legacy system difficult to heal	2	4%
	Debt close to the project end	1	2%
	Differences among stakeholders	1	2%
	Lack of experience	1	2%
	Lack of financial resources	1	2%
	Lack of information	1	2%
	Not sure if client will accepted it	1	2%
	Restrictions on available infrastructure	1	2%

Caption:
#CPAR - Count of citations of each PAR
%PARP - Percentage of CPAR in relation to the total of all projects (49)

Table B.7: PARs for TD non-prevention per type per nature

Nature	PARs for TD non-prevention	#CPAR	%PARP
Managerial	Short deadline	14	29%
	Ineffective management	7	14%
	Lack of predictability in the software development	5	10%
	Pressure for results	4	8%
	Lack of concern about maintainability	2	4%
	lack of process maturity	2	4%
	Lack of qualified professionals	2	4%
	Debt close to the project end	1	2%
	Dev teams interdependency	1	2%
	Differences among stakeholders	1	2%
	Lack of financial resources	1	2%
	Not sure if client will accepted it	1	2%
	People issues	1	2%
Technical	Requirements change	5	10%
	Lack of technical knowledge	3	6%
	Documentation issues (lack of or non-updated)	2	4%
	Lack of good technical solutions	2	4%
	Legacy system difficult to heal	2	4%
	Architectural evolution	1	2%
	Continuous change of coding standards	1	2%
	Lack of experience	1	2%
	Lack of information	1	2%
	Restrictions on available infrastructure	1	2%

Caption:
#CPAR - Count of citations of each PAR
%PARP - Percentage of CPAR in relation to the total of all projects (49)

Table B.8: PARs for TD non-prevention per category

Category	**PARs for TD non-prevention**	**#CPAR**	**%PARP**
Development issues	Requirements change	5	10%
	Lack of good technical solutions	2	4%
	Legacy system difficult to heal	2	4%
	Architectural evolution	1	2%
	Continuous change of coding standards	1	2%
External factors	Pressure for results	4	8%
	Differences among stakeholders	1	2%
	Not sure if client will accepted it	1	2%
Lack of knowledge	Lack of technical knowledge	3	6%
	Lack of information	1	2%
Methodology	Lack of predictability in the software development	5	10%
	Documentation issues (lack of or non-updated)	2	4%
	lack of process maturity	2	4%
	Dev teams interdependency	1	2%
Organizational	Lack of qualified professionals	2	4%
	Lack of financial resources	1	2%
	Restrictions on available infrastructure	1	2%
People	Lack of experience	1	2%
	People issues	1	2%
Planning and management	Short deadline	14	29%
	Ineffective management	7	14%
	Lack of concern about maintainability	2	4%
	Debt close to the project end	1	2%

Caption:
#CPAR - Count of citations of each PAR
%PARP - Percentage of CPAR in relation to the total of all projects (49)

Table B.9: Comparison to related work on TD prevention-related practices

Type	Technical debt prevention-related practices from	
	Our Study	**Related Work**
Enabling practice	Adequate technical management	Managing the requirements, deliveries, backlog delays in progress (BONFIM; BENITTI, 2022)
	Allocation of qualified professionals	-
	Appropriate tasks allocation	Define roles concerning the documentation process (RIOS *et al.*, 2020)
	Awareness of the impact of business decisions on technology	-
	Being committed	-
	Better understanding of development process by businesses	-
	Business experts	-
	Changing team priorities slowly	-
	Contracting a domain expert to architect the project	-
	Cost Benefit analysis	-
	Cultural change	-
	Discipline	-
	Fair rewarding system	-
	Focusing on agile delivery	-
	Good allocation of resources	-
	Good communication between stakeholders	Customer feedback (CODABUX; WILLIAMS; NIU, 2014), Communication structure between business management and development team (YLI-HUUMO; MAGLYAS; SMOLANDER, 2014), Learning from customers (LENARDUZZI *et al.*, 2019), Meetings (ALBUQUERQUE *et al.*, 2022)
	Good communication on team	-
	Having an emotional stability team	-
	Historical knowledge on TD	-
	Implementation of a TD identification strategy	Increasing awareness in the development and test teams on test debt (SAMARTHYAM; MURALIDHARAN; ANNA, 2017)
	Implementation of a TD management strategy	Present already identified debts (ARAGÃO *et al.*, 2022)
	Implementation of a TD payment strategy	Having each development team dedicating one iteration during a set release period towards debt reduction (CODABUX; WILLIAMS; NIU, 2014), Introducing relevant processes can also help stop accumulation of debt (SAMARTHYAM; MURALIDHARAN; ANNA, 2017)
	Improve schedule to include tests and bug fixing	Bugs bashes (CODABUX; WILLIAMS; NIU, 2014), Bug fixing days (YLI-HUUMO; MAGLYAS; SMOLANDER, 2014), Technical debt awareness (GUPTA *et al.*, 2016)
	Improve the understanding of TD concept by PMs	-
	IT Governance	-
	Mirror environment for testing	-
	Negotiate with Line of business	-
	Organizational support	-
	Organized team	-
	Organizing code repository	-
	Plan resources for investigating alternative solutions	-
	Prioritization of TD payment	Having dedicated teams whose primary focus is on reducing TD (CODABUX; WILLIAMS; NIU, 2014)
	Removing low-priority tasks	-
	Risk and impact analysis	-
	Self-confidence	-
	Team open to changes	-
	Technical knowledge	-
	Technical support	-

(*table continues*)

Table B.9: Comparison to related work on TD prevention-related practices (continued)

Type	Technical debt prevention-related practices from	
	Our Study	**Related Work**
	Training	Education and training (CODABUX; WILLIAMS; NIU, 2014), Training on the problems by dont document (RIOS *et al.*, 2020)
	Understand team capabilities	-
	Using agile practices	Performing ceremonies (initials, sprint planning, sprint review) (BONFIM; BENITTI, 2022)
	Utilizing lessons learned	-
	Version control	-
	Well-defined effort estimation methods	Careful estimation (LENARDUZZI *et al.*, 2019)
	Well-defined metrics	Gather and analyze metrics on the code base (ERNST; DELANGE; KAZMAN, 2021)
	Well-defined sales process	-
Prevention practice	Adoption of good programming practices	Coding standards and guides (YLI-HUUMO; MAGLYAS; SMOLANDER, 2014; ARAGÃO *et al.*, 2022), Coding standards (YLI-HUUMO; MAGLYAS; SMOLANDER, 2016; BOMFIM; SANTOS, 2017; ALBUQUERQUE *et al.*, 2022), Comment the code (RIOS *et al.*, 2020)
	Appropriate reusing of code	-
	Appropriate test coverage	-
	Architecture review	-
	Better project management	-
	Better project planning	-
	Bug tracking	-
	Code review	Pair programming (CODABUX; WILLIAMS; NIU, 2014), Code reviews (YLI-HUUMO; MAGLYAS; SMOLANDER, 2014; YLI-HUUMO; MAGLYAS; SMOLANDER, 2016; ALBUQUERQUE *et al.*, 2022), Code revisions (SILVA; JUNIOR; TRAVASSOS, 2019; APA *et al.*, 2020b; APA *et al.*, 2020a), efficient code reviews (ERNST; DELANGE; KAZMAN, 2021)
	Considering technical constraints	-
	Continuous integration	Continuous integration (CODABUX; WILLIAMS; NIU, 2014), Ernst, Delange and Kazman (2021): Automated gradual deployments, Integrate deployment with your continuous integration pipeline
	Creating tests	Automate testing activities (ERNST; DELANGE; KAZMAN, 2021), Automated tests (ALBUQUERQUE *et al.*, 2022)
	Deep analysis of functionality	-
	Design review	-
	Do it right in the first time	-
	Documentation update	-
	Flexibility in deadlines	-
	Focusing on long-term goals	-
	Following the project planning	-
	Following well-defined project process	Conformance to process and standards (CODABUX; WILLIAMS; NIU, 2014)
	Having expert consultation on design choices	-

(*table continues*)

Table B.9: Comparison to related work on TD prevention-related practices (continued)

Type	**Technical debt prevention-related practices from**	
	Our Study	**Related Work**
	Improve documentation	Rios *et al.* (2020): Define process and good practices for documentation, Have a documentation repository, Improve commitment of the team concerning documentation, Involve several roles in documenting the Project, Penalties if not follow the documentation process, Use of peer review, Use of UML to document and share information; Documenting design and technical debt (ERNST; DELANGE; KAZMAN, 2021), Bonfim and Benitti (2022): Keeping and describing the requirements specification document clearly, Developing prototypes as part of the requirements specification document, drawing up sequence and use case diagram and database model
	Improve requirements elicitation	Do a good job of requirements elicitation (ERNST; DELANGE; KAZMAN, 2021), leverage crowd-sourcing techniques for gathering requirements (ERNST; DELANGE; KAZMAN, 2021), Building process flow or BDD to understand the demand (BONFIM; BENITTI, 2022), performing requirements gathering thought multidisciplinary workshops (BONFIM; BENITTI, 2022)
	Improving software development process	Ernst, Delange and Kazman (2021): Adopt test-driven development, Define a deployment process, Implement kill switches on new features; Use a framework (WIESE; RIEBISCH; SCHWARZE, 2021), Following a requirements process (BONFIM; BENITTI, 2022)
	Improving tests	Continuous improvement in functional test automation (GUPTA *et al.*, 2016), Review elaborated test cases (ARAGÃO *et al.*, 2022), Avoid manual testing (ERNST; DELANGE; KAZMAN, 2021)
	Improving the maintainability of the project	-
	Increase time for analysis and design	-
	Planning code structure	-
	Prioritization of test and documentation	Document the project since its begin (RIOS *et al.*, 2020)
	Providing design and requirements sooner	-
	Quality assurance	-
	Refactoring	Refactoring (CODABUX; WILLIAMS; NIU, 2014; YLI-HUUMO; MAGLYAS; SMOLANDER, 2014)
	Release only test-approved components	-
	Requirement validation	Bonfim and Benitti (2022): Defining and validating what will be prioritized, validating prototypes after requirements specification with customer, performing experimentation, prototypes, MVP or customer's researcher
	Requirements changes tracking	Traceability (ERNST; DELANGE; KAZMAN, 2021), Performing requirements traceability (BONFIM; BENITTI, 2022)
	Standardization in carrying out activities	-
	Technical checkings of proposals	-
	Use of diverse test strategies	-
	Use the most appropriate version of the technology	Tools (CODABUX; WILLIAMS; NIU, 2014), Reviews of the used tools (YLI-HUUMO; MAGLYAS; SMOLANDER, 2016), Choose your language and libraries wisely (ERNST; DELANGE; KAZMAN, 2021)
	Using good design practices	Employ a design method such as attribute-driven design (ERNST; DELANGE; KAZMAN, 2021)
	Well planned deadlines	-
	Well-defined architecture	-
	Well-defined ER model	-

(*table continues*)

Table B.9: Comparison to related work on TD prevention-related practices (continued)

Type	**Technical debt prevention-related practices from**	
	Our Study	**Related Work**
	Well-defined requirements	Bonfim and Benitti (2022): Performing the requirements or demands refinement (sprint release), Registering and classifying FR and NFR - using checklist for NFR
-	-	Allowing slack (CODABUX; WILLIAMS; NIU, 2014)
-	-	Reflective improvement (CODABUX; WILLIAMS; NIU, 2014)
-	-	Definition of done to ensure code quality (YLI-HUUMO; MAGLYAS; SMOLANDER, 2016)
-	-	Definition of the done standard (YLI-HUUMO; MAGLYAS; SMOLANDER, 2016), Improved definition of done (GUPTA *et al.*, 2016), Definition of done (SILVA; JUNIOR; TRAVASSOS, 2019; APA *et al.*, 2020b; APA *et al.*, 2020a)
-	-	Following boy scout rule (GUPTA *et al.*, 2016)
-	-	Continuous improvement (GUO; SPINOLA; SEAMAN, 2016), Continuous improvement (LENARDUZZI *et al.*, 2019)
-	-	Use tools (CHARALAMPIDOU *et al.*, 2018)
-	-	Maintain and analyze test runs (ERNST; DELANGE; KAZMAN, 2021)
-	-	Check quality of the documentation as part of the release process (ERNST; DELANGE; KAZMAN, 2021)
-	-	Ernst, Delange and Kazman (2021): Focus on how to properly design machine learning systems, Evaluate the differences in performance and accuracy between the model being used and state-of-the-art models for which benchmarks are available
-	-	Guidelines (ALBUQUERQUE *et al.*, 2022)
-	-	Recording all requirements, demands, and stories in the backlog (BONFIM; BENITTI, 2022)

Appendix C

TECHNICAL DEBT MONITORING - COMPLEMENTARY MATERIAL

Table C.1: TD monitoring-related practices and their examples of citation

TD monitoring-related practice	**Quotes from participants**
Adoption of agile methodology	“Yes, using agile methodologies and tools for that.” “An agile project methodology was adopted to close gaps, monitor project progress and results.”
Architecture reviewing	“An analysis of the initial architecture was carried out and an architecture was designed that fulfilled the same functions but was better organized.”
Assign team for TD monitoring	“In the following sprints, the quality team started to have more monitoring.” “By the documentation team.”
Code review	“Every pull request changing the login page must be looked at by least 2 architects.” “(. . .) code reviews done during sprint.”
Communicating the stakeholders of TD items	“The client is perfectly aware of it.” “It was monitored in the sense that the team had a shared awareness and discussed its ongoing impact.”
Continuous deployment	“(. . .) the tools for automated delivery were defined.”
Continuous integration	“Test with continuous integration, monitoring with sonar etc..”
Cost/benefit analysis	“We look at the cost of switching and the features available.”
Dashboard	“Activity tracking.” “Logs, dashboard and alarms.”
Documentation review	“Through tools such as code coverage and documentation auditing.” “In the sprint review, the documentation is checked.”
Focusing on TD payment	“We were aware of what we are doing and planned to “pay the debt” later on in the maintenance stage of the project.”
Identify the worst debt areas	“Most of us who are involved with creating and supporting the product are aware of the worst debt areas.”
Identifying TD items	“We identified the issue we were placed into and when we can address it.”
Improve documentation	“Forcing proper definition and documentation.”
Improving configuration management practices	“Code refactored, database versioned, git history cleaner by commit rules.”
Improving planning	“Better planning and organization.”

(*table continues*)

Table C.1: TD monitoring-related practices and their examples of citation (continued)

TD monitoring-related practice	**Quotes from participants**
Improving software development process	“Started with a training process, application of agility and the adoption of the best development practices.” “Through the different reprocesses that occurred.”
Improving tests	“A test plan was made in conjunction with the user, and they were run again.” “(. . .) we began to incorporate more end-to-end tests to gate delivery.”
Improving the requirement management	“Active management of the requirements to get to implementation quickly.”
Infrastructure monitoring	“The infrastructure sector monitors on-premise servers in the company’s custody.”
Knowledge sharing	“With the adoption of Scrum and the use of meetings every day to align the understanding of the development team about the solution that was being developed, and also the practice of pair programming that made it possible to disseminate knowledge.”
Measuring the effort	“We often created timeboxes for how much time we want to spend on updating or maintaining a component. Also, we monitor how much support time we are spending on the app.”
Prioritization of TD items	“To have the user story and prioritize it.”
Process automation	“By automating the process which brought problems to our attention immediately.”
Pull request monitoring	“Through bitbucket functionalities which monitored the pull requests of the user with debt.”
Qualified professionals	“Agile methodologies and Devops were implemented, the team grew, and qualified personnel and specialists were brought in each of their areas.”
Quality validation (meetings)	“Continuously evaluate the quality of the system.”
Refactoring	“Refactoring took place before the development of other services took place.” “Refactoring when appropriate and time allowed.”
Risk analysis	“Before each project, risks were reported to the IT manager.”
Support from project management	“Project management committed to supporting both the user and the developer to minimize the impact on other project tasks.”
TD as a task	“Identified as a task and placed in the backlog for later prioritization.” “Technical debt sub-tasks associated with each task were created.”
TD estimation	“There was a constant estimation of the scope and possible cost of refactoring.”
TD item backlog	“Keep a list of all TD and regularly determine how to eliminate it.” “Identified as a task and placed in the backlog for later prioritization.”
TD management plan	“A schedule was developed specifically to address the case.” “We were aware of what we are doing and planned to pay the debt later on in maintenance stage of the project.”
TD status progress report	“Project manager asked for daily reports on fixes and why things were needing refixed.” “Through daily reports on the progress of projects and possible future stoppers.”
Team meetings	“Regular meeting, usually weekly held that would track the progress.” “(. . .) the use of meetings every day to align the understanding of the development team about the solution that was being developed (. . .).”
Team restructuring	“The work team was structured more efficiently.”
Test automation	“Implementation of automatic tests with requirements for increasing or maintaining coverage, separation of services, integration and continuous deployment, and gradual migration of services and clients.”

(*table continues*)

Table C.1: TD monitoring-related practices and their examples of citation (continued)

TD monitoring-related practice	**Quotes from participants**
Tracking TD items	“Work considered part of the technical debt was tracked.” “Through the task/issue tracking system.”
Tracking the cost	“We tracked the cost of corrective defective data.”
Training	“Beginning with a process of training, application of agility, and appropriation of best practices in development.”
Understanding the cause of TD item	“Studying the root cause of the problem.”
Use of measuring reports	“With measurement bulletins.”
Use of metrics for TD identification	“By introducing code/test metrics.” “Monitoring based on metrics during the development of the project.”
Use of tools	“We have to use tool to monitor it, this is an ongoing process.” “Use of tools like SonarQube.”
Using informal practices	“Informally. Having high hopes to rectify it by the end of the project.”

Table C.2: TD monitoring-related practices per type

Type	TD monitoring-related practice	#CMRP	%MRPP
Enabling practice	Use of tools	31	12%
	Improving software development process	20	8%
	TD Management Plan	11	4%
	Assign team for TD monitoring	8	3%
	Adoption of agile methodology	7	3%
	Improving the requirement management	6	2%
	Quality validation (meetings)	4	2%
	Continuous integration	2	0.8%
	Knowledge sharing	2	0.8%
	Improving planning	1	0.4%
	Infrastructure Monitoring	1	0.4%
	Process automation	1	0.4%
	Pull request monitoring	1	0.4%
	Support from project management	1	0.4%
	Team restructuring	1	0.4%
Monitoring action	TD item backlog	34	13%
	Team meetings	23	9%
	Communicating the stakeholders of TD items	16	6%
	Tracking TD items	12	5%
	Prioritization of TD items	7	3%
	TD as a task	7	3%
	TD status progress report	6	2%
	Measuring the effort	5	2%
	TD estimation	4	2%
	Understanding the Cause of TD Item	3	1%
	Dashboard	2	0.8%
	Risk analysis	2	0.8%
	Tracking the cost	2	0.8%
	Cost/benefit analysis	1	0.4%
	Identify the worst debt areas	1	0.4%
	Use of measuring reports	1	0.4%
	Using informal practices	1	0.4%
TD identification	Use of metrics for TD identification	7	3%
	Identifying TD items	5	2%
TD payment	Refactoring	18	7%
	Improve documentation	6	2%
	Focusing on TD payment	5	2%
TD prevention	Improving tests	17	7%
	Code review	16	6%
	Qualified professionals	3	1%
	Training	3	1%
	Documentation review	2	0.8%
	Improving configuration management practices	2	0.8%
	Test automation	2	0.8%
	Architecture reviewing	1	0.4%
	Continuous deployment	1	0.4%

Caption:
#CMRP - Count of monitoring-related practices.
%MRPP - Percentage of #CMRP in relation to the total of all projects (259)

Table C.3: TD monitoring-related practices per nature

Nature	monitoring-related practice	#CMRP	%MRPP
Managerial	TD item backlog	34	13%
	Team meetings	23	9%
	Improving software development process	20	8%
	Communicating the stakeholders of TD items	16	6%
	Tracking TD items	12	5%
	TD Management Plan	11	4%
	Assign team for TD monitoring	8	3%
	Prioritization of TD items	7	3%
	TD as a task	7	3%
	Improving the requirement management	6	2%
	TD status progress report	6	2%
	Focusing on TD payment	5	2%
	Measuring the effort	5	2%
	TD estimation	4	2%
	Qualified professionals	3	1%
	Training	3	1%
	Dashboard	2	0.8%
	Knowledge sharing	2	0.8%
	Risk analysis	2	0.8%
	Tracking the cost	2	0.8%
	Cost/benefit analysis	1	0.4%
	Improving planning	1	0.4%
	Support from project management	1	0.4%
	Team restructuring	1	0.4%
	Use of measuring reports	1	0.4%
	Using informal practices	1	0.4%
Technical	Use of tools	31	12%
	Refactoring	18	7%
	Improving tests	17	7%
	Code review	16	6%
	Adoption of agile methodology	7	3%
	Use of metrics for TD identification	7	3%
	Improve documentation	6	2%
	Identifying TD items	5	2%
	Quality validation (meetings)	4	2%
	Understanding the Cause of TD Item	3	1%
	Continuous integration	2	0.8%
	Documentation review	2	0.8%
	Improving configuration management practices	2	0.8%
	Test automation	2	0.8%
	Architecture reviewing	1	0.4%
	Continuous deployment	1	0.4%
	Identify the worst debt areas	1	0.4%
	Infrastructure Monitoring	1	0.4%
	Process automation	1	0.4%
	Pull request monitoring	1	0.4%

Caption:
#CMRP - Count of monitoring-related practices.
%MRPP - Percentage of #CMRP in relation to the total of all projects (259)

Table C.4: TD monitoring-related practices per category

Category	TD monitoring-related practice	#CMRP	%MRPP
Development issues	Improve documentation	6	2%
Infrastructure	Use of tools	31	12%
	Infrastructure Monitoring	1	0.4%
Internal quality issues	Refactoring	18	7%
	Identifying TD items	5	2%
	Understanding the Cause of TD Item	3	1%
	Identify the worst debt areas	1	0.4%
Methodology	Improving software development process	20	8%
	Improving tests	17	7%
	Code review	16	6%
	Adoption of agile methodology	7	3%
	Improving the requirement management	6	2%
	Quality validation (meetings)	4	2%
	Continuous integration	2	0.8%
	Documentation review	2	0.8%
	Improving configuration management practices	2	0.8%
	Test automation	2	0.8%
	Architecture reviewing	1	0.4%
	Continuous deployment	1	0.4%
	Process automation	1	0.4%
	Pull request monitoring	1	0.4%
	Use of measuring reports	1	0.4%
	Using informal practices	1	0.4%
Organizational	Qualified professionals	3	1%
	Training	3	1%
	Knowledge sharing	2	0.8%
	Team restructuring	1	0.4%
People	Team meetings	23	9%
	Communicating the stakeholders of TD items	16	6%
Planning and management issues	TD item backlog	34	13%
	Tracking TD items	12	5%
	TD Management Plan	11	4%
	Assign team for TD monitoring	8	3%
	Prioritization of TD items	7	3%
	TD as a task	7	3%
	Use of metrics for TD identification	7	3%
	TD status progress report	6	2%
	Focusing on TD payment	5	2%
	Measuring the effort	5	2%
	TD estimation	4	2%
	Dashboard	2	0.8%
	Risk analysis	2	0.8%
	Tracking the cost	2	0.8%
	Cost/benefit analysis	1	0.4%
	Improving planning	1	0.4%
	Support from project management	1	0.4%

Caption:
#CMRP - Count of monitoring-related practices.
%MRPP - Percentage of #CMRP in relation to the total of all projects (259)

Table C.5: PARs for TD non-monitoring and their examples of citation

PAR for TD non-monitoring	Quotes from participants
Business pressure	"There was no time for that, because of business pressure."
Changes in management	"Due to repeated changes in leadership."
Changing in the requirements	"Volatile specifications."
Company with projects beyond capacity	"More customers than the company's capacity."
Complexity of TD items	"The technical debt item was too large to monitor and the team had no resources to monitor it."
Cost	"Because the managers understand that there would be no financial gain, without seeing the maintenance costs."
Effort	"Because the effort in updating the documentation was very great."
Emotional issues of the team	"Emotional issues are not something that is generally acknowledged in the programming world let alone in business. You can't bring those types of things up even privately with the leads, you just have to hold it in."
Focusing on short term goals	"Not in the priority pipeline." "It was not critical for the success of the project."
Inaccurate time estimate	"Because the project times were extremely short and the whole team was already doing a lot of overtime."
Ineffective planning and management	"Ineffective management." "Lack of management."
Lack of confidence in the technical leader	"Lack of confidence of the technical manager of the team."
Lack of effort to know the cause of TD	"In general, they are found guilty, but it is rarely sought to understand the motivational factor of the occurrence."
Lack of experience	"The team lacks the experience in the new scheme selected to solve the problem and all the strategies result in the same problem."
Lack of interest	"Little interest of the company to correct this type of situation." "Management did not care."
Lack of knowledge on TD	"Because the concept of technical debt was not yet applied in the company." "There is no knowledge about TD."
Lack of organizational culture	"This is very dependent on the organization. In this case, it was not identified, monitored, or managed." "Because there is no permanent initiative to generate changes in the organizational culture."
Lack of qualified professionals	"Lack of people who could deal with the problem."
Lack of resources	"Even knowing the problem, there are no resources for immediate solution." "The time and resources of the project were very limited."
Lack of specific team	"It was known about for years but we didn't have the headcount to refactor." "Not enough people or time too."
Lack of TD monitoring process	"There was no process for it." "We weren't tracking it."
Lack of IT governance	"Because there is no IT governance in place, and no PMO in place."
Lack of time	"Because deadlines are tight." "The project timeline didn't allow it."
Lack of understanding about the impact of the debt	"Lack of knowledge of the impact of the TD item in question." "Technical debt was not identified as a problem at the time."
Late TD identification	"Too late identification."
Legacy system	"This project is large and has been developed over years (and is still being actively developed)."
Product delivered	"After the project is finished, refactorings will not be allowed."
Project delayed	"(. . .) all the projects were late."
Project discontinued	"Because the project simply lost its potential value and gradually became ignored." "The user team discarded the project."
React when becoming a problem	"It is neglected until it becomes a problem." "In the absence of planning, the methodology was reactionary to the problems."
TD Item eliminated as soon as identified	"We decided to resolve it soon." "It was not monitored, it was implemented."

(*table continues*)

Table C.5: PARs for TD non-monitoring and their examples of citation (continued)

PAR for TD non-monitoring	**Quotes from participants**
TD item payment do not generate revenue	“Because resolving tech debt is not a revenue-generating. Until it becomes a big enough problem to do something about.”
TD was not documented	“We know that it exists, but not documented.”
Team overload	“There was no time for experts to perform peer reviews because of the amount of work.”
Too many TD items	“Debt occurs with some frequency.”

Table C.6: PARs for TD non-monitoring per type

Type	PAR for TD non-monitoring	#CPAR	%PARP
Decision factor	Lack of interest	44	22%
	Focusing on short term goals	33	17%
	React when become a problem	5	3%
	TD Item eliminated as soon as identified	5	3%
	Ineffective planning and management	4	2%
	TD item payment do not generate revenue	2	1%
	Too many TD items	2	1%
	Lack of effort to know the cause of TD	1	0.5%
	Late TD identification	1	0.5%
	TD was not documented	1	0.5%
Impediment	Lack of time	29	15%
	Lack of knowledge on TD	23	12%
	Lack of understanding about the impact of the debt	12	6%
	Lack of organizational culture	8	4%
	Lack of resources	8	4%
	Lack of TD monitoring process	7	4%
	Lack of specific team	6	3%
	Effort	3	2%
	Product delivered	3	2%
	Changing in the requirements	2	1%
	Company with projects beyond capacity	2	1%
	Cost	2	1%
	Emotional issues of the team	2	1%
	Inaccurate time estimate	2	1%
	Lack of experience	2	1%
	Lack of qualified professionals	2	1%
	Legacy system	2	1%
	Project discontinued	2	1%
	Team overload	2	1%
	Business pressure	1	0.5%
	Changes in management	1	0.5%
	Complexity of TD items	1	0.5%
	Lack of confidence in the technical leader	1	0.5%
	Lack of IT governance	1	0.5%
	Project delayed	1	0.5%

Caption:
#CPAR - Count of citations of each PAR
%PARP - Percentage of CPAR in relation to the total of all projects (197)

Table C.7: PARs for TD non-monitoring per nature

Nature	PARs for TD non-monitoring	#CPAR	%PARP
Managerial	Lack of interest	44	22%
	Focusing on short term goals	33	17%
	Lack of time	29	15%
	Lack of organizational culture	8	4%
	Lack of resources	8	4%
	Lack of TD monitoring process	7	4%
	Lack of specific team	6	3%
	React when become a problem	5	3%
	Ineffective planning and management	4	2%
	Effort	3	2%
	Product delivered	3	2%
	Company with projects beyond capacity	2	1%
	Cost	2	1%
	Emotional issues of the team	2	1%
	Inaccurate time estimate	2	1%
	Lack of experience	2	1%
	Lack of qualified professionals	2	1%
	Project discontinued	2	1%
	TD item payment do not generate revenue	2	1%
	Team overload	2	1%
	Business pressure	1	0.5%
	Changes in management	1	0.5%
	Lack of confidence in the technical leader	1	0.5%
	Lack of effort to know the cause of TD	1	0.5%
	Lack of IT governance	1	0.5%
	Late TD identification	1	0.5%
	Project delayed	1	0.5%
Technical	Lack of knowledge on TD	23	12%
	Lack of understanding about the impact of the debt	12	6%
	TD Item eliminated as soon as identified	5	3%
	Changing in the requirements	2	1%
	Legacy system	2	1%
	Too many TD items	2	1%
	Complexity of TD items	1	0.5%
	TD was not documented	1	0.5%

Caption:
#CPAR - Count of citations of each PAR
%PARP - Percentage of CPAR in relation to the total of all projects (197)

Table C.8: PARs for TD non-monitoring per category

Category	PARs for TD non-monitoring	#CPAR	%PARP
Development issues	Changing in the requirements	2	1%
	Legacy system	2	1%
External factors	Project discontinued	2	1%
	TD item payment do not generate revenue	2	1%
	Business pressure	1	0.5%
Internal quality issues	Too many TD items	2	1%
	Complexity of TD items	1	0.5%
	Lack of effort to know the cause of TD	1	0.5%
Lack of knowledge	Lack of knowledge on TD	23	12%
Methodology	Lack of TD monitoring process	7	4%
	React when become a problem	5	3%
	TD Item eliminated as soon as identified	5	3%
	Lack of IT governance	1	0.5%
	Late TD identification	1	0.5%
	TD was not documented	1	0.5%
Organizational	Lack of interest	44	22%
	Lack of organizational culture	8	4%
	Lack of resources	8	4%
	Lack of specific team	6	3%
	Company with projects beyond capacity	2	1%
	Lack of qualified professionals	2	1%
People	Emotional issues of the team	2	1%
	Lack of experience	2	1%
	Team overload	2	1%
	Lack of confidence in the technical leader	1	0.5%
Planning and management	Focusing on short term goals	33	17%
	Lack of time	29	15%
	Lack of understanding about the impact of the debt	12	6%
	Ineffective planning and management	4	2%
	Effort	3	2%
	Product delivered	3	2%
	Cost	2	1%
	Inaccurate time estimate	2	1%
	Changes in management	1	0.5%
	Project delayed	1	0.5%

Caption:
#CPAR - Count of citations of each PAR
%PARP - Percentage of CPAR in relation to the total of all projects (197)

Table C.9: Comparison to related work on TD monitoring-related practices

Type	**Technical debt monitoring-related practices from**	
	Our Study	**Related Work**
Enabling practice	Adoption of agile methodology	Implementing pair programming or test-driven development (BEHUTIYE *et al.*, 2017)
	Assign team for TD monitoring	Defining a responsible for monitoring each identified and measured TD item (OLIVEIRA; GOLDMAN; SANTOS, 2015)
	Continuous integration	Continuous integration tools (BEHUTIYE *et al.*, 2017)
	Improving planning	-
	Improving software development process	-
	Improving the requirement management	RE-KOMBINE model (RIOS; MENDONÇA; SPINOLA, 2018)
	Infrastructure Monitoring	-
	Knowledge sharing	-
	Process automation	-
	Pull request monitoring	-
	Quality validation (meetings)	-
	Support from project management	-
	TD management plan	Accounting (AMPATZOGLOU *et al.*, 2015), Planning in advance for TD (BEHUTIYE *et al.*, 2017), Accounting (RIOS; MENDONÇA; SPINOLA, 2018), Formal approach to TD decision making (RIOS; MENDONÇA; SPINOLA, 2018)
	Team restructuring	-
	Use of tools	Using data collected from (management or TD measuring) tools (YLI-HUUMO; MAGLYAS; SMOLANDER, 2016), AnaConDebt (MARTINI, 2018), Using system for bug fixing (MARTINI; BESKER; BOSCH, 2018), Measuring symptom severity on a smell thermometer (RIOS; MENDONÇA; SPINOLA, 2018), Sonar TD plugin (RIOS; MENDONÇA; SPINOLA, 2018), DebtFlag (RIOS; MENDONÇA; SPINOLA, 2018), TD evaluation (SQALE) (RIOS; MENDONÇA; SPINOLA, 2018), Software maps tool (RIOS; MENDONÇA; SPINOLA, 2018), Code Christmas tree (RIOS; MENDONÇA; SPINOLA, 2018), VisminerTD (MENDES *et al.*, 2019), tools (APA *et al.*, 2020b), tools (APA *et al.*, 2020a), test tool (ALBUQUERQUE *et al.*, 2022), Wiki (ALBUQUERQUE *et al.*, 2022)
Monitoring action	Communicating the stakeholders of TD items	-
	Cost/benefit analysis	Cost/benefit (AMPATZOGLOU *et al.*, 2015), Cost-benefit analysis (RIOS; MENDONÇA; SPINOLA, 2018), Monitor changes in the cost/benefit ration of the identified debt (ARAGÃO *et al.*, 2022)
	Dashboard	Portfolio management (AMPATZOGLOU *et al.*, 2015), Collective dashboards (BEHUTIYE *et al.*, 2017), Portfolio approach (RIOS; MENDONÇA; SPINOLA, 2018)
	Identify the worst debt areas	Debt symptoms index (RIOS; MENDONÇA; SPINOLA, 2018)
	Measuring the effort	Real options (AMPATZOGLOU *et al.*, 2015), Marketing (AMPATZOGLOU *et al.*, 2015), Options (RIOS; MENDONÇA; SPINOLA, 2018), Marketing (RIOS; MENDONÇA; SPINOLA, 2018)
	Prioritization of TD items	-
	Risk analysis	TD monitoring as part of risk process (ERNST *et al.*, 2015)
	TD as a task	-
	TD estimation	Improving estimation techniques of sprints (BEHUTIYE *et al.*, 2017)

(*table continues*)

Table C.9: Comparison to related work on TD monitoring-related practices (continued)

Type	**Technical debt monitoring-related practices from**	
	Our Study	**Related Work**
	TD item backlog	Backlog grooming (ERNST *et al.*, 2015), including TD tasks in product backlog (BOMFIM; SANTOS, 2017), Creation of TD items in a backlog (MARTINI, 2018), Reporting TD items in backlog (MARTINI; BESKER; BOSCH, 2018), Panels (TO DO, DOING, and DONE) based on the Kanban concept (MENDES *et al.*, 2019)
	TD status progress report	Visualization techniques (BEHUTIYE *et al.*, 2017)
	Team meetings	-
	Tracking TD items	Issue tracker (ALBUQUERQUE *et al.*, 2022)
	Tracking the cost	-
	Understanding the Cause of TD Item	-
	Use of measuring reports	-
	Using informal practices	Manual monitoring (APA *et al.*, 2020a), Manually (ALBUQUERQUE *et al.*, 2022)
TD identification	Identifying TD items	Statically analyzing the code for finding TD items or potential bugs, or security issues (MARTINI; BESKER; BOSCH, 2018)
	Use of metrics for TD identification	Metrics for managing architectural TD (RIOS; MENDONÇA; SPINOLA, 2018)
TD payment	Focusing on TD payment	-
	Improve documentation	-
	Refactoring	-
TD prevention	Architecture reviewing	-
	Code review	SQALE method (RIOS; MENDONÇA; SPINOLA, 2018), Static analysis (ALBUQUERQUE *et al.*, 2022)
	Continuous deployment	-
	Documentation review	-
	Improving configuration management practices	-
	Improving tests	Measuring test coverage (MARTINI; BESKER; BOSCH, 2018), Changes in the test process (ARAGÃO *et al.*, 2022)
	Qualified professionals	-
	Test automation	-
	Training	-
-	-	Setting a commonly agreed definition of done (BEHUTIYE *et al.*, 2017)
-	-	Using comments in the code or other artifacts (MARTINI; BESKER; BOSCH, 2018)
-	-	Documenting issues in text or spreadsheets (MARTINI; BESKER; BOSCH, 2018)
-	-	Monitor triggers (ARAGÃO *et al.*, 2022)
-	-	Making of dependencies and code problems (RIOS; MENDONÇA; SPINOLA, 2018)
-	-	Supply chain management (RIOS; MENDONÇA; SPINOLA, 2018)

Appendix D

TECHNICAL DEBT PAYMENT - COMPLEMENTARY MATERIAL

Table D.1: TD payment-related practices and their examples of citation

TD payment-related practice	**Quotes from participants**
Adjusting Code to follow Good Programming Practices	"Benefits of shared library cutting down on dev. time duplicating or maintaining extra code."
Bug Fixing	"To make it solid and defect free system."
Changing Project Scope	"Redefining the scope of the project."
Changing the Project Management	"There were changes in team leaders (. . .)."
Code Refactoring	"We rewrote the offending code."
Code Reviewing	"The identified tech debt has been resolved through updated designs and refactors. There was a lot of post completion code reviewing to identified areas with performance impacts."
Communicating the customer of TD Items	"(. . .), communication of TD items to the customer, (. . .)."
Conducting Risk Analysis	"Through risk mitigation plans and action plans."
Design Refactoring	"Refactoring and changing architecture."
Hiring Specialized Professionals	"There were cases where it was necessary to hire a specialized team (. . .)."
Implementing Preventive Actions for Avoiding TD Items	"(. . .) periodic reviews were established."
Improving Requirement Elicitation Process	"The project managed to align itself with the milestones by aggressively simplifying the definition of requirements for the implementation process."
Improving the Development Process	"Improvement in the development and changes process."
Improving the Team Collaboration	"Relationship between development team and business team improves and gets better and better."
Increasing the Project Budget	"With cost overruns in money and time, updating the solution to the latest changes in technology."
Investing Effort on TD Payment Activities	"Taking sprints to pay down the debt."
Investing Effort on Testing Activities	"Additional development hours and code testing."
Monitoring and Controlling Project Activities	"With the measurement bulletins, there was the metric of progress."
Negotiating Deadline Extension	"It was possible to negotiate time with the user to make improvements."
Organizing the Project Repository	"After organizing the repository, it was possible to deliver in a flow, (. . .)."
Prioritizing TD Items	"Due to the project deadline, as soon as the TD is detected, we already plan its development for the next sprints of the project."
Restarting the Project from Scratch	"Starting the project 100% from scratch."

(*table continues*)

Table D.1: TD payment-related practices and their examples of citation (continued)

TD payment-related practice	**Quotes from participants**
Solving Technical Issues	“Updated the version of the framework.”
System Retirement	“The delay of the project was so long that it was canceled.”
Update System Documentation	“The documentation was updated.”
Using Automated Deployment	“The deployment and testing process was further automated to streamline developments.”
Using Code Analysis	“Each of the reports generated by Sonar was analyzed and the improvements that made it possible to cover the debt were applied.”
Using Code Reuse	“It was necessary to redo the code, design common functions, reuse code (. . .).”
Using Continuous Integrations and Deliveries	“Everything that is integration and continuous deployment was implemented through Gitlab.”
Using External Tools	“The proposed operations were performed by sonar.”
Using Short Feedback Iterations	“(. . .) shorter and shorter feedback cycles.”
Using Software Models	“(. . .) The project models were developed (. . .)”

Table D.2: TD payment-related practices per type

Type	TD payment-related practice	#CPRP	%PRPP
Enabling TD payment	Investing Effort on TD Payment Activities	33	15.1%
	Negotiating Deadline Extension	14	6.4%
	Improving the Development Process	9	4.1%
	Increasing the Project Budget	9	4.1%
	Hiring Specialized Professionals	8	3.7%
	Improving the Team Collaboration	6	2.8%
	Changing Project Scope	5	2.3%
	Using Continuous Integrations and Deliveries	4	1.8%
	Changing the Project Management	2	0.9%
	Conducting Risk Analysis	2	0.9%
	Communicating the Customer of TD Items	1	0.5%
Payment action	Code Refactoring	81	37.2%
	Design Refactoring	25	11.5%
	Adjusting Code to follow Good Programming Practices	10	4.6%
	Solving Technical Issues	9	4.1%
	Update System Documentation	9	4.1%
	Bug Fixing	6	2.8%
	Restarting the Project from Scratch	4	1.8%
	System Retirement	2	0.9%
TD prevention	Investing Effort on Testing Activities	22	10.1%
	Monitoring and Controlling Project Activities	10	4.6%
	Implementing Preventive Actions for Avoiding TD Items	7	3.2%
	Using Short Feedback Iterations	7	3.2%
	Code Reviewing	3	1.4%
	Improving Requirement Elicitation Process	3	1.4%
	Using Code Analysis	3	1.4%
	Using External Tools	3	1.4%
	Using Code Reuse	2	0.9%
	Using Software Models	2	0.9%
	Organizing the Project Repository	1	0.5%
	Using Automated Deployment	1	0.5%
TD prioritization	Prioritizing TD Items	15	6.9%

Caption:
#CPRP - Count of TD payment-related practices.
%PRPP - Percentage of #CPRP in relation to the total of all projects (218)

Table D.3: TD payment-related practices per nature

Nature	**TD payment-related practice**	**#CPRP**	**%PRPP**
Managerial	Investing Effort on TD Payment Activities	33	15.1%
	Investing Effort on Testing Activities	22	10.1%
	Prioritizing TD Items	15	6.9%
	Negotiating Deadline Extension	14	6.4%
	Monitoring and Controlling Project Activities	10	4.6%
	Improving the Development Process	9	4.1%
	Increasing the Project Budget	9	4.1%
	Hiring Specialized Professionals	8	3.7%
	Using Short Feedback Iterations	7	3.2%
	Improving the Team Collaboration	6	2.8%
	Changing Project Scope	5	2.3%
	Restarting the Project from Scratch	4	1.8%
	Improving Requirement Elicitation Process	3	1.4%
	Changing the Project Management	2	0.9%
	Conducting Risk Analysis	2	0.9%
	System Retirement	2	0.9%
	Communicating the Customer of TD Items	1	0.5%
Technical	Code Refactoring	81	37.2%
	Design Refactoring	25	11.5%
	Adjusting Code to follow Good Programming Practices	10	4.6%
	Solving Technical Issues	9	4.1%
	Update System Documentation	9	4.1%
	Implementing Preventive Actions for Avoiding TD Items	7	3.2%
	Bug Fixing	6	2.8%
	Using Continuous Integrations and Deliveries	4	1.8%
	Code Reviewing	3	1.4%
	Using Code Analysis	3	1.4%
	Using External Tools	3	1.4%
	Using Code Reuse	2	0.9%
	Using Software Models	2	0.9%
	Organizing the Project Repository	1	0.5%
	Using Automated Deployment	1	0.5%

Caption:
#CPRP - Count of TD payment-related practices.
%PRPP - Percentage of #CPRP in relation to the total of all projects (218)

Table D.4: TD payment-related practices per category

Category	**TD payment-related practice**	**#CPRP**	**%PRPP**
Development issues	Adjusting Code to follow Good Programming Practices	10	4.6%
	Solving Technical Issues	9	4.1%
	Update System Documentation	9	4.1%
	Changing Project Scope	5	2.3%
	Restarting the Project from Scratch	4	1.8%
	System Retirement	2	0.9%
External quality issues	Bug fixing	6	2.8%
Infrastructure	Using External Tools	3	1.4%
	Organizing the Project Repository	1	0.5%
Internal quality issues	Code Refactoring	81	37.2%
	Design Refactoring	25	11.5%
Methodology	Investing Effort on TD Payment Activities	33	15.1%
	Investing Effort on Testing Activities	22	10.1%
	Improving the Development Process	9	4.1%
	Implementing Preventive Actions for Avoiding TD Items	7	3.2%
	Using Short Feedback Iterations	7	3.2%
	Using Continuous Integrations and Deliveries	4	1.8%
	Code Reviewing	3	1.4%
	Improving Requirement Elicitation Process	3	1.4%
	Using Code Analysis	3	1.4%
	Using Code Reuse	2	0.9%
	Using Software Models	2	0.9%
	Using Automated Deployment	1	0.5%
Organizational	Hiring Specialized Professionals	8	3.7%
	Changing the Project Management	2	0.9%
People	Improving the Team Collaboration	6	2.8%
	Communicating the Customer of TD Items	1	0.5%
Planning and management	Prioritizing TD Items	15	6.9%
	Negotiating Deadline Extension	14	6.4%
	Monitoring and Controlling Project Activities	10	4.6%
	Increasing the Project Budget	9	4.1%
	Conducting Risk Analysis	2	0.9%

Caption:
#CPRP - Count of TD payment-related practices.
%PRPP - Percentage of #CPRP in relation to the total of all projects (218)

Table D.5: PARs for TD non-payment and their examples of citation

PAR for TD non-payment	**Quotes from participants**
Complexity of the Project	"It is a very large project and the interconnectedness of the data layer throughout the project makes completely rewriting it very difficult."
Complexity of the TD Item	"It is something that cannot be eliminated, the errors generated must be supported."
Cost	"It is too expensive to fix. Too big to succeed."
Customer Decision	"Because of the clients decision."
Decision to do not change the Framework	"Because we are still using the framework."
Effort	"Would take a lot of time to migrate to a new back-end database and change the application code."
Focusing on Short Term Goals	"Focus on next release."
High Team Turnover	"Staff turnover."
Insufficient Management View about TD Payment	"Resistance from the project manager."
Lack of Access on Component Code	"(. . .) Also, other teams own the code that would need to be updated."
Lack of Adoption of Lessons Learned	"Team members who dont learn from past mistakes."
Lack of Committed Team	"It depends, those who are more professional paid off, while others not."
Lack of External Auditing	"The consultancy helped identify critical points to attack and prioritize them. However, this did not prevent changes from being made to the coding that were not contemplated in the initial project estimate."
Lack of Knowledge on TD	"(. . .) The business side of the house is not technical enough to understand the impact that the TD has."
Lack of Monitoring of TD Items	"Lack of monitoring."
Lack of Organizational Interest	"We learned enough to move forward but no one wanted to spend the time."
Lack of Resources	"We did not have the technical resources, or the time required to carry out the activity."
Lack of Technical Knowledge	"Lack of commitment and lack of knowledge of the team."
Lack of Testing	"(. . .)Project deadlines prohibit a full search and fix, and there are no tests in the legacy code to ensure the changes result in no behavior change (. . .)."
Lack of Time	"No time dedicated to significant redesign and rework."
Non-Application of Mitigation Actions on TD Causes	"The root causes were not mitigated."
Number of TD Items	"There is too much. We are attacking as needed."
Risk for the Project	"They are not corrected by the high risk involved in generating application errors in the most critical locations."
TD Items do not affect the user	"Its been considered fallout and If an end user runs across the problem then it gets fixed."
TD Items do not have "interest"	"Because it is easier for the project manager to put out fires and not to avoid them."
Team overload	"Precisely because even the system repair teams are overloaded."
The Project was Discontinued	"Because the project has been abandoned completely."

Table D.6: PARs for TD non-payment per type

Type	PAR for TD non-payment	#CPAR	%PARP
Decision factor	Focusing on Short Term Goals	69	26.4%
	Lack of Organizational Interest	48	18.4%
	Insufficient Management View about TD Payment	10	3.8%
	Lack of Committed Team	4	1.5%
	TD Items do not have “interest”	4	1.5%
	Lack of Adoption of Lessons Learned	3	1.1%
	Number of TD Items	3	1.1%
	Lack of Testing	2	0.8%
	Non-Application of Mitigation Actions on TD Causes	2	0.8%
	TD Items do not affect the user	2	0.8%
	Decision to do not change the Framework	1	0.4%
	Lack of External Auditing	1	0.4%
Impediment	Lack of Time	41	15.7%
	Cost	34	13.0%
	Lack of Resources	19	7.3%
	Customer Decision	13	5.0%
	Complexity of the TD Item	12	4.6%
	Effort	11	4.2%
	Complexity of the Project	10	3.8%
	Team overload	6	2.3%
	The Project was Discontinued	6	2.3%
	Lack of Technical knowledge	5	1.9%
	Risk for the Project	5	1.9%
	Lack of Knowledge on TD	4	1.5%
	High Team Turnover	2	0.8%
	Lack of Access on Component Code	1	0.4%
	Lack of Monitoring of TD Items	1	0.4%

Caption:
#CPAR - Count of citations of each PAR
%PARP - Percentage of CPAR in relation to the total of all projects (261)

Table D.7: PARs for TD non-payment per type per nature

Nature	PARs for TD non-payment	#CPAR	%PARP
Managerial	Focusing on Short Term Goals	69	26.4%
	Lack of Organizational Interest	48	18.4%
	Lack of Time	41	15.7%
	Cost	34	13.0%
	Lack of Resources	19	7.3%
	Customer Decision	13	5.0%
	Effort	11	4.2%
	Complexity of the Project	10	3.8%
	Insufficient Management View about TD Payment	10	3.8%
	Team overload	6	2.3%
	The Project was Discontinued	6	2.3%
	Risk for the Project	5	1.9%
	Lack of Committed Team	4	1.5%
	Lack of Knowledge on TD	4	1.5%
	TD Items do not have “interest”	4	1.5%
	Lack of Adoption of Lessons Learned	3	1.1%
	Number of TD Items	3	1.1%
	High Team Turnover	2	0.8%
	Non-Application of Mitigation Actions on TD Causes	2	0.8%
	TD Items do not affect the user	2	0.8%
	Lack of External Auditing	1	0.4%
Technical	Complexity of the TD Item	12	4.6%
	Lack of Technical knowledge	5	1.9%
	Lack of Testing	2	0.8%
	Decision to do not change the Framework	1	0.4%
	Lack of Access on Component Code	1	0.4%
	Lack of Monitoring of TD Items	1	0.4%

Caption:
#CPAR - Count of citations of each PAR
%PARP - Percentage of CPAR in relation to the total of all projects (261)

Table D.8: PARs for TD non-payment per category

Category	PARs for TD non-payment	#CPAR	%PARP
Devt. issues	Complexity of the Project	10	3.8%
	Decision to do not Change the Framework	1	0.4%
External factors	Customer Decision	13	5.0%
	The Project was Discontinued	6	2.3%
	TD Items do not affect the User	2	0.8%
	Lack of Access on Component Code	1	0.4%
Internal quality issues	Complexity of the TD Item	12	4.6%
	Number of TD Items	3	1.1%
Lack of knowl.	Lack of Technical knowledge	5	1.9%
	Lack of Knowledge on TD	4	1.5%
Methodology	Lack of Adoption of Lessons Learned	3	1.1%
	Lack of Testing	2	0.8%
	Non-Application of Mitigation Actions on TD Causes	2	0.8%
	Lack of External Auditing	1	0.4%
	Lack of Monitoring of TD Items	1	0.4%
Organizational	Lack of Organizational Interest	48	18.4%
	Lack of Resources	19	7.3%
	High Team Turnover	2	0.8%
People	Insufficient Management View about TD Payment	10	3.8%
	Team overload	6	2.3%
	Lack of Committed Team	4	1.5%
Planning and management	Focusing on Short Term Goals	69	26.4%
	Lack of Time	41	15.7%
	Cost	34	13.0%
	Effort	11	4.2%
	Risk for the Project	5	1.9%
	TD Items do not have “interest”	4	1.5%

Caption:
#CPAR - Count of citations of each PAR
%PARP - Percentage of CPAR in relation to the total of all projects (261)

Table D.9: Comparison to related work on TD payment-related practices

Type	**Technical debt payment-related practices from**	
	Our Study	**Related Work**
Enabling practice	Changing project scope	-
	Changing the project management	Strategic management (DAS *et al.*, 2022), Improve coordination of decisions (ERNST; DELANGE; KAZMAN, 2021), Regular cross-team presentation on what they are working on (ERNST; DELANGE; KAZMAN, 2021), Coaching (ERNST; DELANGE; KAZMAN, 2021), Regular 1:1 interviews or daily stand-up meetings (ERNST; DELANGE; KAZMAN, 2021), Mood polling (ERNST; DELANGE; KAZMAN, 2021), Review retrospective (ERNST; DELANGE; KAZMAN, 2021),Give important tasks to junior developers and ask senior developers to mentor them (ERNST; DELANGE; KAZMAN, 2021), As team leads to set a bar where requests from priggish members are challenged... (ERNST; DELANGE; KAZMAN, 2021), Restructure and delimit role and interactions... (ERNST; DELANGE; KAZMAN, 2021), Audit and log developer access and activity... (ERNST; DELANGE; KAZMAN, 2021), Block access to any user not following organization guidelines (ERNST; DELANGE; KAZMAN, 2021), Identify champions and evangelists to technologies and processes... (ERNST; DELANGE; KAZMAN, 2021), Have regular meetings between group leads (ERNST; DELANGE; KAZMAN, 2021), Get feedback from members about protocols (ERNST; DELANGE; KAZMAN, 2021), Identify and sanction lone-wolf behavior (ERNST; DELANGE; KAZMAN, 2021), Proper planning about the technology and migration as soon as possible (de Toledo; MARTINI; SJøBERG, 2021)
	Communicating the customer of TD items	Customer feedback (CODABUX; WILLIAMS; NIU, 2014)
	Conducting risk analysis	Define and execute a governance plan to handle the migration (TOLEDO *et al.*, 2019)
	Hiring specialized professionals	-
	Improving the development process	Challenge process and/or structure (de Toledo; MARTINI; SJøBERG, 2021), Rapid prototyping of a new idea to see if it improves existing issues (de Toledo; MARTINI; SJøBERG, 2021), Give ownership of DevOps to one entity (group/person) (de Toledo; MARTINI; SJøBERG, 2021), Define processes and incentivize developers to use them (de Toledo; MARTINI; SJøBERG, 2021), Convince members with metrics about the efficiency of the new technology or process (de Toledo; MARTINI; SJøBERG, 2021), Investing in a process to evaluate and approve external dependencies as fast as possible (de Toledo; MARTINI; SJøBERG, 2021)
	Improving the team collaboration	Communication structure between business (YLI-HUUMO; MAGLYAS; SMOLANDER, 2014), Continuous collaboration with product owner (GUPTA *et al.*, 2016), Meetings (SILVA; JUNIOR; TRAVASSOS, 2019; ALBUQUERQUE *et al.*, 2022), Incentivize senior developers to have empathy for junior ones... (ERNST; DELANGE; KAZMAN, 2021), Create more diverse teams (ERNST; DELANGE; KAZMAN, 2021), Show priggish members the productivity wasted addressing issues with little consequence (ERNST; DELANGE; KAZMAN, 2021), Abandon/avoid some communication channels (ERNST; DELANGE; KAZMAN, 2021), Organize periodic group meetings between silos (ERNST; DELANGE; KAZMAN, 2021), Communicate information via protocols that reach appropriate members of the organization (ERNST; DELANGE; KAZMAN, 2021), Pair lone wolves with other members (ERNST; DELANGE; KAZMAN, 2021)
	Increasing the project budget	Allocating more budget (ABAD *et al.*, 2016)

(*table continues*)

Table D.9: Comparison to related work on TD payment-related practices (continued)

Type	Technical debt payment-related practices from	
	Our Study	**Related Work**
Payment action	Investing effort on TD payment activities	Ampatzoglou *et al.* (2015): Real options, Cost/benefit, Portfolio management, Value-based, ROI or net present value; Allocating more infrastructure (ABAD *et al.*, 2016), Continuous identification, prioritization, and resolving identified technical debt (GUPTA *et al.*, 2016), Allow engineers to focus 100% of their time to address technical debt and improve overall code (ERNST; DELANGE; KAZMAN, 2021), Usually pays debt in the next sprint (BONFIM; BENITTI, 2022)
	Negotiating deadline extension	Allocating more time (ABAD *et al.*, 2016), Having more flexible schedules (ABAD *et al.*, 2016)
	Using continuous integrations and deliveries	-
	Adjusting code to follow good programming practices	Coding standards and guides (YLI-HUUMO; MAGLYAS; SMOLANDER, 2014), Using design patterns (ABAD *et al.*, 2016), Pair programming (SAMARTHYAM; MURALIDHARAN; ANNA, 2017), Clean coding (SAMARTHYAM; MURALIDHARAN; ANNA, 2017), Using design patterns (BOMFIM; SANTOS, 2017), Toledo *et al.* (2019): Remove the business logic inside the communication layer, Move the business logic to the services, Define a canonical model per domain, Centralize the source code and documentation for all services in a common management system, Provide a common middleware that can be used by all services, Maintain the system working with different solutions during the migration period; Bogner, Verdecchia and Gerostathopoulos (2021): Manage model configuration, Use clear interfaces and APIs, Monitor deployed models, Feature selection, Training data, Data dependencies, Transparency, Data quality, Production, Components, Data validation, Data linting, Data governance; Unify the code base (ERNST; DELANGE; KAZMAN, 2021), de Toledo, Martini and Sjøberg (2021): Add services ownership metadata to the messages (. . .), Implementation of a Canonical Data Model that (. . .), Removal of the dead letter queue (. . .), Add metadata to identify the source of the messages, Splitting the dead letter queue (. . .), Use some time to design generic and independent services, Use of an API-first approach while designing services, Considering slot for continuous API (. . .), Ensure standardization with a Canonical Data Model, Additional effort to stabilize the API (. . .), Management of API versions, Tracking of internal and external consumers, Definition of clear deprecation strategy, Definition of a standard for the APIs, Redesign of services (. . .), Limiting the set of technologies used by the teams, use of language specific mono-repositories and incentive (. . .), Use of a service mesh, Reduction in the use of shared libraries, Replication of simple code (. . .), Creation of repository for configuration settings, Reducing of the amount of configuration settings on services, Requirement of peer approval (. . .), Creation of a configuration server (. . .)
	Bug fixing	Bug fixing days (YLI-HUUMO; MAGLYAS; SMOLANDER, 2014), Bug fixing (FU *et al.*, 2022)

(*table continues*)

Table D.9: Comparison to related work on TD payment-related practices (continued)

Type	**Technical debt payment-related practices from**	
	Our Study	**Related Work**
	Code refactoring	Refactoring (YLI-HUUMO; MAGLYAS; SMOLANDER, 2014; ABAD *et al.*, 2016; YLI-HUUMO; MAGLYAS; SMOLANDER, 2016; SAMARTHYAM; MURALIDHARAN; ANNA, 2017; SILVA; JUNIOR; TRAVASSOS, 2019; APA *et al.*, 2020b; APA *et al.*, 2020a; ERNST; DELANGE; KAZMAN, 2021; ALBUQUERQUE *et al.*, 2022; FU *et al.*, 2022; BOGNER; VERDECCHIA; GEROSTATHOPOULOS, 2021), Rewriting (YLI-HUUMO; MAGLYAS; SMOLANDER, 2016; SILVA; JUNIOR; TRAVASSOS, 2019; APA *et al.*, 2020b; APA *et al.*, 2020a; ALBUQUERQUE *et al.*, 2022), Splitting methods to make them more reusable (BOMFIM; SANTOS, 2017), Refactoring older code (BOMFIM; SANTOS, 2017), Remove unnecessary features (BOGNER; VERDECCHIA; GEROSTATHOPOULOS, 2021)
	Design refactoring	Improving design and architecture (ABAD *et al.*, 2016), Redesigning (YLI-HUUMO; MAGLYAS; SMOLANDER, 2016; SILVA; JUNIOR; TRAVASSOS, 2019; APA *et al.*, 2020b; APA *et al.*, 2020a), Partial refactoring for architectural TD (RIOS; MENDONÇA; SPINOLA, 2018), Rewrite the communication layer(TOLEDO *et al.*, 2019), Migrate the services to use the new architecture(TOLEDO *et al.*, 2019), Rewrite services to use the same middleware(TOLEDO *et al.*, 2019), Update the services to use the newly defined canonical models(TOLEDO *et al.*, 2019), Fixing the antipatterns (ERNST; DELANGE; KAZMAN, 2021), Improve architecture decision communication (ERNST; DELANGE; KAZMAN, 2021), Moving such business logic to the services, keeping the communication layer as thin as possible (de Toledo; MARTINI; SJøBERG, 2021), Architectural redesign (ALBUQUERQUE *et al.*, 2022)
	Restarting the project from scratch	-
	Solving technical issues	Managing TD in database schemas (RIOS; MENDONÇA; SPINOLA, 2018), Having separated databases for each service (de Toledo; MARTINI; SJøBERG, 2021), Creation of distinct database schemes for each service inside the same database (de Toledo; MARTINI; SJøBERG, 2021), Wrapping of the database within a service, preventing direct access (de Toledo; MARTINI; SJøBERG, 2021), The solution is context dependent, depending on the problem, a shared database might be needed, or a more complex transaction mechanism must be implemented (de Toledo; MARTINI; SJøBERG, 2021)
	System retirement	-
	Update system documentation	Once the neglected users need is identified, it is formalized and included in the software requirements specification document (LENARDUZZI; FUCCI, 2019), Keep the documentation updated (RIOS; MENDONÇA; SPINOLA, 2018), Review outdated documentation (RIOS; MENDONÇA; SPINOLA, 2018), Living documents (ERNST; DELANGE; KAZMAN, 2021), Documentation improvement (FU *et al.*, 2022)
TD prevention	Code reviewing	Code reviews (CODABUX; WILLIAMS; NIU, 2014; YLI-HUUMO; MAGLYAS; SMOLANDER, 2014)
	Implementing preventive actions for avoiding TD items	-
	Improving requirement elicitation process	New requirements should be down prioritized (TOLEDO *et al.*, 2019)

(*table continues*)

Table D.9: Comparison to related work on TD payment-related practices (continued)

Type	Technical debt payment-related practices from	
	Our Study	Related Work
	Investing effort on testing activities	Automating unit tests (CODABUX; WILLIAMS; NIU, 2014), Elaborate and perform tests for releases that were not tested (ARAGÃO *et al.*, 2022), Change test cases by analyzing defects (ARAGÃO *et al.*, 2022), Tests (BOGNER; VERDECCHIA; GEROSTATHOPOULOS, 2021), Ask any tester to break the product and demonstrate that part of the product was not complete (ERNST; DELANGE; KAZMAN, 2021), Testing improvement (FU *et al.*, 2022)
	Monitoring and controlling project activities	Visualizing debt with Information radiators (GUPTA *et al.*, 2016), Common product backlog (GUPTA *et al.*, 2016)
	Organizing the project repository	-
	Using automated deployment	-
	Using code analysis	-
	Using code reuse	-
	Using external tools	Using automated tools (DAS *et al.*, 2022), Use of third-party products (e.g., circuit breakers) that provide such mechanisms (de Toledo; MARTINI; SJøBERG, 2021)
	Using short feedback iterations	-
	Using software models	-
TD prioritization	Prioritizing TD items	Prioritizing TD (ABAD *et al.*, 2016), Continuous identification, prioritization, and resolving identified technical debt (GUPTA *et al.*, 2016), Adopt TD payment prioritization criteria (RIOS; MENDONÇA; SPINOLA, 2018)
-	-	Quantifying TD (ABAD *et al.*, 2016)
-	-	Using palliative solutions (BOMFIM; SANTOS, 2017)
-	-	Service developers must learn new technologies (TOLEDO *et al.*, 2019), workshops/ training (SILVA; JUNIOR; TRAVASSOS, 2019), Training and developer education programs (ERNST; DELANGE; KAZMAN, 2021), Invited lectures to introduce developers and managers to new techniques (ERNST; DELANGE; KAZMAN, 2021), Personal project time to incentivize developers to learn new techniques (ERNST; DELANGE; KAZMAN, 2021), Hold workshops and presentations (ERNST; DELANGE; KAZMAN, 2021), Internal training about API development (de Toledo; MARTINI; SJøBERG, 2021)
-	-	Internal debt stories (GUPTA *et al.*, 2016)
-	-	As for code smells, refactoring is needs to be applied to pay back requirements smells (LENARDUZZI; FUCCI, 2019)
-	-	The implementation of the best new solution matching the updated software requirements specification document (LENARDUZZI; FUCCI, 2019)
-	-	Getting a better understanding of technical debt (DAS *et al.*, 2022)
-	-	Artifact change (ALBUQUERQUE *et al.*, 2022)
-	-	Assessing impact of TD (BONFIM; BENITTI, 2022)
-	-	Code change abandonment (FU *et al.*, 2022)
-	-	Asking experts (BOGNER; VERDECCHIA; GEROSTATHOPOULOS, 2021)

(*table continues*)

Table D.9: Comparison to related work on TD payment-related practices (continued)

Type	**Technical debt payment-related practices from**	
	Our Study	**Related Work**
-	-	Have a clear map of communication flow in the organization (ERNST; DELANGE; KAZMAN, 2021), Have a clear map of communication flow in the organization (ERNST; DELANGE; KAZMAN, 2021), Use professional communication intermediaries (ERNST; DELANGE; KAZMAN, 2021)
-	-	Have social time (ERNST; DELANGE; KAZMAN, 2021)
-	-	Outsource ontology alignment efforts to data governance companies (ERNST; DELANGE; KAZMAN, 2021)
-	-	Get data from other organizations and check if your decisions make sense (ERNST; DELANGE; KAZMAN, 2021)
-	-	Consider mitigations for power distance and cognitive distance (ERNST; DELANGE; KAZMAN, 2021)

Appendix E

FOLLOW-UP SURVEY

Perceptions on Technical Debt Payment Landscape

Dear survey participant,

Thank you for agreeing to participate in our study! This study is a follow up to the previous survey performed in the context of the InsighTD project (http://www.td-survey.com/), and it is part of ongoing efforts to report some of its results in the Journal of Systems and Software. Its goal is to investigate the software practitioners' perception on the technical debt conceptual model and the technical debt payment map.

The survey will take about 20-25 minutes to complete. It will be completely voluntary. Please, be assured that the survey follows a high academic standard and is conducted anonymously. We will not associate your email address with your answers and exclusively use them to give you information on results we obtained from this study if you want.

By clicking on the "NEXT" button, you are consenting to participate in this research. Consent also indicates your agreement that all information collected from you individually may be used by current and future researchers in such a fashion that your personal identity will be protected. Such use will include presentations at scientific or professional meetings, publishing in scientific journals, sharing anonymous information with other researchers for checking the accuracy of study findings and for future approved research that has the potential for improving knowledge.

Thank you! For further information, please contact:

Sávio Freire (savio.freire@ifce.edu.br) - PhD candidate at Federal University of Bahia
Dr. Manoel Mendonça (manoel.mendonca@ufba.br) - Federal University of Bahia
Dr. Clemente Izurieta (clemente.izurieta@montana.edu) - Montana State University
Dr. Carolyn Seaman (cseaman@umbc.edu) - University of Maryland Baltimore County
Dr. Rodrigo Spínola (rodrigo.spinola@unifacs.br) - Salvador University

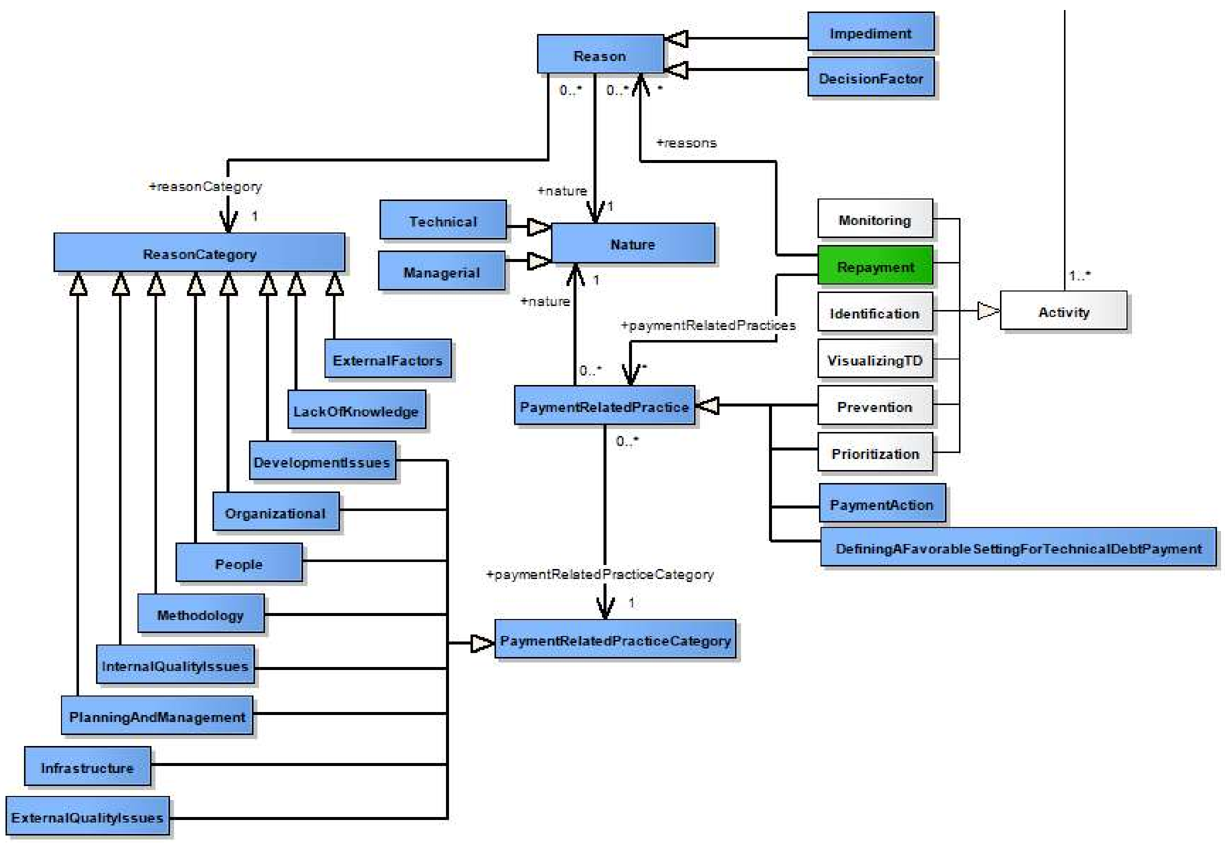
* Required

1. Please provide your email address. *

Perceptions on Technical Debt Conceptual Model	Now, we will evaluate the TD Conceptual Model, which represents the concept of TD as a whole. Before answering the questions, please watch the video presenting the TD Conceptual Model available at https://www.youtube.com/watch?v=h9Tlx3ZxuRM. Also, please download the image of the technical debt conceptual model in high resolution and the transcripts of the video at https://bit.ly/3P0xB9S.

Perceptions on Technical Debt Conceptual Model

Please, only consider the classes and relationships related to TD payment (classes in green and blue) presented in the model to answer the questions. Use zoom to see details.



2. Q1. After watching the video, what changed in your perception about TD payment? For example, are there concepts or ideas that you did not know about before, or that you now think about differently, or that you now understand more completely? *

3. Q2. What have you learned about TD payment from the model? *

4. Q3. Do you disagree with any concept or relationship presented in the conceptual model on TD payment (classes in blue)? *

   Mark only one oval.

   ⬭ Yes

   ⬭ No

5. Q4. If yes, how? If not, why? *

6. Q5. In your opinion, and considering a 0-10 scale, with 0 representing very low accuracy and 10 representing high accuracy, do you think the conceptual model is accurate to represent the TD payment concepts? *

Mark only one oval.

	Very low accuracy
0	◯
1	◯
2	◯
3	◯
4	◯
5	◯
6	◯
7	◯
8	◯
9	◯
10	◯
	High accuracy

7. Q6. In your opinion, and considering a 0-10 scale, with 0 representing very low completeness and 10 representing high completeness, do you think the conceptual model is complete to represent the TD payment concepts? *

Mark only one oval.

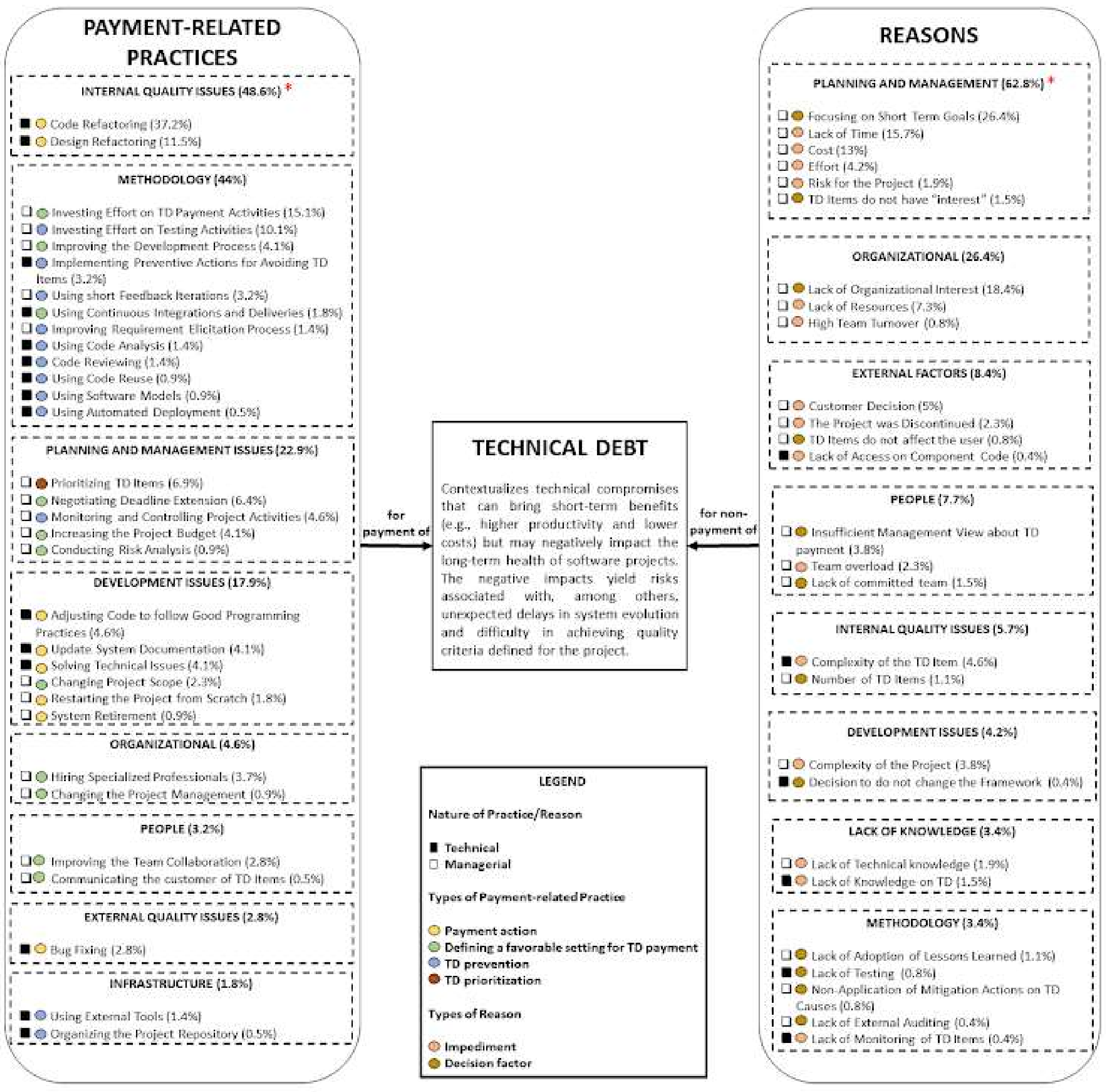
	Very low completeness
0	◯
1	◯
2	◯
3	◯
4	◯
5	◯
6	◯
7	◯
8	◯
9	◯
10	◯
	High completeness

Perceptions on Technical Debt Payment Map	Now, we will evaluate the TD Payment Map. This map is focused on representing the TD payment and its payment practices and reasons for non-payment. Before answering the questions, please watch the video presenting the TD Payment Map available at https://youtu.be/OL5dCSPpXWM. Also, please download the image of the technical debt payment map in high resolution and the transcripts of the video at https://bit.ly/3uhoE4k.

Perceptions on Technical Debt Payment Map

Here, we have the technical debt payment map presented in the video. Use zoom to see details.

TECHNICAL DEBT PAYMENT MAP



PAYMENT-RELATED PRACTICES

INTERNAL QUALITY ISSUES (48.6%) *

- ■ 🟡 Code Refactoring (37.2%)
- ■ 🟡 Design Refactoring (11.5%)

METHODOLOGY (44%)

- □ 🟢 Investing Effort on TD Payment Activities (15.1%)
- □ 🔵 Investing Effort on Testing Activities (10.1%)
- □ 🟢 Improving the Development Process (4.1%)
- ■ 🔵 Implementing Preventive Actions for Avoiding TD Items (3.2%)
- □ 🔵 Using short Feedback Iterations (3.2%)
- ■ 🟢 Using Continuous Integrations and Deliveries (1.8%)
- □ 🔵 Improving Requirement Elicitation Process (1.4%)
- ■ 🔵 Using Code Analysis (1.4%)
- ■ 🔵 Code Reviewing (1.4%)
- ■ 🔵 Using Code Reuse (0.9%)
- ■ 🔵 Using Software Models (0.9%)
- ■ 🔵 Using Automated Deployment (0.5%)

PLANNING AND MANAGEMENT ISSUES (22.9%)

- □ 🔴 Prioritizing TD Items (6.9%)
- □ 🟢 Negotiating Deadline Extension (6.4%)
- □ 🔵 Monitoring and Controlling Project Activities (4.6%)
- □ 🟢 Increasing the Project Budget (4.1%)
- □ 🟢 Conducting Risk Analysis (0.9%)

DEVELOPMENT ISSUES (17.9%)

- ■ 🟡 Adjusting Code to follow Good Programming Practices (4.6%)
- ■ 🟡 Update System Documentation (4.1%)
- ■ 🟡 Solving Technical Issues (4.1%)
- □ 🟢 Changing Project Scope (2.3%)
- □ 🟡 Restarting the Project from Scratch (1.8%)
- □ 🟡 System Retirement (0.9%)

ORGANIZATIONAL (4.6%)

- □ 🟢 Hiring Specialized Professionals (3.7%)
- □ 🟢 Changing the Project Management (0.9%)

PEOPLE (3.2%)

- □ 🟢 Improving the Team Collaboration (2.8%)
- □ 🟢 Communicating the customer of TD Items (0.5%)

EXTERNAL QUALITY ISSUES (2.8%)

- ■ 🟡 Bug Fixing (2.8%)

INFRASTRUCTURE (1.8%)

- ■ 🔵 Using External Tools (1.4%)
- ■ 🔵 Organizing the Project Repository (0.5%)

for payment of

TECHNICAL DEBT

Contextualizes technical compromises that can bring short-term benefits (e.g., higher productivity and lower costs) but may negatively impact the long-term health of software projects. The negative impacts yield risks associated with, among others, unexpected delays in system evolution and difficulty in achieving quality criteria defined for the project.

for non-payment of

REASONS

PLANNING AND MANAGEMENT (62.8%) *

- □ 🟤 Focusing on Short Term Goals (26.4%)
- □ 🟠 Lack of Time (15.7%)
- □ 🟠 Cost (13%)
- □ 🟠 Effort (4.2%)
- □ 🟠 Risk for the Project (1.9%)
- □ 🟤 TD Items do not have "interest" (1.5%)

ORGANIZATIONAL (26.4%)

- □ 🟤 Lack of Organizational Interest (18.4%)
- □ 🟠 Lack of Resources (7.3%)
- □ 🟠 High Team Turnover (0.8%)

EXTERNAL FACTORS (8.4%)

- □ 🟠 Customer Decision (5%)
- □ 🟠 The Project was Discontinued (2.3%)
- □ 🟤 TD Items do not affect the user (0.8%)
- ■ 🟠 Lack of Access on Component Code (0.4%)

PEOPLE (7.7%)

- □ 🟤 Insufficient Management View about TD payment (3.8%)
- □ 🟠 Team overload (2.3%)
- □ 🟤 Lack of committed team (1.5%)

INTERNAL QUALITY ISSUES (5.7%)

- ■ 🟠 Complexity of the TD Item (4.6%)
- □ 🟤 Number of TD Items (1.1%)

DEVELOPMENT ISSUES (4.2%)

- □ 🟠 Complexity of the Project (3.8%)
- ■ 🟤 Decision to do not change the Framework (0.4%)

LACK OF KNOWLEDGE (3.4%)

- □ 🟠 Lack of Technical knowledge (1.9%)
- ■ 🟠 Lack of Knowledge on TD (1.5%)

METHODOLOGY (3.4%)

- □ 🟤 Lack of Adoption of Lessons Learned (1.1%)
- ■ 🟤 Lack of Testing (0.8%)
- □ 🟤 Non-Application of Mitigation Actions on TD Causes (0.8%)
- □ 🟤 Lack of External Auditing (0.4%)
- ■ 🟠 Lack of Monitoring of TD Items (0.4%)

LEGEND

Nature of Practice/Reason

- ■ **Technical**
- □ **Managerial**

Types of Payment-related Practice

- 🟡 **Payment action**
- 🟢 **Defining a favorable setting for TD payment**
- 🔵 **TD prevention**
- 🔴 **TD prioritization**

Types of Reason

- 🟠 **Impediment**
- 🟤 **Decision factor**

* The number in parentheses represents the percentage of the number of citations of each category or payment-related practice or reason over all projects.

8. Q7. Would our results, as captured in this map, influence any of your decisions about HOW to pay TD items? *

Mark only one oval.

◯ Yes

◯ No

9. Q8. If yes, how? If not, why? *

10. Q9. Would our results, as captured in this map, influence any of your decisions about WHEN to pay TD items? *

Mark only one oval.

- ⬭ Yes
- ⬭ No

11. Q10. If yes, how? If no, why? *

12. Q11. Concerning the map, choose the option that best represents your opinion. *

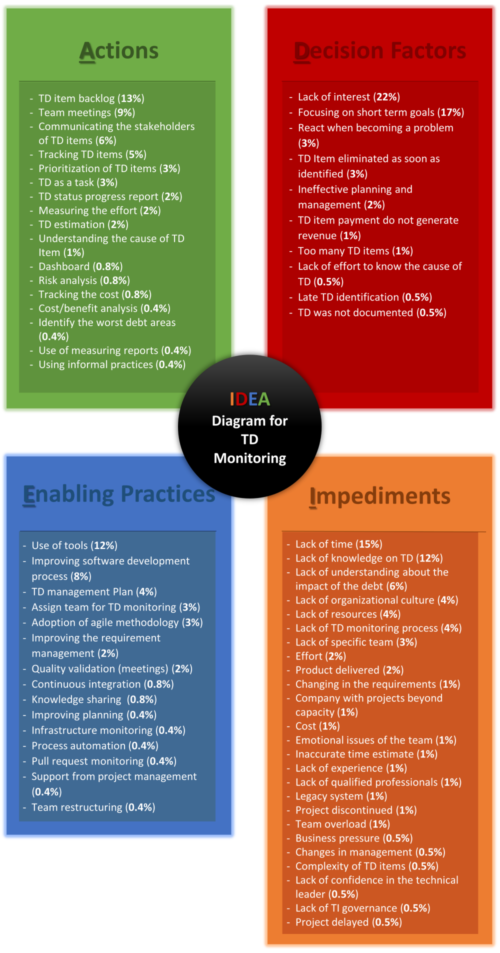
Mark only one oval per row.

	Strongly agree	Agree	Neutral	Disagree	Strongly disagree
I find the TD payment MAP easy to read, follow and use to support decisions about TD payment. (Useful and Controllable).	⬭	⬭	⬭	⬭	⬭
I find the TD payment MAP easy to read, follow and use to support decisions about TD non-payment. (Useful and Controllable).	⬭	⬭	⬭	⬭	⬭
Assuming the payment map were available at my job, I predict that I would use it on a regular basis in the future.	⬭	⬭	⬭	⬭	⬭

Finished! Thank you so much!	Sávio Freire (savio.freire@ifce.edu.br) - PhD candidate at Federal University of Bahia Dr. Manoel Mendonça (manoel.mendonca@ufba.br) - Federal University of Bahia Dr. Clemente Izurieta (clemente.izurieta@montana.edu) - Montana State University Dr. Carolyn Seaman (cseaman@umbc.edu) - University of Maryland Baltimore County Dr. Rodrigo Spínola (rodrigo.spinola@unifacs.br) - Salvador University

Appendix
F

SPECIALIZATIONS OF IDEA DIAGRAMS



Actions

- TD item backlog **(13%)**
- Team meetings **(9%)**
- Communicating the stakeholders of TD items **(6%)**
- Tracking TD items **(5%)**
- Prioritization of TD items **(3%)**
- TD as a task **(3%)**
- TD status progress report **(2%)**
- Measuring the effort **(2%)**
- TD estimation **(2%)**
- Understanding the cause of TD Item **(1%)**
- Dashboard **(0.8%)**
- Risk analysis **(0.8%)**
- Tracking the cost **(0.8%)**
- Cost/benefit analysis **(0.4%)**
- Identify the worst debt areas **(0.4%)**
- Use of measuring reports **(0.4%)**
- Using informal practices **(0.4%)**

Decision Factors

- Lack of interest **(22%)**
- Focusing on short term goals **(17%)**
- React when becoming a problem **(3%)**
- TD Item eliminated as soon as identified **(3%)**
- Ineffective planning and management **(2%)**
- TD item payment do not generate revenue **(1%)**
- Too many TD items **(1%)**
- Lack of effort to know the cause of TD **(0.5%)**
- Late TD identification **(0.5%)**
- TD was not documented **(0.5%)**

Enabling Practices

- Use of tools **(12%)**
- Improving software development process **(8%)**
- TD management Plan **(4%)**
- Assign team for TD monitoring **(3%)**
- Adoption of agile methodology **(3%)**
- Improving the requirement management **(2%)**
- Quality validation (meetings) **(2%)**
- Continuous integration **(0.8%)**
- Knowledge sharing **(0.8%)**
- Improving planning **(0.4%)**
- Infrastructure monitoring **(0.4%)**
- Process automation **(0.4%)**
- Pull request monitoring **(0.4%)**
- Support from project management **(0.4%)**
- Team restructuring **(0.4%)**

Impediments

- Lack of time **(15%)**
- Lack of knowledge on TD **(12%)**
- Lack of understanding about the impact of the debt **(6%)**
- Lack of organizational culture **(4%)**
- Lack of resources **(4%)**
- Lack of TD monitoring process **(4%)**
- Lack of specific team **(3%)**
- Effort **(2%)**
- Product delivered **(2%)**
- Changing in the requirements **(1%)**
- Company with projects beyond capacity **(1%)**
- Cost **(1%)**
- Emotional issues of the team **(1%)**
- Inaccurate time estimate **(1%)**
- Lack of experience **(1%)**
- Lack of qualified professionals **(1%)**
- Legacy system **(1%)**
- Project discontinued **(1%)**
- Team overload **(1%)**
- Business pressure **(0.5%)**
- Changes in management **(0.5%)**
- Complexity of TD items **(0.5%)**
- Lack of confidence in the technical leader **(0.5%)**
- Lack of TI governance **(0.5%)**
- Project delayed **(0.5%)**

Figure F.1 IDEA diagram for TD monitoring

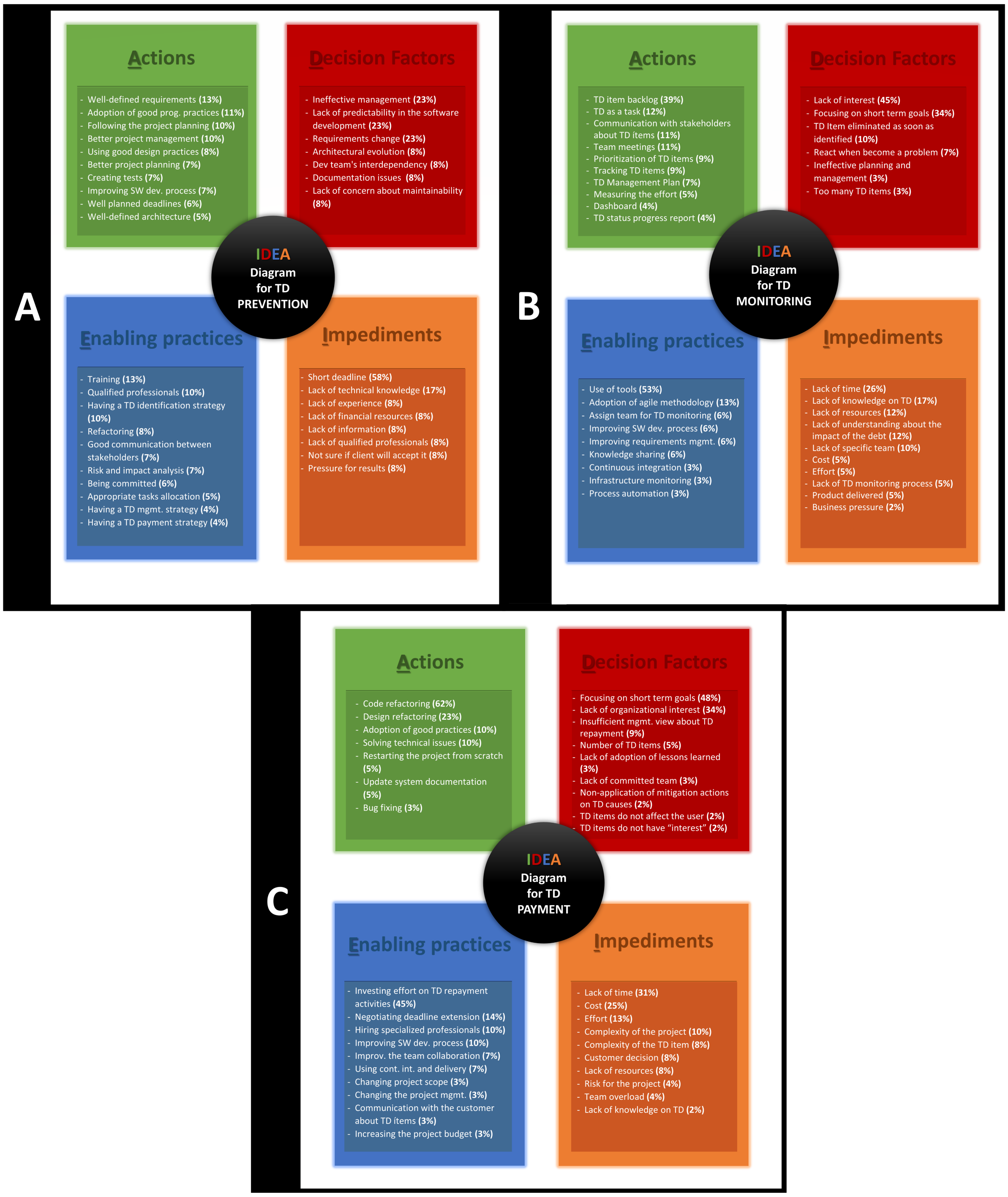

A
Actions
- Well-defined requirements (13%)
- Adoption of good prog. practices (11%)
- Following the project planning (10%)
- Better project management (10%)
- Using good design practices (8%)
- Better project planning (7%)
- Creating tests (7%)
- Improving SW dev. process (7%)
- Well planned deadlines (6%)
- Well-defined architecture (5%)
Decision Factors
- Ineffective management (23%)
- Lack of predictability in the software development (23%)
- Requirements change (23%)
- Architectural evolution (8%)
- Dev team's interdependency (8%)
- Documentation issues (8%)
- Lack of concern about maintainability (8%)
IDEA Diagram for TD PREVENTION
Enabling practices
- Training (13%)
- Qualified professionals (10%)
- Having a TD identification strategy (10%)
- Refactoring (8%)
- Good communication between stakeholders (7%)
- Risk and impact analysis (7%)
- Being committed (6%)
- Appropriate tasks allocation (5%)
- Having a TD mgmt. strategy (4%)
- Having a TD payment strategy (4%)
Impediments
- Short deadline (58%)
- Lack of technical knowledge (17%)
- Lack of experience (8%)
- Lack of financial resources (8%)
- Lack of information (8%)
- Lack of qualified professionals (8%)
- Not sure if client will accept it (8%)
- Pressure for results (8%)
B
Actions
- TD item backlog (39%)
- TD as a task (12%)
- Communication with stakeholders about TD ítems (11%)
- Team meetings (11%)
- Prioritization of TD items (9%)
- Tracking TD items (9%)
- TD Management Plan (7%)
- Measuring the effort (5%)
- Dashboard (4%)
- TD status progress report (4%)
Decision Factors
- Lack of interest (45%)
- Focusing on short term goals (34%)
- TD Item eliminated as soon as identified (10%)
- React when become a problem (7%)
- Ineffective planning and management (3%)
- Too many TD items (3%)
IDEA Diagram for TD MONITORING
Enabling practices
- Use of tools (53%)
- Adoption of agile methodology (13%)
- Assign team for TD monitoring (6%)
- Improving SW dev. process (6%)
- Improving requirements mgmt. (6%)
- Knowledge sharing (6%)
- Continuous integration (3%)
- Infrastructure monitoring (3%)
- Process automation (3%)
Impediments
- Lack of time (26%)
- Lack of knowledge on TD (17%)
- Lack of resources (12%)
- Lack of understanding about the impact of the debt (12%)
- Lack of specific team (10%)
- Cost (5%)
- Effort (5%)
- Lack of TD monitoring process (5%)
- Product delivered (5%)
- Business pressure (2%)
C
Actions
- Code refactoring (62%)
- Design refactoring (23%)
- Adoption of good practices (10%)
- Solving technical issues (10%)
- Restarting the project from scratch (5%)
- Update system documentation (5%)
- Bug fixing (3%)
Decision Factors
- Focusing on short term goals (48%)
- Lack of organizational interest (34%)
- Insufficient mgmt. view about TD repayment (9%)
- Number of TD items (5%)
- Lack of adoption of lessons learned (3%)
- Lack of committed team (3%)
- Non-application of mitigation actions on TD causes (2%)
- TD items do not affect the user (2%)
- TD items do not have "interest" (2%)
IDEA Diagram for TD PAYMENT
Enabling practices
- Investing effort on TD repayment activities (45%)
- Negotiating deadline extension (14%)
- Hiring specialized professionals (10%)
- Improving SW dev. process (10%)
- Improv. the team collaboration (7%)
- Using cont. int. and delivery (7%)
- Changing project scope (3%)
- Changing the project mgmt. (3%)
- Communication with the customer about TD ítems (3%)
- Increasing the project budget (3%)
Impediments
- Lack of time (31%)
- Cost (25%)
- Effort (13%)
- Complexity of the project (10%)
- Complexity of the TD item (8%)
- Customer decision (8%)
- Lack of resources (8%)
- Risk for the project (4%)
- Team overload (4%)
- Lack of knowledge on TD (2%)

Figure F.2 IDEA diagram for TD prevention (a), monitoring (b), and payment (c) in agile processes

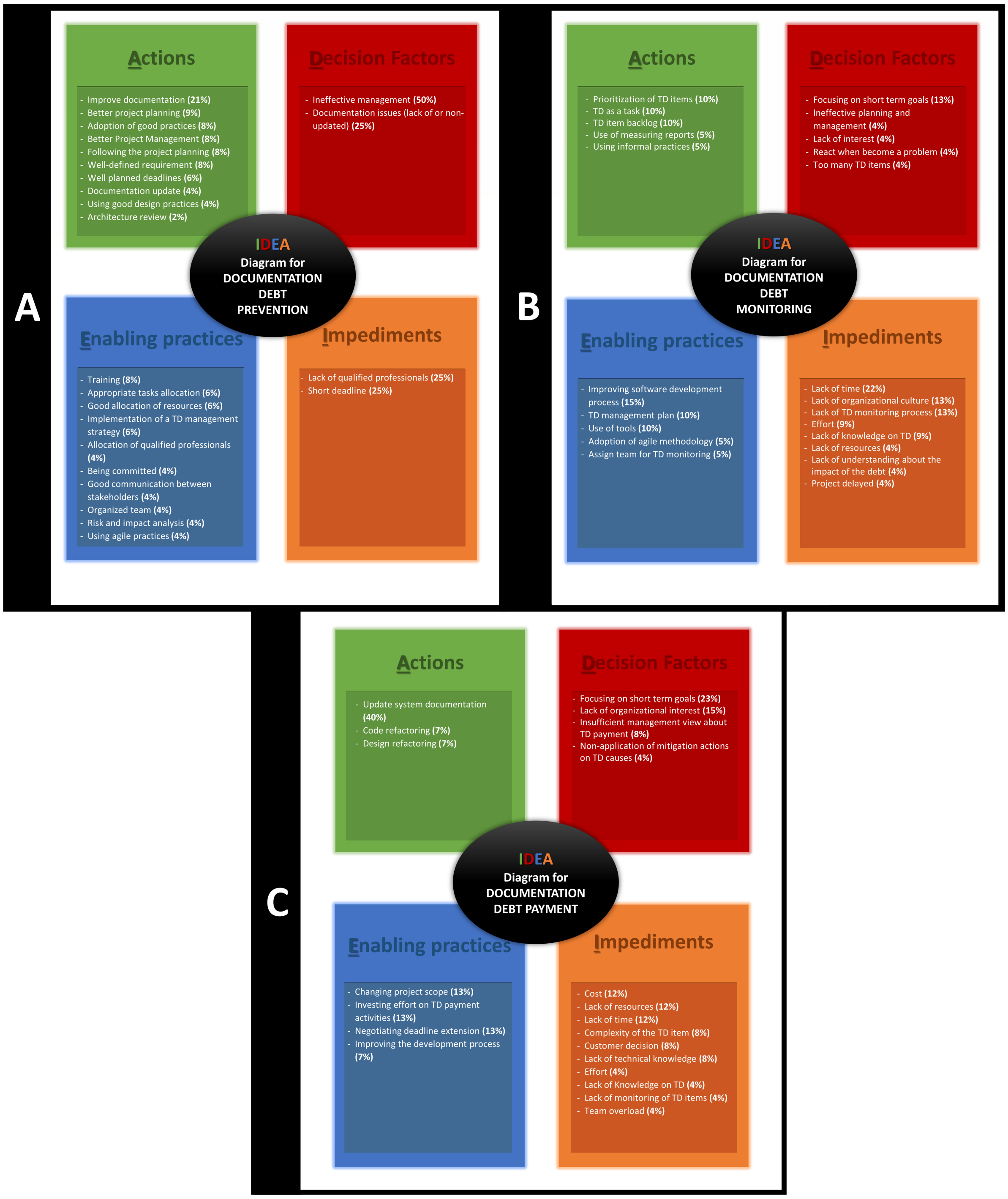
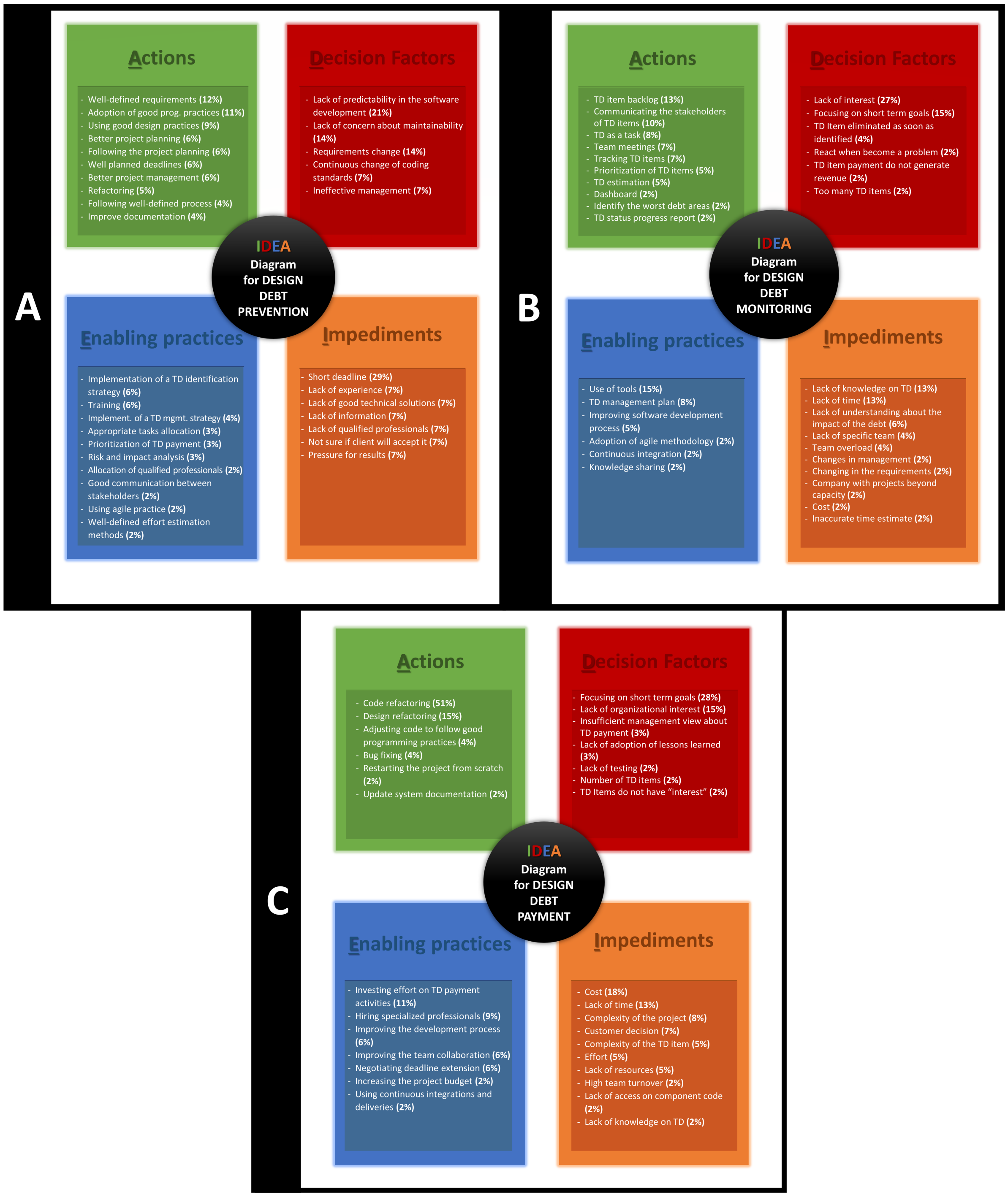

A
Actions
- Improve documentation (21%)
- Better project planning (9%)
- Adoption of good practices (8%)
- Better Project Management (8%)
- Following the project planning (8%)
- Well-defined requirement (8%)
- Well planned deadlines (6%)
- Documentation update (4%)
- Using good design practices (4%)
- Architecture review (2%)
Decision Factors
- Ineffective management (50%)
- Documentation issues (lack of or non-updated) (25%)
IDEA
Diagram for DOCUMENTATION DEBT PREVENTION
Enabling practices
- Training (8%)
- Appropriate tasks allocation (6%)
- Good allocation of resources (6%)
- Implementation of a TD management strategy (6%)
- Allocation of qualified professionals (4%)
- Being committed (4%)
- Good communication between stakeholders (4%)
- Organized team (4%)
- Risk and impact analysis (4%)
- Using agile practices (4%)
Impediments
- Lack of qualified professionals (25%)
- Short deadline (25%)
B
Actions
- Prioritization of TD items (10%)
- TD as a task (10%)
- TD item backlog (10%)
- Use of measuring reports (5%)
- Using informal practices (5%)
Decision Factors
- Focusing on short term goals (13%)
- Ineffective planning and management (4%)
- Lack of interest (4%)
- React when become a problem (4%)
- Too many TD items (4%)
IDEA
Diagram for DOCUMENTATION DEBT MONITORING
Enabling practices
- Improving software development process (15%)
- TD management plan (10%)
- Use of tools (10%)
- Adoption of agile methodology (5%)
- Assign team for TD monitoring (5%)
Impediments
- Lack of time (22%)
- Lack of organizational culture (13%)
- Lack of TD monitoring process (13%)
- Effort (9%)
- Lack of knowledge on TD (9%)
- Lack of resources (4%)
- Lack of understanding about the impact of the debt (4%)
- Project delayed (4%)
C
Actions
- Update system documentation (40%)
- Code refactoring (7%)
- Design refactoring (7%)
Decision Factors
- Focusing on short term goals (23%)
- Lack of organizational interest (15%)
- Insufficient management view about TD payment (8%)
- Non-application of mitigation actions on TD causes (4%)
IDEA
Diagram for DOCUMENTATION DEBT PAYMENT
Enabling practices
- Changing project scope (13%)
- Investing effort on TD payment activities (13%)
- Negotiating deadline extension (13%)
- Improving the development process (7%)
Impediments
- Cost (12%)
- Lack of resources (12%)
- Lack of time (12%)
- Complexity of the TD item (8%)
- Customer decision (8%)
- Lack of technical knowledge (8%)
- Effort (4%)
- Lack of Knowledge on TD (4%)
- Lack of monitoring of TD items (4%)
- Team overload (4%)

Figure F.3 IDEA diagrams for documentation debt prevention (a), monitoring (b), and payment (c)


A
IDEA Diagram for DESIGN DEBT PREVENTION
B
IDEA Diagram for DESIGN DEBT MONITORING
C
IDEA Diagram for DESIGN DEBT PAYMENT

A

Actions

- Well-defined requirements **(12%)**
- Adoption of good prog. practices **(11%)**
- Using good design practices **(9%)**
- Better project planning **(6%)**
- Following the project planning **(6%)**
- Well planned deadlines **(6%)**
- Better project management **(6%)**
- Refactoring **(5%)**
- Following well-defined process **(4%)**
- Improve documentation **(4%)**

Decision Factors

- Lack of predictability in the software development **(21%)**
- Lack of concern about maintainability **(14%)**
- Requirements change **(14%)**
- Continuous change of coding standards **(7%)**
- Ineffective management **(7%)**

Enabling practices

- Implementation of a TD identification strategy **(6%)**
- Training **(6%)**
- Implement. of a TD mgmt. strategy **(4%)**
- Appropriate tasks allocation **(3%)**
- Prioritization of TD payment **(3%)**
- Risk and impact analysis **(3%)**
- Allocation of qualified professionals **(2%)**
- Good communication between stakeholders **(2%)**
- Using agile practice **(2%)**
- Well-defined effort estimation methods **(2%)**

Impediments

- Short deadline **(29%)**
- Lack of experience **(7%)**
- Lack of good technical solutions **(7%)**
- Lack of information **(7%)**
- Lack of qualified professionals **(7%)**
- Not sure if client will accept it **(7%)**
- Pressure for results **(7%)**

B

Actions

- TD item backlog **(13%)**
- Communicating the stakeholders of TD items **(10%)**
- TD as a task **(8%)**
- Team meetings **(7%)**
- Tracking TD items **(7%)**
- Prioritization of TD items **(5%)**
- TD estimation **(5%)**
- Dashboard **(2%)**
- Identify the worst debt areas **(2%)**
- TD status progress report **(2%)**

Decision Factors

- Lack of interest **(27%)**
- Focusing on short term goals **(15%)**
- TD Item eliminated as soon as identified **(4%)**
- React when become a problem **(2%)**
- TD item payment do not generate revenue **(2%)**
- Too many TD items **(2%)**

Enabling practices

- Use of tools **(15%)**
- TD management plan **(8%)**
- Improving software development process **(5%)**
- Adoption of agile methodology **(2%)**
- Continuous integration **(2%)**
- Knowledge sharing **(2%)**

Impediments

- Lack of knowledge on TD **(13%)**
- Lack of time **(13%)**
- Lack of understanding about the impact of the debt **(6%)**
- Lack of specific team **(4%)**
- Team overload **(4%)**
- Changes in management **(2%)**
- Changing in the requirements **(2%)**
- Company with projects beyond capacity **(2%)**
- Cost **(2%)**
- Inaccurate time estimate **(2%)**

C

Actions

- Code refactoring **(51%)**
- Design refactoring **(15%)**
- Adjusting code to follow good programming practices **(4%)**
- Bug fixing **(4%)**
- Restarting the project from scratch **(2%)**
- Update system documentation **(2%)**

Decision Factors

- Focusing on short term goals **(28%)**
- Lack of organizational interest **(15%)**
- Insufficient management view about TD payment **(3%)**
- Lack of adoption of lessons learned **(3%)**
- Lack of testing **(2%)**
- Number of TD items **(2%)**
- TD Items do not have "interest" **(2%)**

Enabling practices

- Investing effort on TD payment activities **(11%)**
- Hiring specialized professionals **(9%)**
- Improving the development process **(6%)**
- Improving the team collaboration **(6%)**
- Negotiating deadline extension **(6%)**
- Increasing the project budget **(2%)**
- Using continuous integrations and deliveries **(2%)**

Impediments

- Cost **(18%)**
- Lack of time **(13%)**
- Complexity of the project **(8%)**
- Customer decision **(7%)**
- Complexity of the TD item **(5%)**
- Effort **(5%)**
- Lack of resources **(5%)**
- High team turnover **(2%)**
- Lack of access on component code **(2%)**
- Lack of knowledge on TD **(2%)**

Figure F.4 IDEA diagrams for design debt prevention (a), monitoring (b), and payment (c)

Appendix G

EVALUATION FORM

Formulário de Avaliação

* Required

1. 1. Nome *

   ____

2. 2. Endereço de e-mail *

   ____

Utilidade dos diagramas

3. 3. Em relação à utilidade do uso dos diagramas para apoiar a identificação de práticas e razões associadas às atividades de PREVENÇÃO da dívida técnica, marque a opção que melhor representa seu ponto de vista: *

Mark only one oval per row.

	Concordo totalmente	Concordo parcialmente	Neutro	Discordo parcialmente	Discordo totalmente
a. Usando os diagramas propostos, eu seria capaz de identificar práticas de prevenção da dívida técnica mais rapidamente.	◯	◯	◯	◯	◯
b. Usando os diagramas propostos, eu iria melhorar o meu desempenho na identificação de práticas de prevenção da dívida técnica.	◯	◯	◯	◯	◯
c. Usando os diagramas propostos, eu iria melhorar a minha eficácia em identificar práticas de prevenção da dívida técnica.	◯	◯	◯	◯	◯
d. Usando os diagramas propostos, eu tornaria mais fácil a	◯	◯	◯	◯	◯

identificação de práticas de prevenção da dívida técnica.					
e. Usando os diagramas propostos, eu seria capaz de identificar razões para a não-prevenção da dívida técnica mais rapidamente.	○	○	○	○	○
f. Usando os diagramas propostos, eu iria melhorar a minha eficácia em identificar razões para a não-prevenção da dívida técnica.	○	○	○	○	○
g. Usando os diagramas propostos, eu iria melhorar o meu desempenho na identificação de razões para a não-prevenção da dívida técnica.	○	○	○	○	○
h. Usando os diagramas propostos, eu tornaria mais fácil a	○	○	○	○	○

identificação de razões para a não-prevenção da dívida técnica.

4. 4. Em relação à utilidade do uso dos diagramas para apoiar a identificação de práticas e razões associadas às atividades de MONITORAMENTO da dívida técnica, marque a opção que melhor representa seu ponto de vista: *

Mark only one oval per row.

	Concordo totalmente	Concordo parcialmente	Neutro	Discordo parcialmente	Discordo totalmente
a. Usando os diagramas propostos, eu seria capaz de identificar práticas de monitoramento da dívida técnica mais rapidamente.	⬭	⬭	⬭	⬭	⬭
b. Usando os diagramas propostos, eu iria melhorar o meu desempenho na identificação de práticas de monitoramento da dívida técnica.	⬭	⬭	⬭	⬭	⬭
c. Usando os diagramas propostos, eu iria melhorar a minha eficácia em identificar práticas de monitoramento da dívida técnica.	⬭	⬭	⬭	⬭	⬭
d. Usando os diagramas propostos, eu tornaria mais fácil a identificação de práticas de monitoramento da dívida técnica.	⬭	⬭	⬭	⬭	⬭

e. Usando os diagramas propostos, eu seria capaz de identificar razões para o não-monitoramento da dívida técnica mais rapidamente.	◯	◯	◯	◯	◯
f. Usando os diagramas propostos, eu iria melhorar a minha eficácia em identificar razões para o não-monitoramento da dívida técnica.	◯	◯	◯	◯	◯
g. Usando os diagramas propostos, eu iria melhorar o meu desempenho na identificação de razões para o não-monitoramento da dívida técnica.	◯	◯	◯	◯	◯
h. Usando os diagramas propostos, eu tornaria mais fácil a identificação de razões para o não-monitoramento da dívida técnica.	◯	◯	◯	◯	◯

5. 5. Em relação à utilidade do uso dos diagramas para apoiar a identificação de práticas e razões associadas às atividades de PAGAMENTO da dívida técnica, marque a opção que melhor representa seu ponto de vista: *

Mark only one oval per row.

	Concordo totalmente	Concordo parcialmente	Neutro	Discordo parcialmente	Discordo totalmente
a. Usando os diagramas propostos, eu seria capaz de identificar práticas de pagamento da dívida técnica mais rapidamente.	⬭	⬭	⬭	⬭	⬭
b. Usando os diagramas propostos, eu iria melhorar o meu desempenho na identificação de práticas de pagamento da dívida técnica.	⬭	⬭	⬭	⬭	⬭
c. Usando os diagramas propostos, eu iria melhorar a minha eficácia em identificar práticas de pagamento da dívida técnica.	⬭	⬭	⬭	⬭	⬭
d. Usando os diagramas propostos, eu tornaria mais fácil a	⬭	⬭	⬭	⬭	⬭

identificação de práticas de pagamento da dívida técnica.					
e. Usando os diagramas propostos, eu seria capaz de identificar razões para o não-pagamento da dívida técnica mais rapidamente.	◯	◯	◯	◯	◯
f. Usando os diagramas propostos, eu iria melhorar a minha eficácia em identificar razões para o não-pagamento da dívida técnica.	◯	◯	◯	◯	◯
g. Usando os diagramas propostos, eu iria melhorar o meu desempenho na identificação de razões para o não-pagamento da dívida técnica.	◯	◯	◯	◯	◯
h. Usando os diagramas propostos, eu tornaria mais fácil a	◯	◯	◯	◯	◯

identificação de razões para o não-pagamento da dívida técnica.					

6. 6. Em relação à utilidade do uso dos diagramas para apoiar a identificação de práticas e razões associadas às atividades de PREVENÇÃO, MONITORAMENTO e PAGAMENTO da dívida técnica, marque a opção que melhor representa seu ponto de vista: *

Mark only one oval per row.

	Concordo totalmente	Concordo parcialmente	Neutro	Discordo parcialmente	Discordo totalmente
a. Usando os diagramas propostos, eu iria aumentar a minha produtividade.	◯	◯	◯	◯	◯
b. Eu acredito que os diagramas propostos seriam úteis para apoiar a gestão da dívida técnica.	◯	◯	◯	◯	◯

Facilidade de uso e uso futuro

7. 7. Em relação à facilidade de uso dos diagramas para apoiar a identificação de práticas e razões associadas às atividades de PREVENÇÃO, MONITORAMENTO e PAGAMENTO da dívida técnica marque a opção que melhor representa seu ponto de vista: *

Mark only one oval per row.

	Concordo totalmente	Concordo parcialmente	Neutro	Discordo parcialmente	Discordo totalmente
a. Aprender a utilizar os diagramas propostos seria fácil para mim.	⬭	⬭	⬭	⬭	⬭
b. Minha interação com os diagramas propostos seria clara e compreensível.	⬭	⬭	⬭	⬭	⬭
c. Eu acharia fácil utilizar os diagramas para identificar práticas associadas à PREVENÇÃO da dívida técnica.	⬭	⬭	⬭	⬭	⬭
d. Eu acharia fácil utilizar os diagramas para identificar razões para a não-PREVENÇÃO da dívida técnica.	⬭	⬭	⬭	⬭	⬭
e. Eu acharia fácil utilizar os diagramas para identificar práticas associadas ao MONITORAMENTO da dívida técnica.	⬭	⬭	⬭	⬭	⬭
f. Eu acharia fácil utilizar os diagramas para identificar razões para o não-MONITORAMENTO da dívida técnica.	⬭	⬭	⬭	⬭	⬭

g. Eu acharia fácil utilizar os diagramas para identificar práticas associadas ao PAGAMENTO da dívida técnica.	◯	◯	◯	◯	◯
h. Eu acharia fácil utilizar os diagramas para identificar razões para o não-PAGAMENTO da dívida técnica.	◯	◯	◯	◯	◯
Seria fácil tornar-se hábil no uso dos diagramas propostos.	◯	◯	◯	◯	◯
Seria fácil se lembrar de como identificar práticas e razões associadas à PREVENÇÃO, ao MONITORAMENTO e ao PAGAMENTO usas e efeitos da DT utilizando os diagramas propostos.	◯	◯	◯	◯	◯
Eu acredito que os diagramas propostos são fáceis de utilizar.	◯	◯	◯	◯	◯

8. 8. Em relação a um possível uso futuro dos diagramas para apoiar a identificação de práticas e razões associadas às atividades de PREVENÇÃO, MONITORAMENTO e PAGAMENTO da dívida técnica, marque a opção que melhor representa seu ponto de vista: *

Mark only one oval per row.

	Concordo totalmente	Concordo parcialmente	Neutro	Discordo parcialmente	Discordo totalmente
a. Assumindo que os diagramas propostos estariam disponíveis para gerenciar a dívida técnica, eu os utilizaria no futuro.	◯	◯	◯	◯	◯
b. Eu preferiria usar os diagramas propostos a identificar práticas e razões associadas às atividades de PREVENÇÃO, MONITORAMENTO e PAGAMENTO da dívida técnica da maneira usual (sem o apoio dos diagramas).	◯	◯	◯	◯	◯

Sugestões e melhorias

9. 9. De acordo com sua opinião, liste os aspectos positivos da utilização dos diagramas: *

10. 10. De acordo com sua opinião, liste os aspectos negativos da utilização dos diagrama: *

11. 11. Caso tenha, por favor, indique alguma sugestão para melhoria dos diagramas utilizados. *

12. 12. O uso dos diagramas ajudou a identificar práticas de prevenção, monitoramento ou pagamento da dívida técnica que você não teria identificado sem o apoio deles? *

Mark only one oval.

- ◯ Sim
- ◯ Não

13. 13. O uso dos diagramas ajudou a identificar razões para a não-prevenção, não-monitoramento ou não-pagamento da dívida técnica que você não teria identificado sem o apoio deles? *

Mark only one oval.

- ◯ Sim
- ◯ Não