UNIVERSITY OF TURKU

Evaluating a Refined Technical Debt Management Framework in Long-Lived Agile Projects: A Mixed-Methods Case Study

Software Engineering
Department of Computing
Master of Science in Technology Thesis

Author:
Mikael Järvinen

18.5.2026
Turku

The originality of this thesis has been checked in accordance with the University of Turku quality assurance system using the Turnitin Originality Check service.

Master's thesis

Subject: Software Engineering
Author(s): Mikael Järvinen
Title: Evaluating a Refined Technical Debt Management Framework in Long-Lived Agile Projects: A Mixed-Methods Case Study
Supervisor(s): Ville Leppänen
Number of pages: 102 + 7
Date: 4.5.2026

Long-lived Agile software projects accumulate technical debt (TD) under feature pressure. This thesis evaluates the practical applicability of a refined Technical Debt Management Framework (TDMF), developed from the author's bachelor's-thesis precursor, in an industrial Agile software development team. The research questions concern how the framework's proposed application works in practice, how team members perceive it, what benefits and challenges arise, and how it should be adapted. The study is a single-team, four-week mixed-methods case study at a Finnish software organisation maintaining a two-decade-old SaaS platform. Data were collected through a focus group, ten observed team meetings, five post-implementation interviews, and pre- and post-implementation Likert questionnaires. Qualitative data were analysed through reflexive thematic analysis; quantitative data were reported as group-level descriptive statistics. The framework was applied selectively. The team adopted its structural elements — a recurring TD discussion forum, allocated time, and consistent recording of TD items — at kick-off, but sustained enactment was partial under capacity pressure. The framework's calculative prioritisation step (ROI based on principal-and-interest estimates) was declined at the first refinement and not reinstated. The team identified the framework's most valuable potential as upward communication of TD to management — a register the framework did not address. Four adaptation directions are derived: upward-facing artefacts for management communication, presenting the calculative step as conditional rather than default, substrate-aware framing, and definitional conventions for TD measurement. The case study contributes evidence that calculative TDM steps require enabling conditions often absent in long-lived Agile projects, and identifies upward-facing communication as an underdeveloped leverage point for framework design.

Key words: technical debt management, TDMF, Agile software development, Scrum, case study, reflexive thematic analysis, framework evaluation.

Table of contents

1 Introduction

1.1 Background and Motivation

Technical Debt is a well-established metaphor that refers to sub-optimal technical implementations or design decisions [1], [2], [3]. Incurring technical debt may speed up the development and shorten time to market, but in the long run it can slow down development, decrease maintainability, cause bugs etc. [1]. Especially in long-lived software projects technical debt can accumulate to a point where software organizations might need to rewrite the whole software project due to significant hindrance to productivity [1].

To address this issue, several approaches to technical debt management (TDM) have been proposed in both research and practice [2], [4], [5]. One of the most influential is the Technical Debt Management Framework (TDMF) [6], which provides a universal structured method for identifying, monitoring, and prioritizing technical debt [4], [6]. However, most applications of TDMF in literature have focused on the use and practicality of TDMF [4]. This raises the question of how to use the TDMF in Agile projects, and how it fits into the Agile process.

The author's bachelor's thesis proposed an extension of TDMF tailored for Scrum rituals [7] which can be used in any Agile project not just in Scrum projects. Its applicability in real-world projects has not been evaluated. The organization of this thesis maintains and develops a two-decade-old SaaS platform using development practices close to Scrum. This context presents a practical challenge for the organization and a research opportunity to test the extension of TDMF in an Agile setting.

From a practical perspective, the case organization currently lacks a structured approach for handling technical debt in its day-to-day work. The development team is aware of technical debt and have documented some in the organization's knowledge base. Technical debt is evaluated and prioritized accordingly in yearly intervals. However, the instances of technical debt lack documentation on how large they are in terms of their influence on development and maintenance and how much it would cost to fix them, and the evaluation of technical debt is a vague process outside of the Agile process of the team. This shows up as technical debt is rarely being repaid, and developers fixing them on their own outside of the team's Agile process. Introducing a systematic approach could provide concrete benefits for decision-making, prioritization and long-term maintainability.

From an academic perspective, studying the applicability of the extended TDMF in this environment will contribute to the broader understanding of how technical debt management frameworks can be applied and how they perform in different types of Agile projects, especially those with possible legacy constraints.

1.2 Problem Statement and Goals

Although TDMF has been widely discussed in the literature [2], [4], [5], [6], its applicability in different types of long-lived projects is still underexplored. Legacy projects face unique challenges, such as accumulated large instances of technical debt, outdated technologies, and heterogeneous development practices. The problem addressed in this thesis is the lack of knowledge about how TDMF can be applied and adapted to Agile projects.

The goal of the thesis:

- **Academic goal**: To study and evaluate the applicability of the refined TDMF, which builds upon the previously proposed extended TDMF, in an Agile context and to provide new insights into the use of technical debt management frameworks, especially in this context.
- **Practical goal**: To provide the case organization with a concrete approach for identifying, managing, and prioritizing technical debt in their day-to-day work.

1.3 Research Questions and Methodology

To achieve the academic and practical goals of this thesis the following research questions were chosen:

- **RQ1**: *How does the proposed application of the refined Technical Debt Management Framework (TDMF) work in practice in an Agile software project, and what differences emerge between the intended theoretical application and its actual implementation?*
    - **Justification**: This research question focuses not on designing a new way to apply the TDMF, but on examining how the theoretically proposed application (presented in Chapter 3) works in practice in a real Agile software team. The goal is to compare the originally proposed application model with the team's actual use of the framework and to identify necessary context-specific modifications, constraints, and deviations from the theoretical model.

    - **Method**: Focus groups were organized to introduce the refined TDMF to the development team, support its adoption, and to collaboratively produce a plan on how to apply the refined TDMF into the development team's practices.
    - **Data**: Focus group transcriptions, application plan on how to apply the refined TDMF, observation notes on the development team's early usage of the framework.
    - **Analysis**: Thematic analysis.
- **RQ2**: *How do development team members perceive and experience the framework as part of their Agile processes?*
    - **Justification**: RQ1 focuses on application, RQ2 addresses perception. Adoption of a framework doesn't depend only on its theoretical fit but also on whether practitioners find it usable and valuable. This question ensures that the voices of the development team members are considered in the evaluation.
    - **Method**: Structured interviews with developers and team leaders.
    - **Data**: Interview transcripts containing perceptions, ease of use, usefulness, and challenges.
    - **Analysis**: Thematic analysis.
- **RQ3**: *What benefits and challenges arise from applying the framework in an Agile project?*
    - **Justification**: This question was chosen to capture the practical outcomes, both positive and negative, of using the refined TDMF in an Agile environment. It connects directly to the practical goal of helping the case organization understand the consequences of framework adoption.
    - **Method**: Questionnaire at the start and end of the research. Direct observation of the team's daily practices over the course of the framework's application.
    - **Data**: Questionnaire answers containing how well technical debt is managed. Field notes on how technical debt was identified, prioritized, and managed, combined with interview insights.
    - **Analysis**: Comparison of questionnaire answers from the start and end of the research combined with insights from the interviews.
- **RQ4**: *How should the framework be adapted to better support Agile projects?*
    - **Justification**: Finally, RQ4 is forward-looking. It synthesizes the findings from RQ1-RQ3 to propose adjustments that could make the framework more effective in contexts similar to the case project. This question addresses the academic goal by contributing new insights to the literature on adapting TDMF beyond Scrum. This also addresses the practical goal by providing recommendations for the case organization and similar teams.
    - **Method**: Integration of findings from focus groups, questionnaires, observations, and interviews.
    - **Data**: Suggestions for improvements made by team members, plus researcher reflections on misalignments between the refined TDMF and Scrum-like practices.

- **Analysis:** Synthesis of lessons learned into recommended adjustments to the framework for Agile projects.

To address these questions, this thesis employs a case study approach in the case organization maintaining a two-decade-old SaaS platform. A mixed-methods approach combining qualitative methods (focus groups, observation, interviews) with limited quantitative pre- and post-implementation questionnaires was selected, prioritising in-depth understanding of applicability, perceptions, and contextual challenges over quantitative measurement of outcomes.

The chosen methodology combines elements from case study research and participatory inquiry. A case study design was chosen because the research investigates a phenomenon [8], [9], the applicability of refined TDMF, in its real-life context. At the same time, focus groups and interactions with the subject team members provide an opportunity for collaborative exploration similar to action research [8], [9], though the primary focus is on evaluating applicability rather than iterative improvement cycles.

Focus groups, questionnaires, observation, and interviews provide complementary perspectives [8], [9]: Focus groups capture the participants' intended application of the refined TDMF, observation captures adoption and real-world practice, questionnaires and interviews capture subjective experiences. These methods together enable a holistic understanding of how the refined TDMF functions in the chosen context.

1.4 Thesis Structure

This thesis is structured as follows:

- **Chapter 2** presents background literature on Agile methodologies, technical debt definition and its impact, existing frameworks for managing technical debt and technical debt prioritization methods.
- **Chapter 3** introduces the extended TDMF proposed in prior work, refined TDMF which is built upon the prior work, and discusses the reasoning behind it and its potential applicability when the case organization does not strictly follow Scrum.
- **Chapter 4** describes the research methodology: the case study design, data collection, and analysis.

- **Chapter 5** reports how the refined framework was applied in the case project through focus groups, questionnaires, observation and interviews.
- **Chapter 6** presents the results, including team members' experiences, observed benefits, and challenges.
- **Chapter** 7, the discussion chapter answers the research questions and reflects on the implications for long-lived Agile projects.
- **Chapter 8** concludes the thesis. This chapter summarizes key findings, contributions, limitations, and directions for future research.

1.5 Use of AI Assistance

An AI assistant — Claude Code by Anthropic, powered by the Claude Opus 4.7 large language model with 1M context — was used during the research and writing of this thesis as a research assistant, not as a co-author. The boundary between assistance and authorship is intentionally explicit and is described here so the reader can interpret the work that follows accordingly.

The thesis's intellectual contributions are the authors. All research questions, theoretical positioning, framing of the case study, judgements about what counts as evidence, interpretations of qualitative and quantitative data, themes constructed from the coded material, claims about what the findings mean, conclusions drawn in answer to the research questions, recommendations for framework adaptation, the structure and flow of the written argument, and the analytical narrative throughout — these are the author's own work. The thesis was written by the author; AI assistance worked on the author's drafts, not in place of them. Where the author drew on AI-suggested edits or rewordings, the suggestions were reviewed, judged for fit and accuracy, and either accepted, modified, or rejected before any text was retained in the manuscript.

AI assistance was bounded to two categories of work, both performed under the author's review and revision authority. First, in data analysis: AI proposed initial codes from interview, focus-group, and observation transcripts, computed descriptive statistics from the questionnaire data, generated cross-source summaries, and produced structured outputs such as code distributions and triangulation tables; the author validated, modified, or rejected every AI-proposed code and made all interpretive decisions, in accordance with the workflow

described in Section 4.4 and detailed in Appendix 4. Second, in writing and editing: the author drafted the prose of each chapter, and AI then assisted with mechanical work on the author's existing drafts — cross-checking content that appears in multiple chapters for consistency, reviewing and fixing references and citations, suggesting language adjustments to tighten the academic register, reviewing heading structure and section ordering, and generating tables and figures from the analysis data. Every AI-suggested edit to the author's prose was reviewed by the author and accepted, modified, or rejected based on the author's own judgement.

In addition to the assistance described above, the audio recordings of the focus group and interviews were transcribed using OpenAI's Whisper, an open-source automatic speech-recognition model that was run locally on the researcher's hardware so that no audio left the researcher's machine. The researcher verified each AI-produced transcript against the original recording before any analysis was undertaken. This use of AI for transcription was covered by the original participant consent form and is further described in Section 4.3 (Data Collection Methods).

Where AI assistance touched a methodological commitment — for example, in qualitative coding, where coding decisions ultimately shape the findings — the assistance is described in the relevant methodological section so the reader can assess whether the boundaries described here have been respected in practice. The author bears full responsibility for the substantive content, the analytical claims, and any errors that remain.

2 Background

In this chapter we review the related literature that was used as background for the extended TDMF and refined TDMF presented in Chapter 3. We start with a brief description of **Agile Methods** (Section 2.1) after which we introduce **Scrum** (Section 2.2) which was used in defining the extended TDMF in prior work. From **Agile Methods** we move on to **Technical Debt** (Section 2.3) defining **Technical Debt** (Section 2.4) and Technical Debt Management (Section 2.5).

2.1 Agile Methods

Agile Methods are a group of project management methods that originally emerged from software development, but they are not restricted to software development. The methods emphasize flexibility, open communication and customer collaboration [10]. The methods are based on the Agile manifesto [10] that presents the Agile values and principles. These values and principles focus on collaboration, continuous delivery, adaptability to changes, and on self-management and continuous improvement of a motivated team [10]. The methods were developed as an alternative to the traditional **Waterfall method**, where the requirements are specified well in advance and the development work is done in planned phases. The **Waterfall method,** due to its rigidity, does not work well with changing and unclear requirements, which are typical in software development. The most popular Agile frameworks are **Scrum** and **Kanban**.

2.2 Scrum

The Scrum guide is an official guidebook for the Scrum framework [11]. The guide is a detailed document that describes the Scrum framework in project management. The guide clearly defines the different roles and responsibilities, processes (or "rituals") and the rules for utilizing Scrum.

Scrum is the most popular framework of all Agile frameworks, and it stands out with its detailed framework, specific roles and rituals. Scrum is based on empiricism and Lean thinking. "*Empiricism asserts that knowledge comes from experience and making decisions based on what is observed. Lean thinking reduces waste and focuses on the essentials.*" [11] Scrum is partially incomplete in a sense that it offers a degree of freedom for implementing it, it only specifies those parts that are necessary for implementing Scrum [11]. Scrum's partial

incompleteness makes it possible to utilize it in other industries and projects outside of software development.

In Scrum, as in other Agile frameworks, backlogs are utilized:

Table 1. Description of and comparison of Scrum's backlogs [11]

Product backlog	Sprint backlog
A dynamic document or a list that contains all the desired features, requirements, fixes and improvements, that can be a part of the project. It exists throughout the project tracking what the team should do next. The purpose of this backlog is to contain all the possible requirements for the project in one place.	Contains all the tasks and requirements which the team has committed to complete during the current Sprint. It is the result of a self-managing team, when they select tasks and requirements from the Product Backlog and commit to completing them during the Sprint. This backlog is a tool for the team to track and keep the focus on the tasks to be completed in the current Sprint.

One of the distinguishing features of Scrum is roles:

- **Product Owner** is responsible for the value created by the Scrum team and the Product Backlog. Product Owner develops and presents the Product Goal, creates, communicates and prioritizes the Product Backlog items. [11]
- **Scrum Master** has a coaching role in the Scrum team. Their responsibility is to uphold the team's and the organization's understanding of Scrum and make sure the team follows the Scrum theory [11]. The Scrum Master also helps the team in different Scrum events, strives to identify and remove blockers and develop the team's working practices [11]. Combining all these responsibilities the Scrum Master's responsibility is the team's effectiveness.
- **Developers** are the people in the team who work to accomplish the items on the Sprint Backlog chosen from the Product Backlog [11]. These developers could be programmers, designers, quality assurance engineers, or experts from other disciplines since Scrum is not restricted only to software development, but in Scrum all of these are called developers.

In the Scrum guide the different stakeholders have not been defined as Scrum roles, but they still have an important role to play when it comes to Scrum and the development work – customers, users, sponsors, etc. These stakeholders are usually present in the Sprint Review, where they give feedback on the outcome of the Sprint, and where they can impact the future development work. These stakeholders have a more indirect role when compared to Scrum roles. The Product Owner, unlike the other stakeholders, is part of the Scrum team, and they strive to bring the thoughts of the other stakeholders into the team. The different stakeholders have a central impact on what the team is working on, when it will be worked on and in which order.

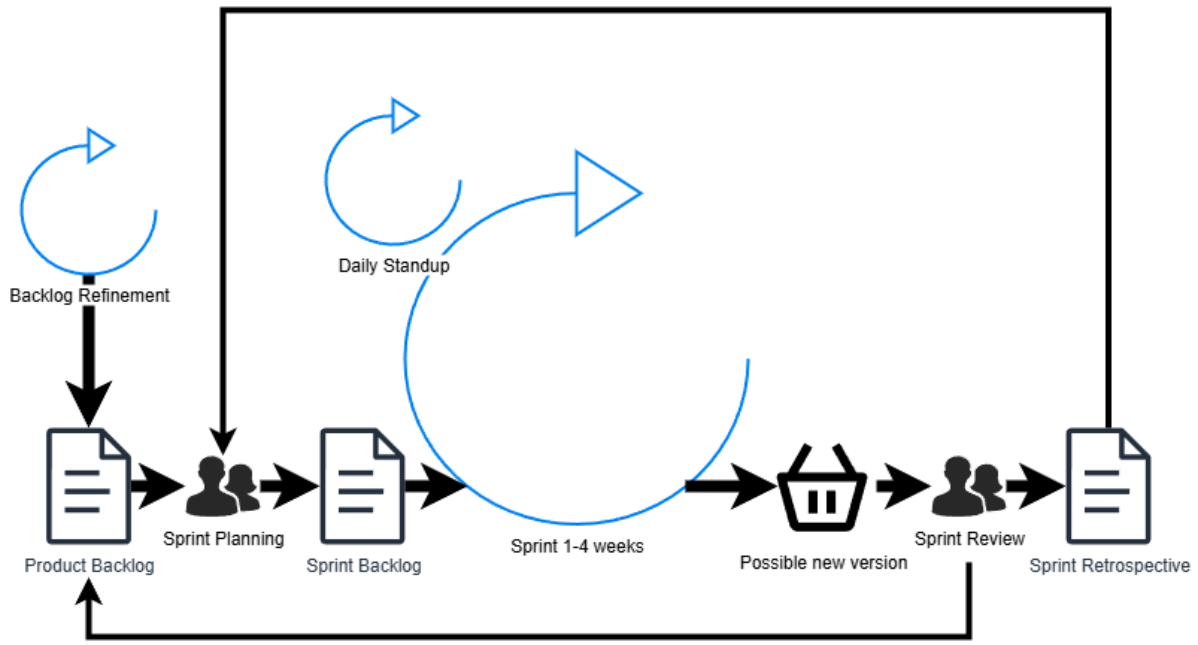


Figure 1. Scrum rituals and their relationships [11]

Figure 1 explains the different relationships between the different Scrum events. Sprint planning is the starting point of the Sprint, Product Owner proposes how this Sprint could provide value based on which the team forms a Sprint Goal that directs the team's work and decision making, and it communicates to different stakeholders what kind of value the team is trying to produce. The team collaborates on choosing the relevant items from the Product Backlog to fulfil the Sprint Goal. After the items have been chosen the team plans how to accomplish these items and moves these to the Sprint Backlog. The Sprint Backlog consists of the Sprint Goal, items chosen from the Product Backlog and the plan on accomplishing them. [11]

Once the Sprint Backlog is ready the team starts the Sprint. Sprint is a fixed time-interval 1-4 weeks –most often 2 weeks– during which the team strives to create a functional, testable, and a potentially shippable version [11]. The purpose of the Sprint is to create a clear time-interval during which the team strictly focuses on the committed tasks and goals [11]. The next Sprint starts right after the previous one [11]. The Daily Standup is an important event in Scrum where the team gathers daily to discuss progress and possible blockers [11]. The Daily Standup enables quick adaptation to challenges and changing plans [11].

At the end of the Sprint the team gathers with different stakeholders to inspect the outcome of the Sprint, this event can include a demo of the potentially shippable version. This event is

called the Sprint Review. The participants discuss potential next steps and work items and possibly adjust the Product Backlog if needed [11].

The Sprint Retrospective is the last event before the next Sprint; it follows the Sprint review and precedes the next Sprint Planning. In the retrospective the team inspects accomplishments, challenges and possible work improvements [11]. The inspection is not limited to work, but it can also include other things depending on the context of the work. The team decides on possible changes which can also include changes to the Product Backlog [11].

2.3 Technical Debt

Technical Debt (TD) is an established metaphor used to describe quality-wise bad but functional code, that can cause hindrances in software development. Ward Cunningham's description of immature but functional code [1] is the original description of TD. Cunningham describes a software practice where it is normal for a new feature to deviate from existing software architecture which is later on fixed to fit in to the existing architecture [1]. This software practice allows features to be released earlier, this type of practice led to the original metaphor [1]: "*Shipping first time code is like going into debt. A little debt speeds development so long as it is paid back promptly with a rewrite*." The risk is not paying this debt, just like financial debt TD incurs interest, TD can possibly slow down development time and accumulated TD can completely halt the development of a software system [1]. In other words, Cunningham's description is a balancing act between software quality and speed of development.

Since Cunningham's original description of TD, the metaphor has been used to also describe other issues, not just program code [4], [12], [13]. The concept of TD has extended outside of program code to describe different artefacts in a software project and in the software development lifecycle [3].

2.4 Defining Technical Debt

From Cunningham's metaphor it can be concluded that a human is a big factor in the emergence of TD, as a human's understanding of the software, knowledge and skill determines what kind of software is developed, as does non-technical factors like the business

and organization [13]. Martin Fowler made the Technical Debt Quadrant to visualize different causes for TD [14] as seen in Figure 2.

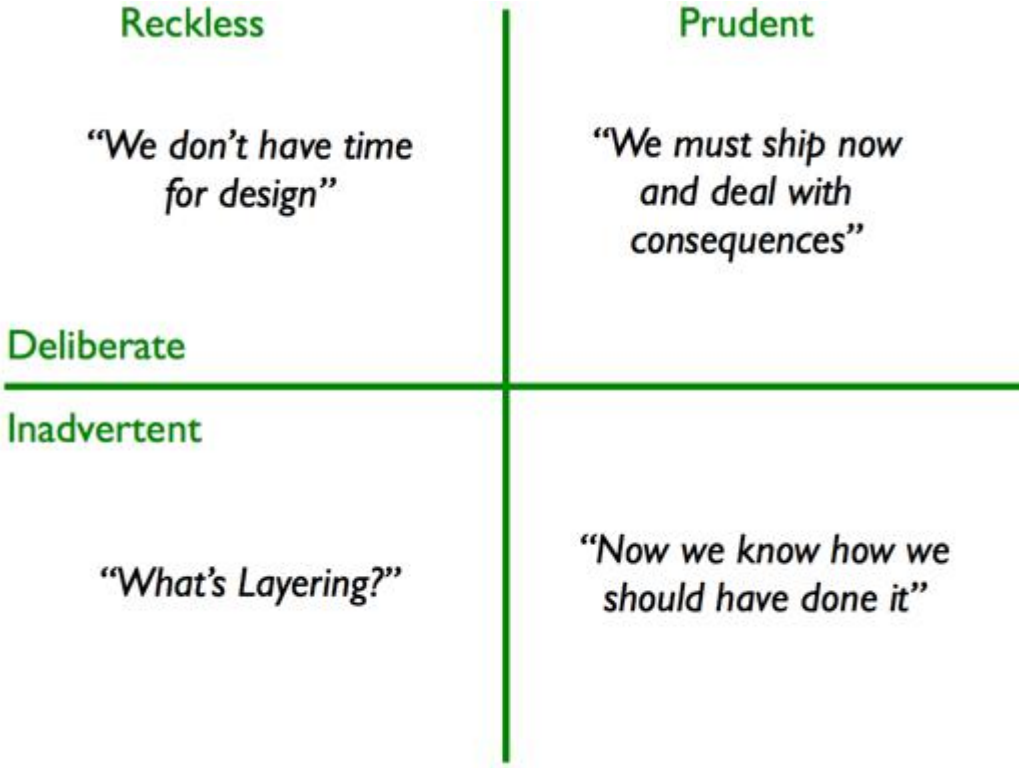


Figure 2. Martin Fowler's Technical Debt Quadrant [14]

It is common to think that time pressure is a big cause for TD, but in reality recklessness, lack of education, inadequate processes, incomprehensive quality assurance or just plain ineptitude can be as big or bigger causes of TD than time pressure [12]. In Figure 2. the Technical Debt Quadrant [14] divides TD to 4 categories based on intent and context of the emergence of TD: reckless and prudent, which are further divided into deliberate and inadvertent.

Reckless-Deliberate TD refers to the TD that is knowingly taken due to reasons such as strict deadlines or unwillingness to refactor code to not introduce TD. **Prudent-Deliberate** is also knowingly taken much like **Reckless-Deliberate**, but the difference lies in that the consequences are acknowledged, and further care is taken to "deal with the consequences" meaning it is acknowledged that it must be dealt with later. [14]

Reckless-Inadvertent debt refers to debt that occurs due to negligence, ignorance or just plain lack of knowledge without being aware of it and its consequences. **Prudent-Inadvertent** debt occurs when the team produces a clean design using the best practices known at the time, but later — through experience or new knowledge — realises that a better

approach existed; the debt is recognised retrospectively rather than at the moment of incurring it. [14]

To further illustrate the different types of technical debt, Li et al. identified several categories of technical debt in their systematic mapping study, as summarized in Table 2 [4].

Table 2. Technical Debt types [4]

TD type	Definition
Requirement debt	This refers to the difference between the requirements and the implemented system.
Architectural Debt	This is caused by architectural decisions that make compromises on internal quality matters such as maintainability.
Design debt	Shortcuts taken in software design.
Code debt	This is badly written code that violates best practices or standards. For example code duplication or overly complex code.
Testing debt	Shortcuts taken in testing, for example lack of manual or automated tests.
Build debt	This refers to shortcuts taken in the software's build system that can make it overly complex or slow.
Documentation debt	This can mean missing, inadequate or outdated documentation. This can be shown by developers being unable to understand parts of the system.
Infrastructural debt	This refers to inadequacies in the processes, technologies, supporting tools related to the software development process.
Versioning debt	This refers to the problems in source code versioning, for example unnecessary branch outs.
Defect debt	This means mistakes, bugs and defects found in the software system.

TD can exist in many different areas of the software development lifecycle; it does not always refer to debt in the source code itself.

2.5 Technical Debt Management Framework

The basis for technical debt management (TDM) in practice and research is the framework proposed by Guo and Seaman (2011) (Technical Debt Management Framework, TDMF) [6]. This framework has been used in numerous studies [4] and is the starting point for the extended TDMF proposed in earlier work, it serves as the basis for TDM [4]. In this Chapter we go through the framework in detail (Figure 3).

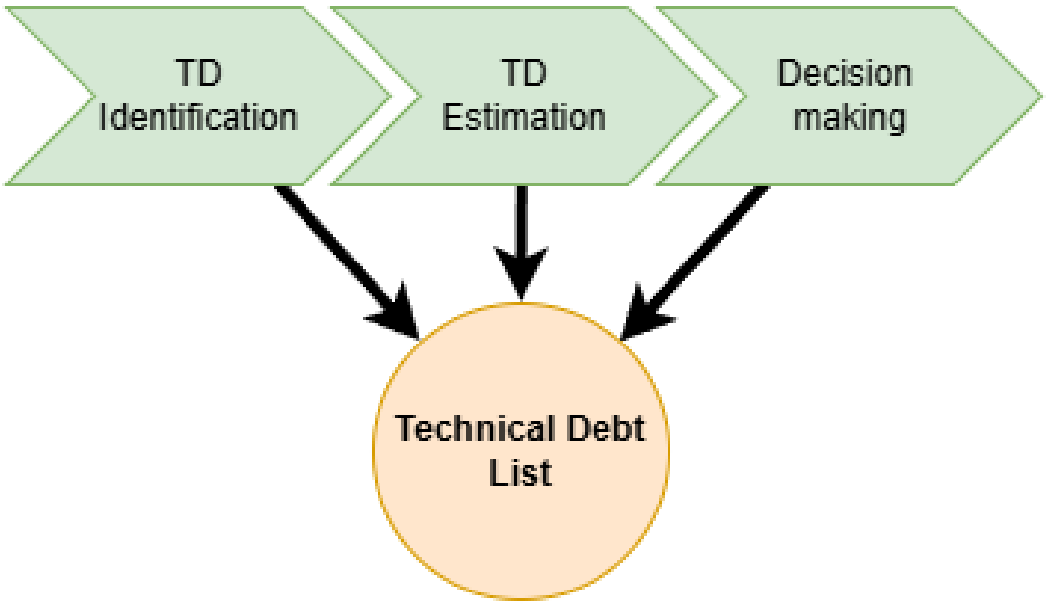


Figure 3. Technical Debt Management Framework [6]

The TDMF is very generic and flexible enough to be applied in many different contexts. The central part of the TDMF is the Technical Debt List (TDL) that all the activities revolve around as illustrated in Figure 3 the TDL is a dynamic list of technical debt items (TDI). TDI represents an instance of TD found in the software project. TDI contains a description of why it exists and a description of its location and most importantly related to TDM it contains estimations of the principal, interest and interest probability. [2], [6]

2.5.1 TD Identification

First part of the TDMF is *identification*. In addition to identifying existing TD, future TD that is to be incurred can also be identified [2], [6], since teams sometimes plan to incur TD to speed up development. The TDMF does not go into detail about how the identification happens, only that it is the first part of managing TD. Identifying existing TD can be done in many ways, most of which include human evaluation, but in addition to a human identifying the TD, automatic tools exist to enforce style rules and automatically fix them [4]. In addition, static code analysis has evolved from syntax validation to code analysis tools that can detect multiple issues related to TD [4], [15], overly complex code and poor architectural patterns as few examples. Trivial issues can be detected automatically, but a lot of TD instances are context dependent, and their scope can be very large, therefore most TD instances still need to be manually identified and evaluated [3].

TDMF does not include TD prevention, although software developers still tend to avoid incurring TD. Scrum's Definition of Done (DoD) [11] can be used to formalize TD prevention [16]. For example, we can add rules to DoD to prevent shipping code that is not reviewed by other software developers to prevent reckless TD, and we can add a minimum requirement for the test coverage to prevent test TD [16]. Also, static-analysis tools can be run on new code to detect style violations, overly complex code and poor programming patterns. All these mentioned prevention techniques are well established practices in software development and usually run automatically in a CI environment to check pull requests before they are merged.

2.5.2 TD Evaluation

Second part of the TDMF is *evaluation* where TDIs' principal, interest and interest probabilities are estimated [2] by the team. Principal is the estimated cost to repay this TDI, Interest is the estimate of how much work this TDI when left unpaid will incur extra work during a given time period for example before the next release [2]. Historical data can be used to further improve these estimates [2]. Even if this historical data is not available it is important to add these estimations, an approach called planning poker can be used by the developers in the team to achieve these estimations [17], much like estimating the work required for user stories. It is these estimations that enable the decision making and prioritization part of TDMF. Table 3 has an example of a documented TDI. Based on the TD metaphor it contains fields that describe it as debt: principal and interest, and it contains fields that are specific to TD: type, location, description and interest probability. Principal, interest and interest probability are the fields that enable decision making and prioritization.

Table 3. An example of a documented TDI

Field	Value	Description
ID	34	Serial number
Date	30.10.2025	Date or alternatively version number
Location	User authentication	Description of the location
Reporter	Mikael Järvinen	Person who reported this TDI
Type	Document	TD type
Description	Original author forgot to document how the authentication works	Description of how the TD emerged
Principal	Low	Description of the principal, an estimation of how long it takes to fix
Interest	Medium, developers unfamiliar with how this	Estimation of the amount of extra work this TD can incur if left unattended

	works can spend unnecessarily long on making changes.	
Interest probability	High	Estimation of the probability of realizing the interest.

2.5.3 Decision Making

The third part of the TDMF is decision making. Decision making includes the decision of which TDIs are chosen to be repaid, and how they are prioritized [2]. To achieve this, the team needs to have sufficient information and knowledge about TDIs which should be done in the identification part of TDMF and evaluated in the evaluation part of TDMF.

Guo and Seaman present 2 scenarios when decision making occurs: release planning and monitoring over time [2]. In the release planning scenario, a decision is made if TD is going to be repaid, how much is to be repaid, and which TDIs in the upcoming release, the assumption being that there is an up-to-date TDL. In the second scenario TDL is monitored over time to figure out when there is enough debt to justify devoting resources to repaying it. Monitoring can be done by plotting aggregated measures such as total amount of TDIs, TDIs with high principal or interest, total principal, or total interest.

The TDIs in the TDL can be prioritized in multiple different ways as there are three different fields (principal, interest and interest probability) that defines its impact. A team can prioritize based on the interest alone, but that might not be enough as a TDI also has a principal and interest probability. Next, we will review prioritization methods.

2.5.4 Prioritization

In prior work the goal was to find prioritization methods that could easily describe the benefit of repaying a TDI so that it would be easy to understand and communicate. If such a prioritization method could be found TDI could be compared with other items on the Product Backlog. Four methods were found [4], [18]:

1. High principal first [2], [15]
2. High interest first [2]
3. Cost-benefit analysis [19], [20]
4. Return of investment (ROI) [18], [21]

Prioritizing based on the amount of work required to repay a TDI or in other words the “high principal first” method does not fit our criteria. This method does not consider the interest or interest probability and the principal itself does not describe the benefit of repaying a TDI,

this also means that the TDI would not be easily comparable to other items on the Product Backlog.

In the “high interest first” method TDIs are prioritized based on a calculated interest. This method assumes that the average cost of time for changes in the location of the examined TDI is available. The interest is calculated as follows [4]:

C = Average amount of time it has taken to introduce changes to the location of the examined TDI.

W = Factor that is taken from the initial interest estimation for example 0.2 for low interest, 0.5 for medium, and 0.8 for high interest.

*Interest sum = W * C*

This method provides a numerical value based on historical data which is easy to sort. Because of the requirement of historical data, this method might be hard to use in practice, and it might be more meaningful to evaluate interest without historical data.

In the “cost-benefit analysis” method prioritization happens based on principal to interest ratio [19], [20] which is calculated as follows:

P = Principal of TDI

I = Interest sum of TDI

$$\textit{Principal to interest ratio} = \frac{P}{I}$$

This method requires a numerical value for principal and interest. Human estimations can be used to get these numerical values, and for the interest sum we can utilise the same calculation as in the “high interest first” method to utilise historical data. The same applies here that the values can be based entirely upon human estimation as historical data might not always be available. Principal to interest ratio can be used to describe the benefit of repaying a TDI, as it considers the principal and the interest. The smaller the ratio the more time you are saving compared to the time you are spending in repaying a TDI. Therefore, TDIs can be sorted based on this ratio.

The “return on investment” method acts largely the same as the “cost-benefit analysis” method. The difference is that instead of calculating the ratio of principal to interest, we calculate the difference [18], [21]:

P = *Principal of TDI*

I = *Interest sum of TDI*

$ROI = I - P$

The ROI is easy to understand as the benefit of repaying a TDI, it tells how much time is saved by repaying a TDI. The larger the ROI the more beneficial it is to repay a TDI, the smaller the less beneficial it is. It is possible for the ROI to be negative, which would in this method mean that repaying a TDI means we are wasting time, but in these cases a deeper understanding about the TDI is needed to decide if the TDI will be repaid.

Guo and Seaman also proposed a weighted approach to the ROI approach [2] where the interest is multiplied by the interest probability:

$Cost = P$ = *Principal of TDI*

I = *Interest sum of TDI*

P_I = *Interest probability of TDI*

$Benefit = I * P_I$

$ROI = Benefit - Cost = I * P_I - P$

This approach of multiplying the interest with the interest probability can be utilized with the other prioritization approaches as well. This leads to a numerical value that is based on principal, interest and interest probability providing a more meaningful value to sort the TDL based on. In Chapter 3 we evaluate these prioritization methods in the context of Scrum and discuss which one of these might be a suitable option.

2.5.5 Summary of the TDMF Approach

The central part of this approach is the TDL as seen in Figure 3. TDL is a living document containing all the TDIs that exist or may be taken in the system. Using various sources and techniques these TDIs are identified. The TDMF does not specify which sources and

techniques to use, that is the responsibility of the implementor. Using the identified TDIs the TDL can be constructed. In the next step the TDIs are evaluated by estimating the principal, interest and interest probabilities. Initial estimations can be done on a coarse scale of low, medium and high. The initial estimations can be then further defined using historical information if available. The final step is decision making by monitoring the TDL. TDL decision making can be done over time once enough TDIs have accumulated or in release planning. Several approaches can be used to quantify the benefit of repaying a TDI and to prioritize TDIs on the TDL.

3 Extended Technical Debt Management Framework

In this chapter we discuss the use of the TDMF [2] in a Scrum context and propose how to apply it in this context. We start by discussing TD related challenges in Scrum (Section 3.1), the previously proposed Scrum application of TDMF (Section 3.2), examining learnings from previous applications of TDMF in literature (Section 3.3) evaluating prioritization methods in Scrum context (Section 3.4), and our improvements to the extended TDMF (Section 3.5). In this chapter, three versions of the TDMF are discussed. The **original TDMF** refers to the framework proposed by Guo and Seaman [2]. The **extended TDMF** refers to the Scrum-specific application proposed in the author's prior work [7]. The **refined TDMF**, developed in this chapter, builds upon the extended TDMF by incorporating learnings and design principles derived from literature.

3.1 Technical Debt Related Challenges in Scrum

From Cunningham's TD metaphor and other existing literature we find time pressure and meeting deadlines as one of the most commonly mentioned reasons for the emergence of TD [17], [18], [22], [23]. And the study by Ramač et al. [24] even found deadlines to be the most dominant emergence cause for TD. Time pressure is prevalent in Scrum as well. Sprints are most commonly timeboxed to 1-4 weeks, during which the team commits to accomplishing all the tasks that were taken to the Sprint Backlog during the Sprint Planning [11]. This naturally creates pressure to complete the tasks in time. This time pressure can cause the team to take TD deliberately and inadvertently [3].

Prioritization of work happens on the Product Backlog by the Product Owner [11]. The Product Owner is not necessarily a technical person, and they need to balance the needs of all the stakeholders. TD does not have any immediate business value and explaining the benefit of repaying TD to a non-technical person can be quite hard [19], which means justifying TD repayment can be very hard [22], and instead tasks producing immediate business value are prioritized. Therefore, it is crucial to find ways to effectively communicate the benefit of repaying TD [21], and the possible impact TD can have if deferred.

To address these challenges, we can utilize the TDM approaches presented in Section 2.5.

Sprint's time pressure leads to deliberate and inadvertent TD. For the deliberate TD we identify it and create a TDI of it into the TDL, now the TD is not forgotten, and we have

enabled the monitoring and management of it. Inadvertent TD naturally needs to be identified later, but the emergence of it can be prevented by utilising Scrum's definition of done, as explained in Section 2.5.1, for example we can force peer reviews, static analysis and minimum test coverage.

Repaying TD **lacks immediate business value** which leads to the difficultness of communicating TD, and difficultness of prioritizing TD. Difficultness of communicating TD can be remediated by utilising the TDL and TDI concepts. TDL makes TD visible and transparent. And specifically, the TDI's principal and interest enable quantifying TD and communicating the impact of it if deferred. The ROI calculation mentioned in Section 2.5.4 enables quantifying the actual benefit of which then helps in communicating about TD. The prioritization methods mentioned in Section 2.5.4 will directly address the challenge of prioritizing TD.

3.2 Extended TDMF from Prior Work

The original TDMF proposed by Guo and Seaman consists of four main artefacts: TDL, identification, evaluation and decision-making. In the author's prior work [7], this framework was extended by mapping these core activities onto Scrum events and artefacts as illustrated in Figure 4. The key ideas were to embed the framework into the normal rhythm of Scrum and not treat it as a separate or ad hoc activity outside the Agile process. To avoid ambiguity, we distinguish between three versions the TDMF. The original TDMF refers to the framework proposed by Guo and Seaman. Extended TDMF refers to the version proposed in author's prior work. Refined TDMF refers to the further refined version developed in Section 3.5.

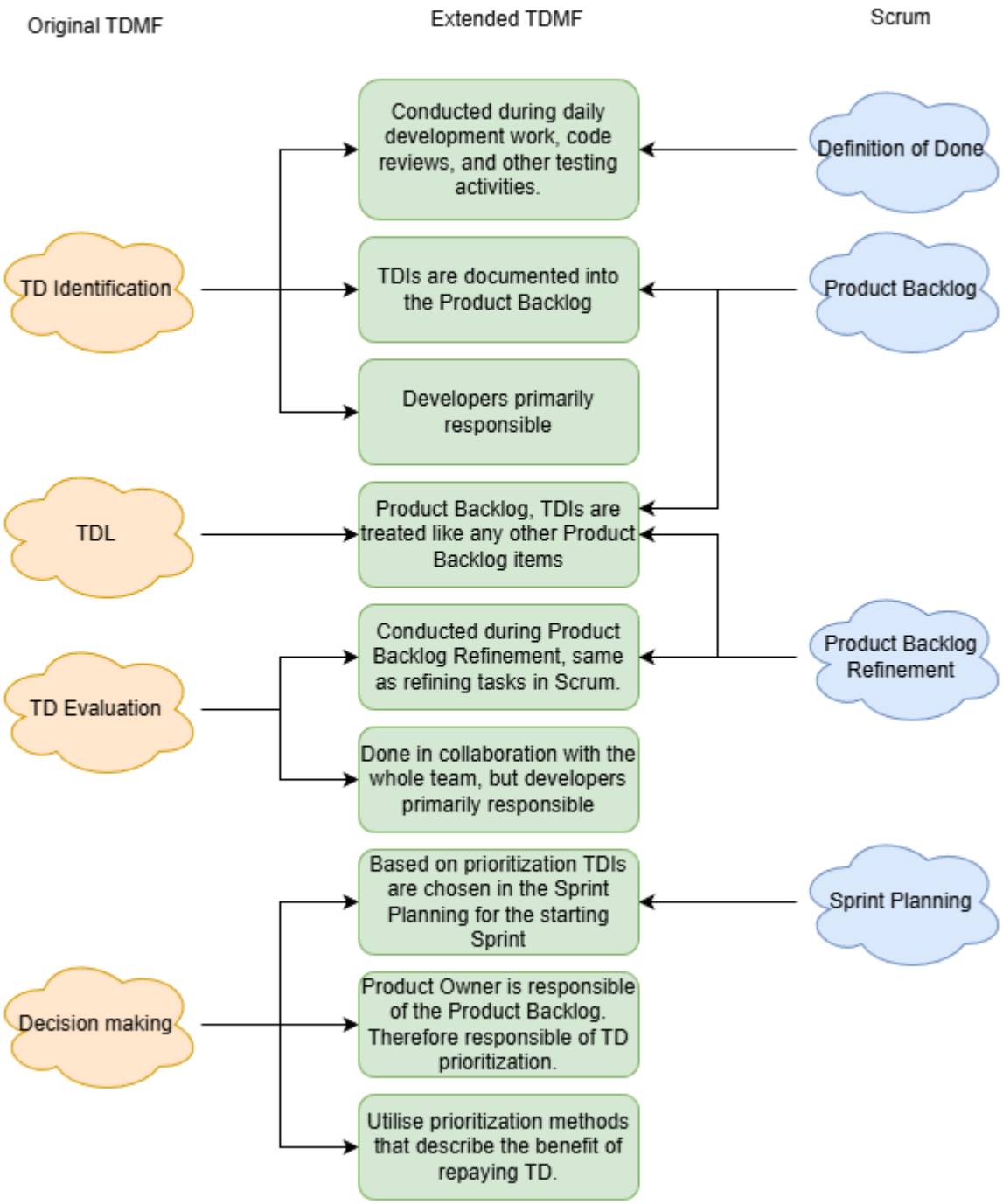


Figure 4. Extended TDMF and its relation to Original TDMF and Scrum.

The arrows in Figure 4 illustrate how the core artefacts of the original TDMF are mapped to Scrum artefacts along with detailed clarifications in between the mappings. The direction of the arrows does not represent a sequential process but a structural alignment between the different artefacts.

In the proposed extension TD Identification is integrated into multiple Scrum activities. Developers identify TD during their daily development work, code reviews, and other testing activities. All TDIs are recorded in the Product Backlog following the same structure as in the original TDMF.

TD Evaluation is performed collaboratively by the development team. The evaluation focuses on estimating the principal, interest, and interest probability using coarse estimations (low, medium, high). Estimation techniques like effort-estimation in Scrum, such as planning poker can be used to support shared understanding and consensus.

Decision making and prioritization are aligned with Scrum's Product Backlog management. In the extended framework, TDIs are treated as backlog items that can be compared with other items such as user stories and bugs on the Product Backlog. To support this comparison prioritization methods based on cost-benefit analysis and especially ROI methods are emphasized. By expressing the benefit of repaying TD in terms of reduced future effort, the framework aims to make TD visible, understandable and quantifiable to non-technical stakeholders, particularly the Product Owner.

TD Repayment is then planned either by explicitly allocating capacity within Sprints or by scheduling dedicated TD items alongside feature development. Sprint Planning and Sprint Review serve as checkpoints for deciding whether and how TD is addressed within a given Sprint.

Scrum artefacts are utilised in this framework to manage TD in a reactive and pre-emptive fashion. The Product backlog acts as the primary coordination mechanism between business-driven work and TD repayment. The Sprint Backlog reflects short-term commitments that may include both feature work and TD items. The Definition of Done is used as a preventive mechanism to reduce inadvertent TD by enforcing quality related criteria such as code reviews, test coverage, static analysis, and programming standards.

Scrum roles have been given distinct responsibilities in this framework. Developers are primarily responsible for identifying, documenting, and estimating TD. The Product Owner is responsible for prioritization decisions, informed by the estimated impact of TD on future development. The Scrum Master supports the process by facilitating discussions around TD and ensuring that it is not systematically neglected due to short-term delivery pressure.

The extended TDMF represents a theoretical application model of TD management in Scrum-like Agile projects. While the model is grounded in existing literature and Agile practices, its practical applicability has not been previously evaluated in a real-world project. In the following sections we further refine this theoretical application model and in Chapters 4-6 we examine how this works in practice.

3.3 Learnings from Previous Applications of TDMF

In this section we review the main learnings from existing applications of the original TDMF and TDM related practices from the existing literature [17], [23], [25], [26], [27]. The reviewed studies include both direct applications of the original TDMF as well as closely related TDM practices aligned with its core concepts. The papers were found from www.webofscience.com using the following search string "Managing AND technical AND debt AND (Agile or Scrum)". The papers used to derive the learnings were selected based on them using the original TDMF in Agile context. Studies reporting empirical findings or practitioner-oriented insights related to TDM were prioritized. The learnings were derived from the papers' research findings, lessons learned and discussions sections.

Table 4. Learnings from existing literature.

Learning (L)	Description	References
L1. TD Identification must be collaborative.	TD Identification is not the sole responsibility of developers. Architects, testers, Product Owners, and even business analysts can contribute.	[17], [26]
L2. TD Measurement should be simple and practical.	Teams struggle to measure interest probability. Instead using only principal and interest is more feasible.	[26]
L3. Estimating TD using planning poker or story points improves clarity.	Using Planning Poker and Story Points improves engagement, transparency, and consistency of measurement.	[26]
L4. Visualizing TD improves awareness and decision-making.	Dashboards, TD trend charts, debt backlog, and burndown-style graphs help stakeholders understand impact and motivate repayment.	[23], [27]

L5. TD should be treated as backlog items.	Recording TD in the backlog helps teams prioritize TD alongside user stories.	[23], [27]
L6. Product Owners need visibility into TD Impact for buy-in.	When TD impact is visualized or quantified, Product Owners are more willing to allocate time and resources for repayment.	[17], [23], [26]
L7. Lack of time and short-term delivery pressure are the biggest barriers.	The most common impediment to TDM is schedule pressure and focus on short-term goals.	[25], [27]
L8. Dedicated time improves TD Management.	Allocating fixed capacity for example 20% from a Sprint improves long-term TD reduction.	[23], [25]
L9. Mindset and awareness are critical success factors.	TDM fails when teams lack awareness, interest, training or leadership support.	[27]
L10. Integration with Scrum ceremonies is essential.	Integrating TD activities into Scrum events, especially Sprint Planning, improves accountability.	[23], [26]
L11. Tool support enables monitoring and tracking.	Using tools like Jira, Trello and excel improves visibility, tracking and decision-making.	[23], [27]
L12. TDM should be continuous not reactive.	Effective TDM means monitoring, reassessing, and updating TD status regularly.	[26], [27]

Based on the learnings identified (Table 4), a set of design principles (Table 5) was derived to guide the refinement of the extended TDMF. Each design principle represents an abstraction of one or more recurring themes observed across the reviewed papers. The goal of these principles is to translate empirical insights and learnings into actionable guidance that informs how to further refine the extended TDMF.

The learnings were grouped based on conceptual similarity. These groups were generalized into design principles that capture practical requirements for TDM in Agile context. Therefore, each design principle is grounded in one or multiple empirical findings.

Table 5. Design principles derived from learnings.

Design principle (DP)	**Description.**	**Derived from learnings**

DP1. TDM must be collaborative and role inclusive.	Since TD can be identified by multiple roles TD identification must be a shared responsibility.	L1
DP2. Measurement must be simple, consistent and effort based.	Use only principal and interest and not interest probability. Utilise simple estimation techniques: planning poker, Fibonacci scales, or t-shirt sizing.	L2, L3
DP3. Visualization is important for awareness, buy-in, and decision making.	Utilize visual dashboards to make debt amount and debt evolution visible over time. Aggregate principal and interest.	L4, L11
DP4. TD must be represented in the backlog as actionable work.	TDIs must be added to the Product Backlog so it can be tracked and prioritized the same way as other items on the backlog.	L5, L7, L9
DP5. Product owners require clear justification using business language.	Utilise TDMF and especially the TDI concept. Quantify technical debt using principal and interest.	L6, L9
DP6. Debt budgeting. Time and capacity must be allocated to managing debt.	Dedicate a fixed capacity for managing debt, for example 20% of Sprint.	L8, L12
DP7. TDM should be built into Scrum ceremonies.	Instead of treating TDM as additional work, integrate it into existing Agile processes.	L5, L10, L11, L12
DP8. Tool support is essential but should stay lightweight.	Visualization, monitoring and TD documenting is much easier with proper tool support, existing issue tracking software should be utilized if possible to keep the tooling lightweight.	L4, L5, L11

3.4 Evaluating Prioritization Methods

The TD prioritization methods mentioned in Section 2.5.4 address the difficulty in communicating TD between the developers and the PO. The PO cannot prioritize work items based on the amount of work it takes to accomplish. Prioritizing based on the interest is a better approach, as that would mean that repaying TD would save us from paying the interest, but it does not compare the interest to the principal. Therefore, the “high principal first” and “high interest first” methods are not suitable prioritization methods as they do not quantify the

benefit of repaying TD. The better approaches would be the “cost-benefit analysis” and “return on investment” (ROI) methods as they consider both principal and interest and compare them together, which helps the PO and the team to find the TDIs that offer the most benefit when repaid.

“Cost-benefit analysis” produces the interest to principal ratio based on which we can find the TDI that will save us the most time when compared to the amount of work it takes to repay. ROI approach produces the difference between principal and interest based on which we can find the TDIs where we will save the most time when repaid. Both approaches act mostly the same way, but in Scrum context the ROI approach is preferred due to it being simpler to understand from the point of view from other stakeholders. For example, in cost-benefit analysis the team would have to explain what the ratio is calculated from, whereas in the ROI approach the result of calculating will give us the ROI or in other words the time we are winning.

3.5 Refined TDMF

Based on the challenges identified in Section 3.1, the extended TDMF presented in Section 3.2, and the synthesized learning and design principles derived from prior literature in Section 3.3, this section proposes refined TDMF (Figure 5) which includes refinements to the extended TDMF. The goal of these refinements is to strengthen the framework’s practical applicability. This improved version retains the core activities of the extended TDMF but introduces clarifications and constraints that align the framework more closely with empirical insights from prior studies.

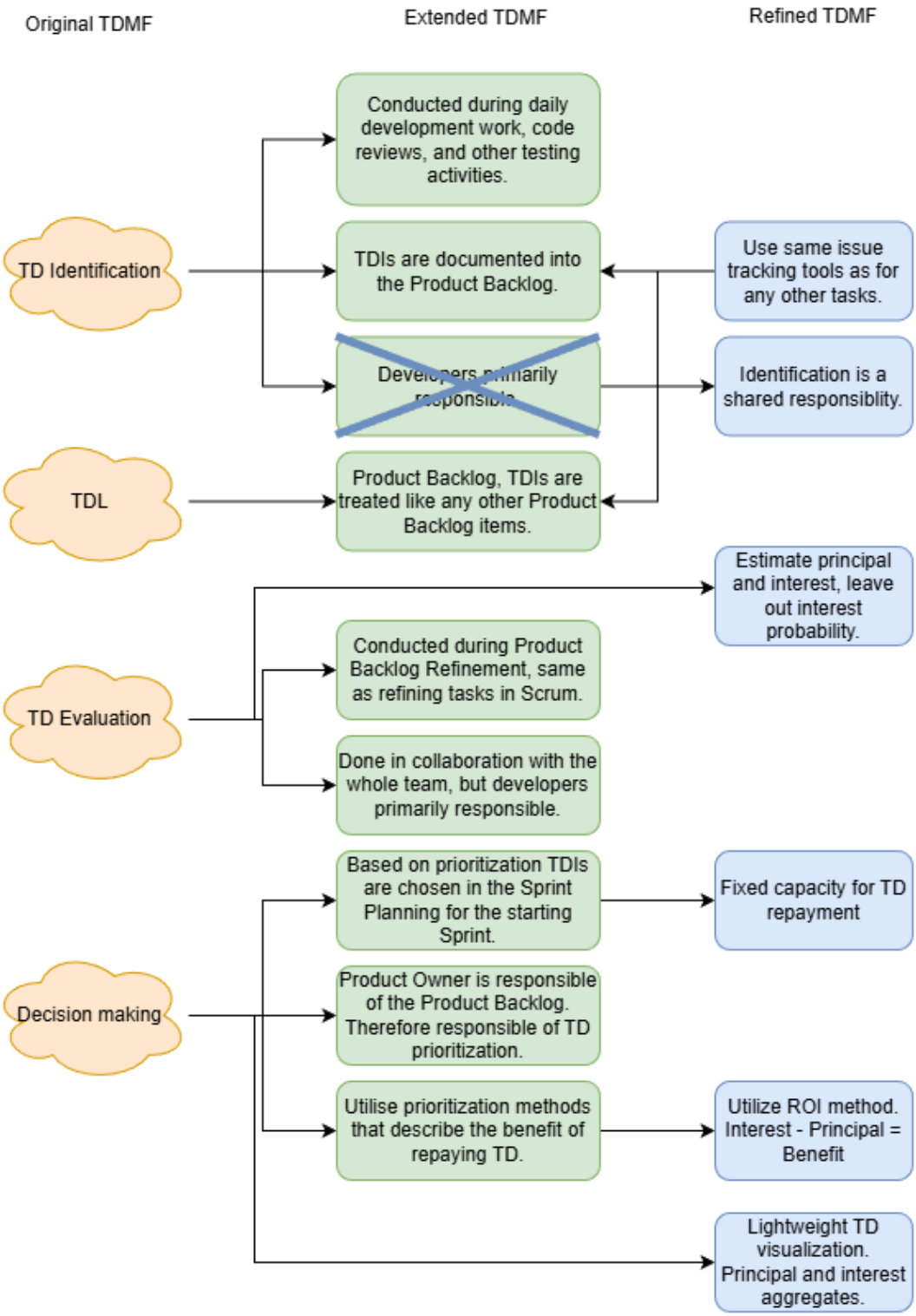


Figure 5. Refined TDMF, evolution from extended TDMF.

The changes in Figure 5 are derived from the design principles formed in Section 3.3. While the extended TDMF primarily focused on mapping the original TDMF artefacts to Scrum events and artefacts, the refined TDMF incorporates additional structural clarifications which

utilize these principles. For example, the emphasis on collaboration and role inclusivity (DP1) is reflected in explicit framing of identification and evaluation as shared team responsibilities. The simplification of evaluation practices (DP2) is visible in the removal of interest probability. In this way, the refined TDMF represents a design-principle-driven evolution of the extended framework, grounded in insights from prior literature.

3.5.1 Continuation from the Previously Proposed Extended TDMF

The refined TDMF still contains the same core structure of the extended TDMF. In particular, the fundamental idea of embedding TDM activities into existing Agile practices remain unchanged. Identification, evaluation, prioritization, and repayment of TD are still aligned with recurring development activities rather than treated as separate or exceptional tasks. Scrum events continue to serve as the development activities where TDM is embedded. Sprint Planning supports prioritization and capacity allocation; Sprint Retrospectives provide a forum for reflection and identification of new TD items. In the same way Scrum artefacts such as the Product Backlog and Definition of Done remain central mechanisms for coordinating TD with feature development and preventing unnecessary debt accumulation.

3.5.2 Collaborative and Continuous TD Identification

In line with DP1 and DP7, TD Identification is treated as a continuous and collaborative activity embedded into existing Scrum ceremonies rather than as a separate process. Developers, testers, and other technical roles are responsible for identifying TD during daily development work, code reviews, and testing activities. Sprint Retrospectives serve as a checkpoint for reflecting on technical issues and even documenting newly identified TDIs.

All identified TDIs are recorded in a centralized backlog, preferably using existing issue-tracking tools to minimize overhead and ensure visibility (DP4, DP8).

3.5.3 Simplified Evaluation and Estimation

Following DP2, the refined framework emphasizes simplicity and consistency in TD Evaluation. Instead of estimating interest probabilities which has been shown to be difficult in practice, the evaluation focuses on two dimensions: principal and interest. Both dimensions are estimated using familiar Agile techniques such as planning poker, story points, or t-shirt sizing. This approach improves shared understanding, reduces cognitive load, and aligns TD Estimation with existing team practices.

3.5.4 Prioritization Based on Return on Investment

Building on the analysis in Section 3.4 and DP5, the refined framework adopts ROI as the primary prioritization mechanism for TD. By comparing the estimated interest savings to the principal required for repayment, the ROI approach enables meaningful comparison between TDIs and feature-related backlog items.

This framing allows TD to be discussed using business-relevant language, supporting informed prioritization decisions by the Product Owner and increasing organizational buy-in.

3.5.5 Planned and Budgeted TD Repayment

Consistent with DP6, in this framework allocating dedicated capacity for TD repayment is explicitly recommended. This may take the form of a fixed percentage of Sprint capacity or explicit TDI planned alongside feature work during Sprint Planning.

Sprint Retrospectives act as feedback loops, enabling the team to assess whether TD repayment efforts are sufficient and to adjust future allocations accordingly. To help with this assessment visualization and monitoring is recommended, which is discussed in the next section.

3.5.6 Visualization and Monitoring

To support DP3 and DP11, the refined TDMF emphasizes lightweight visualization of TD. Aggregated views of TD principal and interest over time, such as dashboards or backlog summaries, are recommended to support awareness, transparency, and long-term monitoring and decision-making.

Visualization supports continuous reassessment of TD and reinforces the notion that TDM is an ongoing activity rather than a one-time clean-up effort (DP12).

3.5.7 Applicability Beyond Strict Scrum Implementations

Although Scrum events and artefacts are used as illustrative anchors, the extended TDMF and its refined version, the refined TDMF, can be used in Scrum-like Agile projects rather than strictly conforming Scrum implementations. In practice, many Agile teams adapt, combine, or omit formal Scrum events or artefacts while retaining iterative development, regular planning, and continuous improvement.

In contexts where formal Scrum events are absent or modified, the framework can be applied by mapping TDM activities to functionally equivalent practices. For example, prioritization decisions may be made during backlog refinement sessions or informal planning discussions rather than formal Sprint Planning. Similarly, review and monitoring activities can be integrated into release reviews, milestone meetings, or continuous delivery feedback loops.

By focusing on the intent and function of Agile practices rather than their formal labels, the framework remains adaptable to a wide range of Agile working models. This flexibility is particularly important for long-lived projects, where process evolution and hybrid Agile practices are common.

3.5.8 Example of TDI Lifecycle

To illustrate how the refined TDMF operates in practice, this section presents an example of TDI lifecycle (Figure 6), from initial identification to repayment, following the principles and processes described above.

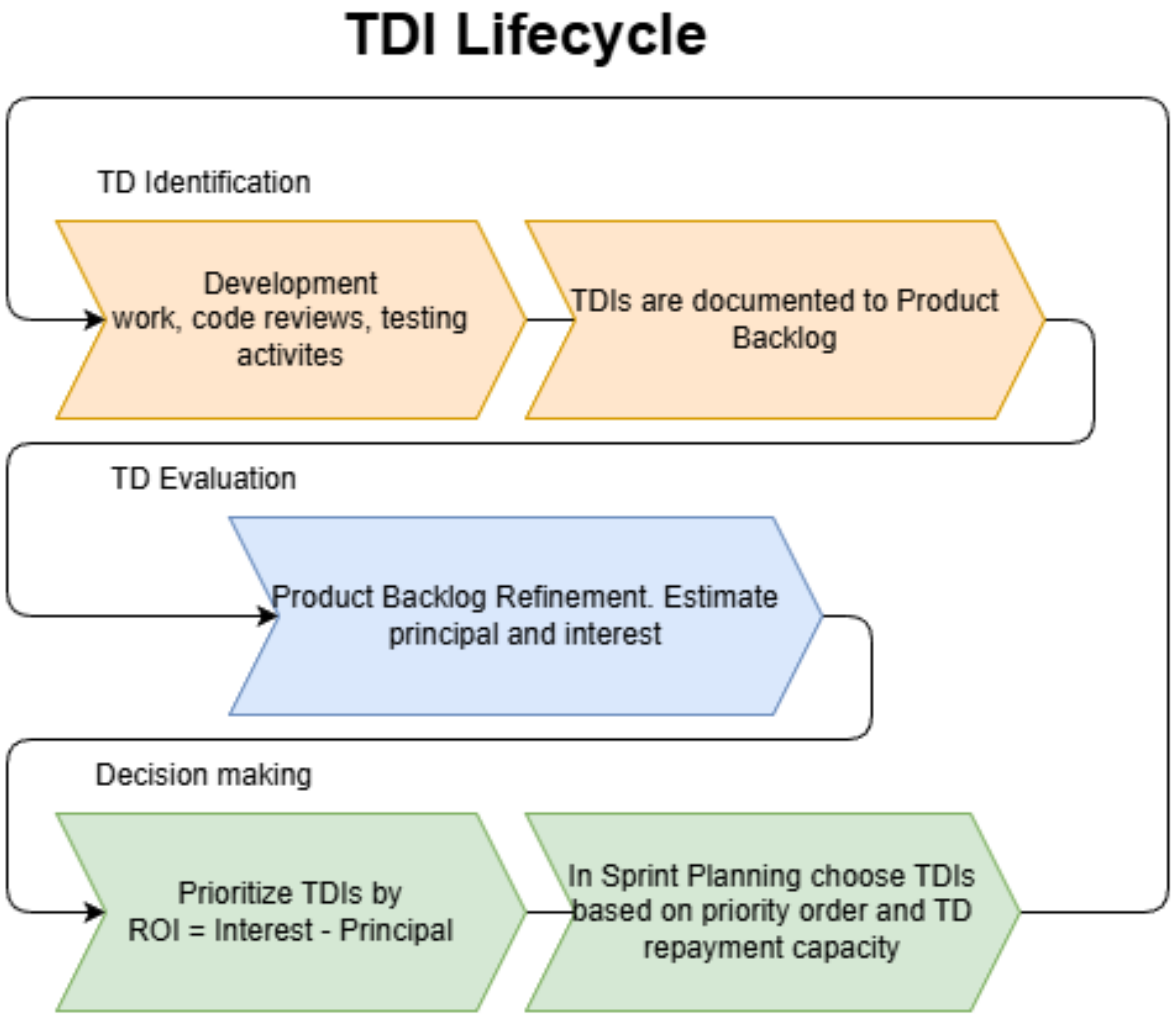


Figure 6. TDI Lifecycle in Refined TDMF.

During the Sprint's development work a developer notices that a core software class contains duplicated validation logic that has evolved inconsistently over time. This increases maintenance cost and has the potential to cause defects if changes are applied unevenly. The issue is discussed informally during a code review and later raised in the Sprint Retrospective, where the team agrees that this is an issue. Following the refined TDMF, the identified TDI is documented in the Product Backlog. The TDI includes a short description of the problem and any other necessary contextual information. During the next Product Backlog Refinement, the team evaluates the TDI. The principal is estimated as the effort in workdays required to refactor the duplicated validation logic into a single shared software component. The interest is estimated as the expected future costs, such as possible bug fixes and slower implementation of features relying on the validation logic. The team utilises planning poker and estimates the principal as two story points and the interest as three story points. Based on

the estimates the team calculates the ROI as one. The Product Owner uses this information to compare the TDI with other backlog items and prioritises this TDI sufficiently high to be considered already on the next Sprint Planning. During the Sprint Planning the team reviews the prioritized Product Backlog. Following the refined TDMF's recommendation for debt budgeting a fixed portion of Sprint capacity is reserved for TD repayment. The TDI is chosen for the Sprint alongside feature work. The refactoring is implemented during the Sprint and the duplication is fixed. Following the Definition of Done the team makes sure that tests are updated and that no new TD is introduced and any identified TD is identified and documented. In the Sprint Retrospective, the team reflects on the outcome. They note reduced complexity in the affected module. Aggregated visualization of TD principal and interest is reviewed from the team's issue tracking system. The team concludes that the current budgeted capacity for technical debt repayment is sufficient, while acknowledging the need for continued monitoring of principal and interest.

4 Research Methodology

This chapter presents the methodology used to address the research questions defined in Chapter 1. The chapter starts by describing the design of the study (Section 4.1), the case organization and project context (Section 4.2) followed by the description and rationale behind the data collection methods (Section 4.3), the data analysis procedures (Section 4.4), researcher positionality and reflexivity (Section 4.5), and ethical considerations (Section 4.6).

This thesis adopts a **case study design** [8], [9], [28], as the goal is to explore how the refined TDMF can be applied and experienced in a real-life context within a software development team. The study combines qualitative and quantitative methods to obtain a comprehensive understanding of the refined TDMF's applicability, benefits and challenges. Qualitative data was collected through focus groups, observation, and structured interviews, while quantitative data was collected using questionnaires before and after the application of the refined TDMF.

4.1 Case Study Design

A **case study design** is well suited for investigating complex phenomena in their real-world contexts [8]. The purpose of this study is to explore how the refined TDMF can be applied in a long-lived Agile software project and to evaluate its perceived usefulness among the development team. Because the goal is to understand the *how* and *why* of applying the framework rather than to test hypotheses or measure performance quantitatively, a case study approach provides the necessary depth and contextual understanding [9], [28]. As the refined TDMF has never been applied in a real-world setting, the case study can be classified as an exploratory **case study** [8].

To obtain a comprehensive view of the case, the study uses a **mixed-method approach** [8] combining qualitative and limited quantitative research methods. Qualitative methods [8]: focus groups, direct observation, and structured interviews, are used to explore framework application, team interactions, perceptions, and practical challenges. Quantitative data is collected through pre- and post-implementation questionnaires [8] completed by all team members to capture changes in their perceptions of TDM effectiveness. As the small sample size is small it does not work well with statistical inference, but these structured responses provide complementary evidence that supports the qualitative findings.

This case study approach allows for **methodological triangulation** [8], combining the findings from the qualitative and quantitative methods. By combining participatory collaboration, observed behaviour, participant feedback, and structured perception data, the study aims to form a holistic understanding of how the refined TDMF performs in a real-world long-lived Agile project.

4.2 Subject Organization and Project Context

The case study was done in a software development team of a medium-sized Finnish software organization that develops and maintains a two-decade-old software project written in PHP and JavaScript. The software development team follows Agile principles in its software development but does not strictly adhere to a single Agile framework, instead their practices can be described as stripped-down Scrum. The team utilizes a Product Backlog and separate Kanban board to which the team pulls tasks from the Product Backlog. Regular Product Backlog refinements are kept like Scrum. In comparison to Scrum the team does not have a dedicated Scrum Master or a Product Owner, instead the technical lead handles these responsibilities. In addition, the team does not utilize Sprints, therefore they do not also utilize Sprint Reviews or Sprint Retrospectives, instead review meetings and retrospectives are held on a need basis.

The specific focus of this study is the two-decade-old software project, which serves as the core system and product of the organization. Over its lifespan, the software has evolved through multiple generations of developers, technologies, and architectural decisions which show heterogeneous architectural patterns, coding practices and technology decisions. As a result, the system contains a significant amount of untested legacy code and a significant amount of technical debt in addition to large instances of technical debt, all of which present challenges for maintainability, scalability, and new feature development.

The development team responsible for the project consists of 4 software developers and 1 technical lead. The team is mostly self-organizing with the technical lead responsible for backlog prioritization and facilitating day-to-day, in addition the team collaborates with a product owner for upstream development tasks, a product designer for the product’s user interface and user experience, and technical support representatives for reporting bugs and prioritizing them.

While the team has an established workflow for managing defects and new features, systematic TDM has not been a formalized practice, although the team is aware of the significant amount of technical debt and has managed to document some large technical debt items. Technical debt items are typically identified informally during regular development or maintenance work. These conditions make the case organization a suitable environment for studying how a structured approach for TDM–such as the refined TDMF–can be integrated into the team's processes.

Although the refined TDMF uses Scrum events as illustrative anchors, in this case study the framework was applied by mapping its activities to functionally equivalent Agile practices used by the team, such as Product Backlog Refinement and ad hoc planning meetings.

4.3 Data Collection Methods

Multiple data collection methods were used to gain a comprehensive understanding of how the refined TDMF was applied and experienced in the case organization. The data was collected through focus groups, observation, interviews, and pre- and post-implementation questionnaires across the three phases of the case study (see Chapter 5). Using multiple sources of data is allowed for methodological triangulation [8].

4.3.1 Focus Groups

Focus groups were used for examining how the refined TDMF could be applied in practice within the case organization, directly addressing RQ1. They also contributed to RQ4 by enabling the identification of context-specific adjustments needed to better support Agile projects. The detailed focus group guide, including the questions used during the sessions is provided in Appendix 1.

The focus groups provided a structured setting for open discussion, enabling team members to reflect on how the theoretically proposed application of the refined TDMF aligns with their existing Agile practices. While the refined TDMF already maps TDM activities to common Agile events and artefacts, the purpose of the focus groups was not to redesign this mapping, but to contextualize it. Specifically, the discussions focused on identifying which elements of the proposed framework could be adopted as-is, which required adjustment, and where practical constraints or deviations from the theoretical model were likely to emerge in the case organization.

The focus groups followed a semi-structured format consisting of three phases. First, participants were asked to describe how TD was currently identified, evaluated, and managed in the project. This phase established a baseline understanding of existing practices. Second, the refined TDMF was presented by the researcher, including its core activities, and underlying design principles. In the third phase, the team examined how the proposed application of the refined TDMF aligns with their current Agile practices, identifying which elements could be adopted as-is, which required contextual adjustment, and where practical constraints or deviations from the theoretical model might occur. This discussion resulted in a context-specific application plan for operationalizing the refined TDMF within the team's existing process.

The focus groups were facilitated by the researcher and lasted approximately an hour. Discussions were transcribed using OpenAI's Whisper, an open-source automatic speech-recognition model run locally on the researcher's hardware; the resulting transcripts were verified by the researcher against the original recording. The gathered data provided valuable insight into the team's reasoning, decision making and understanding of TDM practices, and how a software team would apply the framework.

4.3.2 Observation

After the refined TDMF was introduced and adapted, the researcher conducted **participant observation** of the team's day-to-day development activities over a period of approximately one month. The purpose was to examine the applied refined TDMF and how it was actually used in practice and how it influenced the team's workflow and decision making. Observations focused on team discussions, backlog management and other situations where technical debt was identified or prioritized. Field notes were taken during the observation period, containing data about events and the researcher's analytical reflections. The observation data was later used to cross-check and contextualize the data gathered from interviews and questionnaires.

4.3.3 Interviews

Structured interviews were held at the end of the study to gain insight into the team members' individual experiences with the refined TDMF, directly addressing RQ2, RQ3 and contributing to RQ4. The interviews focused on participants' perceptions of overall experience, the framework's usefulness, impact on their day-to-day and challenges. The

interview questions followed a structured guide derived from the research questions. The detailed interview guide is provided in Appendix 2. Interviews were conducted individually, and lasted approximately 15-30 minutes, and were recorded with participants' consent, and the recordings were transcribed using OpenAI's Whisper run locally on the researcher's hardware, with each transcript verified by the researcher against the original recording. Transcripts were analysed thematically to identify common patterns and differences in perceptions.

4.3.4 Questionnaires

To complement the qualitative data, **short questionnaires** were used before and after the implementation of the refined TDMF. The questionnaires primarily addressed RQ3 by capturing perceived changes in how TD is managed during the study period, and they also supported RQ2 by providing insight into participants' attitudes toward TDM practices.

The questionnaires consisted mainly of Likert-scale questions and a few open-ended questions. The questions were derived from the research questions and focused on key aspects of TDM, such as perceived visibility of TD, clarity of prioritization, perceived ability to address TD, and the overall effectiveness of TDM practices. The pre-implementation questionnaire established a baseline, while the post-implementation questionnaire enabled comparison of perceived changes following the introduction of the refined TDMF.

While the small sample size does not enable statistical analysis, the quantitatively styled results provide structured evidence of the change in perception over the case period. These findings are used to support and triangulate the qualitative results from interviews and observations.

The questionnaire items are provided in Appendix 3.

4.4 Data Analysis Methods

A combination of qualitative and quantitative analysis techniques [8], [9] was used to analyse the collected data. The aim was to provide a comprehensive understanding of how the refined TDMF was applied in the case organization (RQ1), how it was perceived by the development team (RQ2), and what benefits or challenges emerged during its application (RQ3). Analysis using multiple methods enabled triangulation of findings across different data sources [9].

4.4.1 Qualitative Data Analysis

Qualitative data consisted of focus group transcripts, observation notes, and interview transcripts. The analysis followed a **thematic analysis approach** [9], which enables identifying, organizing, and interpreting patterns within qualitative data. The analysis process involved multiple steps:

1. Familiarization: the researcher reviewed all textual data, including transcriptions and notes, to gain an initial understanding of the content.
2. Coding: meaningful segments of text related to TDM practices, framework application, perceived benefits, and challenges were coded.
3. Theme development: codes were grouped into broader themes that represented recurring ideas or issues across data sources.
4. Refinement: themes were reviewed to ensure consistency and to align with the research questions.
5. Interpretation: the final themes were interpreted considering the theoretical background and the objectives of the study.

Thematic analysis was chosen because it is flexible and suitable for exploring experiences and perceptions across multiple data sources. This method allowed the researcher to capture explicit opinions and implicit meaning related to the refined TDMF.

4.4.2 Quantitative Data Analysis

The quantitative part of the analysis contains responses to the pre- and post-implementation questionnaires. The small number of participants limits the use of statistical tests; therefore, the analysis focused on descriptive comparison of individual and group level responses. For each Likert-scale item, pre- and post-implementation averages were calculated to identify directional changes in perceptions of how well TD is managed. Changes were visualized on a simple table. The purpose of this quantitative analysis was not to draw statistical conclusions but to provide structured measurable evidence that supports or contrasts the qualitative insights. This mixed approach strengthens the overall findings.

4.4.3 AI-Assisted Analysis

To support the efficiency of data analysis, an agentic AI tool, Claude Code by Anthropic, was used during selected stages of both the qualitative and quantitative analysis. Claude Code is a command-line interface tool that provides an interactive AI assistant powered by the Claude large language model. The tool operates on the researcher's local file system, enabling it to read research files, write and execute analysis scripts, and process textual data within a single context window. The model used within the Claude Code was Opus 4.7 with 1M context, a so-called thinking model with a larger context window, which is better suited for analysis work compared to other models.

The tool was positioned as a research assistant rather than an analyst: the AI proposed codes, computed survey result statistics, and generated cross-references, while the researcher validated all outputs, made interpretive decisions, and retained authority over all analytical claims. For quantitative analysis, Claude Code generated and executed Python scripts to compute descriptive statistics from the questionnaire data; the outputs were spot-checked manually by the researcher. For qualitative analysis, the researcher first independently familiarized themselves with each transcript without AI involvement to avoid anchoring bias. Claude Code then proposed initial codes with line-level references to the source text. The researcher reviewed every proposed code against the source, accepting, modifying, or rejecting each based on their reading and contextual knowledge of the case organization. New codes that the AI had not identified were added by the researcher. Theme development, interpretation, and the writing of analytical narratives remained entirely the researcher's work.

The use of AI in analysis was disclosed to participants through an amendment to the original consent form (see Section 4.6.1). All research data was anonymized before being processed by the tool. A detailed description of the human-AI collaboration workflow is provided in Appendix 4.

4.5 Researcher Positionality and Reflexivity

Reflexive thematic analysis [29] treats researcher subjectivity not as a bias to be eliminated but as a resource that shapes interpretation, alongside whatever constraints the researcher's position imposes. The researcher's position in this study requires explicit acknowledgement on three dimensions.

First, the researcher is a member of the case organization but not of the case team. The researcher works as a developer in a different team within the same organization, on a different codebase. Two of the five study participants were previous teammates of the researcher in another team approximately one to two years before this study; with all five, the researcher is familiar through ongoing collegial contact in the same organization. There is no reporting line between the researcher and any participant. This near-insider position carries less direct stake in the case team's output than a fully internal observer would and lacks day-to-day operational knowledge of the team's specific codebase or workflow rhythms.

Second, the researcher is the designer of the refined TDMF being evaluated. This is the strongest concern raised by the position. Participants were asked, by the framework's author, to evaluate the framework. Findings sympathetic to the framework therefore require additional scrutiny.

Third, the researcher facilitated the focus group and conducted all five post-implementation interviews. Co-worker status with participants creates real but bounded social pressure on candour.

Two features of the data partially mitigate these constraints. Participants did dissent in places where dissent mattered: the team's collective rejection of principal-and-interest scoring at the first refinement was sustained without softening through to the second refinement and through all five interviews, indicating that scepticism toward the framework's central calculative mechanic was reportable to the framework's designer in person. More importantly, the recurring finding that the framework's value would be greatest as a tool for upward communication to management appears not only in interviews and observations but also in the anonymous written responses to POSTQ9, where one respondent independently called for an "acceptable debt threshold" mechanism. The convergence of an anonymous written channel with the interview-mediated finding strengthens the claim that this conclusion is the team's own, rather than an artefact of facilitator presence or framework-designer accommodation.

4.6 Ethical Considerations

This research followed ethical principles and guidelines of the University of Turku.

4.6.1 Confidentiality, Data Protection and Anonymity

- All personal data and identifiable information about participants and the organization were treated confidentially.
- Participants were anonymized in all transcripts, notes and reports.
- Any references that could have revealed a person's identity or sensitive corporate information were removed from published reports.
- Collected research data was stored securely on the University of Turku's cloud environment, and the access was restricted to the researcher and the supervisor.
- Collected research data was stored only as long as required by the research.

Considering the small size of the team that was participating in the research, there is a risk that people can be identified based on the context of what was said so extra care was taken to anonymize the participants in the research data.

The original consent form covered the use of an AI-assisted transcription tool for processing recorded discussions. As the analysis stage of the research subsequently expanded to include AI-assisted analysis (see Section 4.4.3), an amendment to the consent form was prepared and sent to all participants. The amendment described the use of Claude Code by Anthropic for data analysis, specified that all data would be anonymized before being processed by the tool, that the tool does not retain or learn from the research data, and that the researcher retains full control over which data is processed. All participants confirmed their consent to the amendment. When using the AI-assisted analysis tool, anonymized data is sent to Anthropic's servers for processing. Given the small team size and the indirect identifiability risk noted above, anonymization extends beyond participant names to include the organization name, product names, tool references, and other contextually identifying details.

4.6.2 Voluntary Participation

Participation in the case study was voluntary. Information sheet was provided to all the participants explaining the scope of the study, what participation entails, potential benefits or risks, the right to refuse or withdraw participation at any time, and how the data will be used, stored and published. Each participant signed, or accepted through email, a consent form agreeing to these terms, which they could withdraw at any point during the research, which would have led to all the data collected from them being excluded from the research.

4.6.3 Risks and Mitigation

As this study deals only with participants' work practices, experiences and opinions and not sensitive personal information, the risks to participants are minimal, but still potential risks and their mitigation include:

- Risk of discomfort or perceived criticism.
    - Participants may feel uncomfortable discussing difficulties or challenges in their workflows. This was mitigated by reassuring them that there are no right or wrong answers and that the data is confidential.
- Data breach.
    - This was mitigated by storing the collected data on University of Turku's cloud environment with only the researcher and supervisor having access.
    - In addition, all the participants and the organization were anonymized in the collected data.
- Unequal influence.
    - Because the researcher was part of the organization, work relationships and work environment dynamics may affect participant openness. This was mitigated by emphasizing voluntary participation, anonymity and that there were no repercussions for honest feedback.

5 Applying the refined TDMF

This chapter reports how the refined TDMF was applied in the case organization. It describes, in chronological order, the three phases of the case study: the pre-implementation phase (pre-survey and focus group), the one-month observation period during which the team trialled the refined TDMF, and the post-implementation phase (post-survey and interviews). The chapter focuses on what was done, the team's adaptation decisions, the actual application of the framework, and the deviations that emerged between the agreed plan and observed practice. Findings, themes, and the interpretive analysis are presented in Chapter 6; the integrated discussion in answer to the research questions is in Chapter 7.

As detailed in Section 4.5, the researcher held three concurrent roles during this case study: a member of the case organization working as a developer in a different team on a different codebase, the designer of the refined TDMF being evaluated, and the facilitator of both the focus group and the post-implementation interviews. The researcher had prior collaborative history with two of the five participants from a previous team approximately one to two years before the study and was familiar with all five as colleagues. The descriptive narrative below is presented in third person register accordingly.

5.1 Pre-implementation Phase

5.1.1 Pre-implementation Survey

The pre-implementation questionnaire (Appendix 3) was distributed to all five team members and remained open until the focus group session the following day. All five participants responded (n=5, 100% response rate). The survey collected baseline self-assessments on seven Likert-scale items measuring perceived TD identification, documentation, discussion, decision-making transparency, prioritisation, time allocation, and overall management. One open-ended question invited free-text reflections on the main challenges of TD management in the project. The survey was anonymous.

5.1.2 Focus Group Session

All five team members attended in a hybrid format (Google Meet plus in-person). The session was scheduled for approximately one hour as described in Chapter 4; in practice the discussion ran longer, lasting approximately two and a half hours, with the team continuing

engagement through Phase 3 of the guide. The session followed the three-phase guide in Appendix 1.

Phase 1 elicited the team's current TD management practices. Participants described TD as identified informally and tracked across multiple, partially overlapping locations: a shared technical-state document recording the platform's accumulated debt items, individual issue-tracker tickets, periodic third-party audit findings, automated dependency-vulnerability scanner outputs, and for many smaller items the team's collective memory rather than any explicit record. Decisions about when to address TD were described as predominantly reactive, driven by external triggers (audits, customer-reported issues, immediate business impact) rather than by systematic planning. Several participants noted that the team's capacity for dedicated TD work had declined over the previous year, attributing this to a strategic shift toward new feature development on the legacy platform after the wind-down of a parallel refactoring project.

In Phase 2, the researcher presented the refined TDMF, including its design principles, its mapping of TD activities to Agile events and artefacts, and an illustrated lifecycle of a TD item from identification through repayment. Participants engaged with the presentation through clarifying questions. The framework's calculative core, assigning principal and interest values to TD items as a basis for prioritisation, provoked the most discussion.

Phase 3 contextualised the refined TDMF against the team's specific way of working. Three discussion strands ran through the conversation. First, several participants felt the framework's structural elements largely matched what the team already did informally. The framework's value, in their view, would be in making implicit structure explicit. Second, the calculative layer of the framework was met with significant scepticism. One participant argued that principal/interest scoring would add limited value within a small co-located team that already shared tacit knowledge of its TD. The scoring's intended audience, the same participant suggested, was external stakeholders to whom the team currently had limited communication. Another participant partly disagreed, seeing principal/interest as useful at least for comparing TD items against each other when allocation decisions involved larger items. Third, participants raised concrete adaptation suggestions: simpler one-number metrics, an urgent-meaningful binary, and the cross-referencing of TD items with planned upcoming features as a roadmap-based prioritisation lens.

5.1.3 Application Plan

The team committed to trialling the refined TDMF for approximately one month following the focus group. The application plan, as agreed, comprised four operational decisions:

1. TD identification by every developer. Any team member who encountered TD in their daily work would document it as a Jira ticket. The team acknowledged that the existing backlog size was already at capacity, so creating new tickets for every observed TD item was not realistic in practice. Identification would be best effort within existing constraints.

2. Recurring TD discussion in the longer daily. The team's calendared "refinement" meeting, which functioned in practice as a longer daily, would serve as the forum for TD identification, evaluation, and team-level prioritisation. Discussions would occur approximately every two to three weeks, fitting the rhythm of the team's existing schedule.

3. Approximately 20% of backlog top items as TD. Every fifth ticket at the prioritised top of the backlog would be a TD item. This commitment was understood as aspirational, participants flagged that incoming SLA tickets would likely displace planned TD allocation in practice.

4. Visualisation of TD volume. The visualisation/aggregation step of the refined TDMF was acknowledged as part of the framework but not explicitly agreed as a team practice during the trial period.

A notable feature of the application plan, observable already at the focus group, was that the formal calculation of principal and interest values for individual TD items was not committed to. The team's stated reason was that the existing TD volume was so large that the more pressing first step was to begin repayment, with formal scoring to be considered if and when the backlog was reduced to a more manageable size.

5.2 Observation Period

The observation period comprised approximately one calendar month. The researcher attended the team's regular dailies and the longer-daily refinements as a silent participant observer. All sessions were hybrid. Of the eleven sessions scheduled in this period, ten were observed (one daily was missed by the researcher and is recorded as such in the field notes).

The first and second refinements (in the field notes) were substantively dedicated to TD management, while the remaining eight observed sessions were standard dailies attended primarily for context and continuity.

5.2.1 Kick-off Refinement

The first refinement following the focus group operationalised the application plan. After the team's regular daily agenda, time was made within the session to discuss the framework's adoption. The lead presented an inventory of the existing backlog: 106 work items in total, of which 31 were SLA cases, with technical-improvement and dependency-vulnerability epics already containing TD-relevant items.

A developer raised the question of whether TD items in the backlog should now be assigned principal and interest values. The team collectively decided not to adopt formal scoring at this stage, on the grounds that the existing volume of TD items was substantial and selecting items to address took precedence over evaluating them. The lead proposed a simpler effort-based heuristic — "if you see something doable in under a week, take it; if it's doable in under a day, take it" — and a strategy of preferring smaller items to clear the list and build momentum. Approximately ten TD items were selected and moved to the prioritised top of the backlog; some of these were pre-assigned to specific developers based on their familiarity with the relevant code areas. By the end of the session, roughly one in five items at the top of the backlog was a TD item, fulfilling the "approximately 20%" commitment structurally.

When the lead asked the team how the meeting felt, the responses ranged from cautious endorsement to a candid observation by one developer that the discussion had felt limited from their perspective because most of the items being prioritised did not relate to their current work. Another developer expressed agreement with skipping formal scoring, framing it as appropriate given the team's level of TD load. The lead articulated the role of providing "a debt sanctuary for developers so debt can be paid even though new projects are pushing already on devs" — a framing that recurred in subsequent sessions.

5.2.2 Mid-period Dailies and Observed Practice

Across the observed dailies, two regularities were observed. First, TD repayment did happen, but predominantly as a side-effect of feature work, SLA bug-fixes, and externally triggered security and accessibility audit findings, rather than as standalone work selected through the

framework's mechanisms. Second, the TD-specific discussion remained largely confined to the two refinement sessions held during the trial period, the first and second refinements.

Notable mid-period observations include at one daily, with approximately half the team on holiday leave, the team articulated that many of the SLA bug tickets being addressed were in their own assessment "debt items" caused by suboptimal implementations, although these tickets were not formally tagged under the technical-debt epic in the issue tracker. At another daily, a large-scope ticket entered development that the lead had not previously seen — approved by other management actors and other developers — illustrating that the team's distributed approval flow operated outside the framework's prioritisation channel. At one refinement session, the lead noted the backlog had shortened slightly (from 106 items at the first refinement to 105), a small but visible effect of the prune-and-prioritise mechanic agreed at the first refinement. At another daily, three to five prioritised TD items remained at the top of the backlog, indicating that the first refinement's "approximately one in five" allocation pattern was sustained across the trial period.

5.2.3 Closing Reflection

The second and last refinements included a deliberate reflection on the trial period, structured around metric review and team discussion. The lead presented metrics they had aggregated independently using filtered queries on the team's issue tracker: of all completed work in the four-week trial period, two tickets explicitly tagged under the technical-debt epic had been closed (2.1% of total completed work); broadening the definition to include TD-adjacent items not formally tagged under the TD epic but in the team's working understanding raised the figure to 13%; including SLA bug tickets in their entirety raised it to over 50%. The lead acknowledged that the team had likely not been consistent in tagging items under the TD epic during the trial.

The discussion that followed surfaced two recurring points from the team. First, formal evaluation of TD items via principal and interest had remained difficult and was acknowledged as not having been operationalised during the trial. Second, and articulated by multiple participants, the framework's effectiveness was, in their view, bounded by capacity: the framework could not by itself address the underlying constraint of insufficient time for TD work, and any meaningful change would require management-level allocation of bandwidth or relief from new feature development pressure. Several participants raised the suggestion that the framework's most-valuable role might be in communicating TD state and risk to

organizational leadership rather than in internal team coordination, a topic that is taken up analytically in Chapter 6.

5.3 Post-implementation Phase

5.3.1 Post-implementation Survey

The post-implementation questionnaire (Appendix 3) was distributed to all five team members, immediately following the second and last refinement. The survey replicated the seven Likert-scale items of the pre-implementation questionnaire and added two open-ended questions probing perceived improvements from the framework and challenges encountered during its application. Four of five team members responded; despite a reminder cycle over the subsequent week, one response was not received, yielding a response rate of 80%. As with the pre-implementation survey, responses were anonymous (timestamp only), so paired pre-post comparison at the individual level was not possible.

5.3.2 Post-implementation Interviews

Five semi-structured individual interviews were conducted after the trial period, one with each team member, following the interview guide in Appendix 2. Interviews were conducted in hybrid format, recorded with participant consent, and lasted between approximately 11 and 45 minutes. One interview terminated after approximately 11 minutes due to a recording failure; the interviewee's responses to interview themes 1 (overall experience) and 2 (perceived usefulness) were fully captured, and the opening of theme 3 (impact on day-to-day work) was partially captured before the recording was lost. Themes 4 (challenges and limitations) and 5 (improvement and reflection) were not captured for that participant. This data-coverage limitation is acknowledged in Chapter 7 and flagged in the Triangulation rows that draw on improvement-related interview content.

The interview transcripts were produced using OpenAI's Whisper (see Section 4.3 for the transcription procedure and Appendix 4 for the AI-assisted analysis pipeline), manually verified by the researcher against the recordings, anonymised (participant pseudonyms, organizational identifiers, and product names redacted), and prepared for analysis as described in Chapter 4.

5.4 Summary

The refined TDMF was applied with selective adoption: its structural elements — the recurring TD discussion forum, the team's commitment to a roughly 20% TD allocation, and the recording of TD items in the centralised issue tracker — were operationalised in the first refinement and partially sustained across the observation period. The team also enacted the framework's prioritisation intent through standard Scrum backlog ordering — top-of-backlog signalling for TD items — in lieu of the framework's ROI prioritisation step. Its calculative element — principal and interest scoring of TD items — was not adopted during the trial. Multiple alternative formulations were discussed in the focus group and at the first refinement (effort heuristics, low-medium-high triples, must-should-nice classifications, single-number scalars, and cross-referencing with planned features), but none was operationalised as a method; in actual practice, TD items were evaluated and prioritised case-by-case through team discussion of each item on its merits. By the second refinement, several participants had explicitly raised, and the lead reiterated, the suggestion that the framework's potential value might extend beyond internal team coordination to upward-facing communication with organizational leadership. The interpretive analysis of these events, including their relation to the four research questions, is presented in Chapter 6.

6 Results

This chapter presents the empirical findings of the case study, drawing on the five data sources described in Chapter 4 and the trial described in Chapter 5. Sections 6.1–6.2 present descriptive Likert findings ($n_{pre} = 5$, $n_{post} = 4$ unmatched, no inferential test; see Appendix 4). Sections 6.3–6.7 present five themes constructed from the qualitative data through reflexive thematic analysis, each with cross-source supporting evidence and, where applicable, a survey-side counterpart. Section 6.8 synthesises. One of the five post-implementation interviews was truncated by a recording failure after interview themes 1 and 2 (overall experience and perceived usefulness) had been fully captured; the truncated interview's content for these earlier themes contributes to RTA Themes T1 and T2, while cross-source analyses for RTA Themes T3, T4, and T5 draw on four of the five interviews (see Section 8.3 for the corresponding limitation entry). Participant quotations in this chapter are presented in English translation; transcription and verification procedures are described in Section 4.3. The four-week trial duration further bounds the claims this chapter can support to within-trial effects; sustainability or durability of any change beyond the observation window cannot be adjudicated from this design and is taken up as a limitation in Chapter 8.

The five themes are:

- T1 — the capacity ceiling
- T2 — from calculation to conversation
- T3 — visibility, not identification, is the gap
- T4 — the framework's untapped potential is upward-facing
- T5 — the substrate: legacy code and team structure shape what's possible

T1 through T4 are findings about how the framework was applied and what it produced; T5 is a contextual theme that frames the others as case-specific rather than universal. T1 and T3 connect directly to specific Likert items on the instrument and carry a survey-side counterpart subsection; T2, T4 and T5 do not, because the items they address (the team's calculative-versus-structural adoption pattern, the upward-facing register, and the codebase substrate) were not measured by the seven-item Likert instrument. The themes are presented descriptively in this chapter; interpretation in answer to the four research questions and

recommended adaptations is taken up in Chapter 7; limitations bearing on the analysis are addressed in Chapter 8. As is common in reflexive thematic analysis, theme construction was iterative: codes were generated, refined, merged, and in some cases rejected as the analysis cycle progressed (the supporting analysis log is summarised in Appendix 4). The five themes presented here are the constructions the analyst found most defensible against the data and the research questions, not an exhaustive enumeration of all patterns considered. In line with Braun and Clarke's framing of reflexive thematic analysis [29], coding and theme construction were conducted by a single analyst (the researcher); the analyst's position and its bearing on the constructions are discussed in Chapter 4's reflexivity subsection.

6.1 Pre-implementation Baseline

The pre-implementation survey was administered to all five members of the case team during the trial's opening, immediately after the focus group and before the four-week intervention window. The survey comprised seven five-point Likert items (1 = strongly disagree, 5 = strongly agree) covering identification, documentation, discussion, transparency of decision making, prioritisation, time allocation, and an overall self-assessment of TD management, followed by one open-ended item (PREQ8) inviting respondents to describe the main challenges they experienced in managing technical debt. The survey was distributed and returned anonymously, with only submission timestamps recorded. All five team members responded, yielding a 100% response rate at baseline ($n_{pre} = 5$).

Table 6 summarises the per-item descriptive statistics and the response distribution across the 1–5 scale.

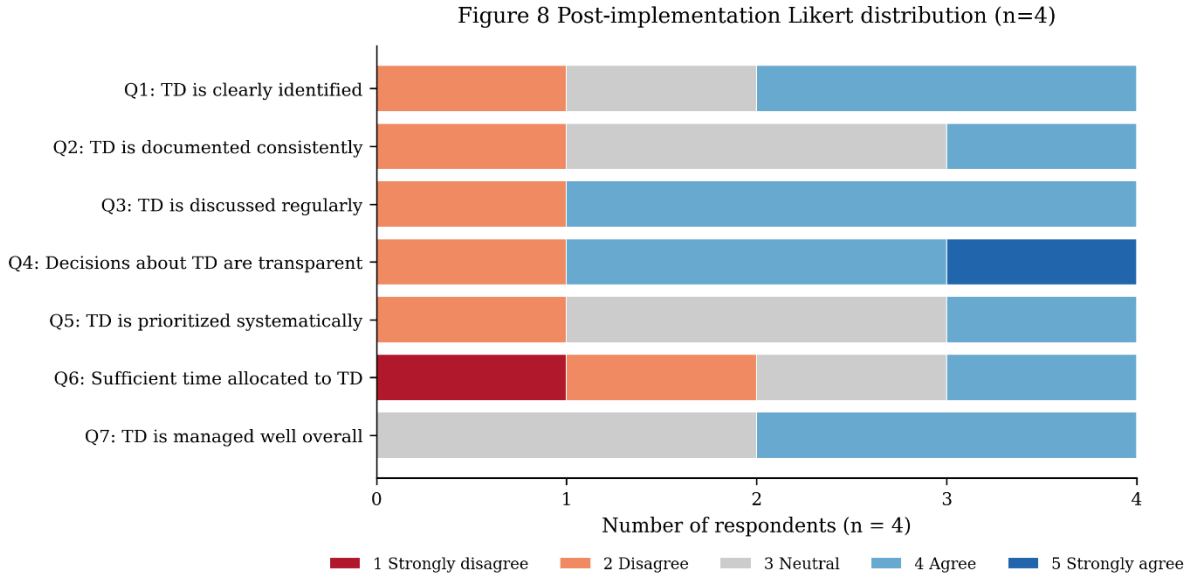
Table 6. Pre-implementation Likert results, n = 5. Distribution column lists counts in the order 1-2-3-4-5.

Item	Statement	Mean	Median	Std	Min	Max	Distribution (1-2-3-4-5)
PREQ1	TD is clearly identified	3.4	4	0.89	2	4	0-1-1-3-0
PREQ2	TD is documented consistently	2.4	2	0.89	2	4	0-4-0-1-0

PREQ3	TD is discussed regularly	3.0	4	1.41	1	4	1-1-0-3-0
PREQ4	Decisions about TD are transparent	2.6	3	0.55	2	3	0-2-3-0-0
PREQ5	TD is prioritized systematically	2.4	2	0.55	2	3	0-3-2-0-0
PREQ6	Sufficient time allocated to TD	2.4	2	0.55	2	3	0-3-2-0-0
PREQ7	TD is managed well overall	2.6	3	0.55	2	3	0-2-3-0-0

Three baseline patterns stand out. First, the highest-rated item was PREQ1 ("TD is clearly identified", mean 3.4, median 4), which was also the only item to attract a majority of responses at the agreement end of the scale (three "agree" responses out of five). Second, three of the seven items clustered at a mean of 2.4 — PREQ2 (documentation), PREQ5 (prioritisation), and PREQ6 (time allocation) — placing them below the scale midpoint. Together with PREQ4 (transparency, 2.6) and PREQ7 (overall management, 2.6), this means five of the seven items received baseline means below 3.0. The pattern is consistent with a self-assessment in which the team perceived itself to be reasonably aware of technical debt while rating the surrounding processes — recording it, prioritising it, and reserving capacity to act on it — less favourably. Third, PREQ3 ("TD is discussed regularly") returned the highest item-level variance (std 1.41), with responses split across 1, 2, and 4 (one "strongly disagree", one "disagree", three "agree"). The polarised distribution on this item indicates that respondents did not converge on a shared view of how regularly TD was being discussed at baseline, in contrast to the tighter clustering observed on PREQ4–PREQ7 (std 0.55 each). Figure 7 visualises the full distribution.

Figure 7. Pre-implementation Likert response distribution (n=5)



The five free-text responses to PREQ8, which asked respondents to describe the main challenges in managing TD, mapped onto three categories. The dominant category was time and resource constraints, named explicitly by Respondents 1, 2, and 4. Respondent 4 answered with the single word "time"; Respondent 2 wrote "Time and resources. Always creating new features."; and Respondent 1 elaborated that the team had "recognized and documented debt items", but that work was "addressed mainly when there's immediate business impact for it for compliance matters," with other items "categorized to lower priority." A second category, codebase scale, was represented by Respondent 3, who identified "the size of the codebase" as the principal challenge. A third category, lack of shared understanding, was articulated by Respondent 5: "I feel like everyone has a different understanding of what tech debt we have and whether it needs fixing and when." The categorisation of these open-ended responses is consistent with the items that received the lowest Likert means: time and resourcing concerns align with PREQ6 (mean 2.4); the documentation and shared-understanding concern aligns with PREQ2 (mean 2.4); and the implicit prioritisation logic described by Respondent 1 aligns with PREQ5 (mean 2.4). The codebase-scale concern raised by Respondent 3 sits outside the seven instrument items and is recorded here as a contextual observation.

6.2 Post-implementation Findings and Pre/post Comparison

The post-implementation survey was distributed after the last refinement at the end of the 4-week observation period, replicating the seven Likert items from the baseline instrument verbatim and adding two new open-ended items: POSTQ8, asking in what ways (if any) the

refined TDMF had improved TD management, and POSTQ9, asking what challenges or limitations the respondent had experienced when using it. The instrument retained the anonymous, timestamp-only design of the baseline. Four of the five team members responded after one week of reminders, yielding a response rate of 80% ($n_{post} = 4$); one team member did not respond. Because both surveys were anonymous, responses cannot be paired across the two waves, and the comparison reported here is therefore strictly group-level.

The figures in Table 7 are descriptive directional indicators only. With $n_{pre} = 5$ and $n_{post} = 4$ unmatched, no inferential test was conducted, in line with the analytical workflow specified in Appendix 4. Items are ordered by absolute change (Δ) descending so that the largest movement leads. The post-implementation distribution is visualised in Figure 8 and the pre/post mean comparison in Figure 9.

Figure 8 Post-implementation Likert distribution (n=4)

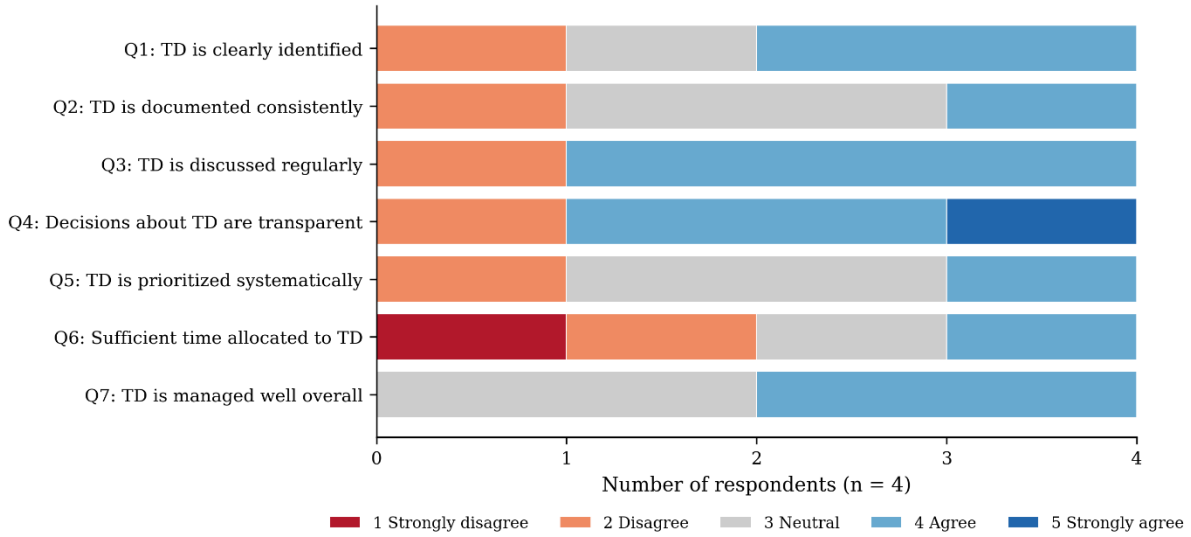


Figure 8. Post-implementation Likert response distribution by item (n=4). Stacked bars show counts of responses on the 1–5 scale.

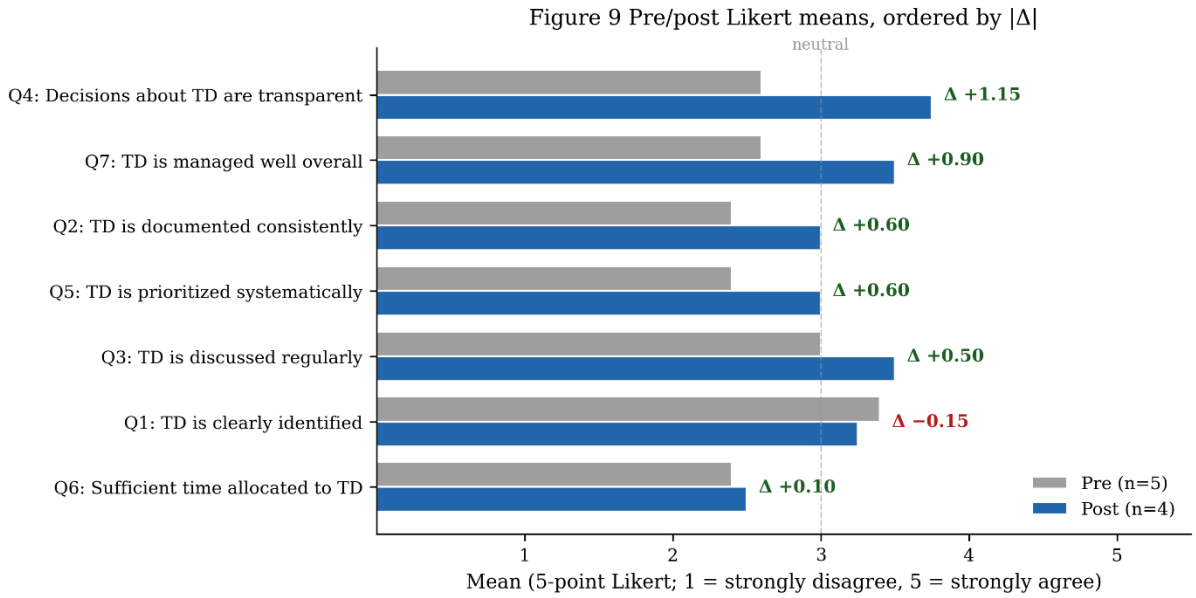


Figure 9. Pre/post Likert means, ordered by absolute change (Δ) descending. Pre-implementation values from n=5 unmatched responses; post-implementation values from n=4 unmatched responses.

Table 7. Pre/post comparison of Likert items, ordered by |Δ| descending. n_{pre} = 5, n_{post} = 4, unpaired.

Item	**Statement**	**Pre Mean**	**Post Mean**	**Δ**	**Pre Std**	**Post Std**	**Pre Distribution (1-2-3-4-5)**	Post Distribution (1-2-3-4-5)
Q4	Decisions about TD are transparent	2.6	3.75	+1.15	0.55	1.26	0-2-3-0-0	0-1-0-2-1
Q7	TD is managed well overall	2.6	3.5	+0.90	0.55	0.58	0-2-3-0-0	0-0-2-2-0
Q2	TD is documented consistently	2.4	3.0	+0.60	0.89	0.82	0-4-0-1-0	0-1-2-1-0
Q5	TD is prioritized systematically	2.4	3.0	+0.60	0.55	0.82	0-3-2-0-0	0-1-2-1-0
Q3	TD is discussed regularly	3.0	3.5	+0.50	1.41	1.00	1-1-0-3-0	0-1-0-3-0

Q1	TD is clearly identified	3.4	3.25	-0.15	0.89	0.96	0-1-1-3-0	0-1-1-2-0
Q6	Sufficient time allocated to TD	2.4	2.5	+0.10	0.55	1.29	0-3-2-0-0	1-1-1-1-0

Six of seven items moved positively. The largest shift was on Q4 (transparency, 2.6 → 3.75, Δ = +1.15), which also widened in dispersion (std 0.55 → 1.26): one respondent moved to the top of the scale where there had been none at baseline, but responses spread further apart (Figure 9). Q7 (overall management, +0.90) and Q2/Q5 (documentation, prioritisation, +0.60 each) form a secondary cluster. Q3 (discussion) rose by +0.50 and tightened (std 1.41 → 1.00). Q6 (time) was essentially flat (+0.10), and Q1 (identification) slipped marginally (−0.15). The pattern indicates directional improvement on the process-oriented items, no change on the resource item, and a small downward movement on identification — returned to in Section 6.5.

The four POSTQ8 responses to "in what ways, if any, did the refined TDMF improve TD management" categorised into a single dominant theme (visibility, discussion, and awareness of TD) named by three of the four respondents, with one non-response (Respondent 4). Respondent 1 wrote that the framework "opened discussions of TD" and that "the team is more conscious of TD." Respondent 2 framed the same observation more cautiously: "It brought the technical debt more visible to discussions in team level, at least temporarily." Respondent 3 described an awareness shift directed inward, noting that the framework made them "more aware of our handling of the tech debt, and the insufficient amount of resources that are allocated to it." The three responses thus converge on visibility/awareness as the primary perceived improvement, while two of them (Respondents 2 and 3) qualify the gain — Respondent 2 with respect to its sustainability, Respondent 3 with respect to the resource shortfall it surfaced.

The three POSTQ9 responses to "what challenges or limitations did you experience when using the refined TDMF" categorised under a single theme of resourcing and capacity, with one non-response (Respondent 4). Respondent 1 named "limitation of resourcing" and "challenge of choosing the next item." Respondent 2 described the surrounding context in detail: "The amount of technical debt is so high and the resourcing in the team is limited. We have multiple ongoing new feature development projects at the same time which causes us to have limited possibilities to handle the technical debt or the debt payment feels slow," further

noting that "the work is reactive" and that "a systematic, more planned way to handle debt would be a better way." Respondent 3 raised a distinct framework-level limitation, calling for an "acceptable debt threshold" and writing that the framework "is not sufficiently opinionated on the amount of tech debt that is maintainable" and "should have some guiding principle telling us, and the management, what is acceptable and what amount will end up crushing the system unless handled." All three POSTQ9 respondents thus pointed at capacity, while R3 additionally pointed at a missing decision criterion within the framework itself.

The Q4 / Q6 split mirrors the open-ended pattern: visibility as primary improvement, resourcing as primary challenge — anticipating Themes T1 (Section 6.3) and T3 (Section 6.5). No interpretive claims are advanced here; the descriptive convergence is recorded as a pointer for the integrated reading developed in those later sections and in Chapter 7.

6.3 Theme T1 — The Capacity Ceiling

Every operational feature of the refined TDMF was bounded by the case team's available capacity, and capacity itself was not, and could not be, shifted by the framework during the trial.

Capacity is the most consistently described constraint across all five qualitative sources. The pre-implementation survey, focus group, observation notes, post-implementation survey, and post-implementation interviews each return to it under different vocabularies — "drowning," "below maintenance," "urgent and important." The Likert item probing perceived sufficiency of TD-allocated time was the lowest-rated baseline item (PREQ6, mean 2.4) and was the smallest pre-post mover (POSTQ6 mean 2.5, $\Delta = +0.10$). The capacity throughline across the trial period explains both the agreement to try the framework and the partial degree to which it was operationalised.

6.3.1 A Drowning Team

The most evocative articulation came from one developer in the focus group, in response to the researcher's proposal that the longer daily could host the framework's identification, evaluation, and prioritisation cycle: "if you're drowning and somebody is offering you a cup of tea, they're like, I really don't need any more water on my lungs at this point... it's just survival at this point" (focus group). Another developer in the same session expressed the same idea in resourcing terms: "we've for quite some time now we've been below what we

need to do to maintain the system... in terms of resourcing" (focus group). The same developer also reframed it as an Eisenhower-quadrant lock: "right now there are so many urgent tech debt tasks that we don't have time for that more long-term smart prioritisation thing... we are in urgent and important... there are customers with issues and there's tech debt or bugs essentially most of the time" (focus group). When the researcher proposed a 20% commitment to TD, the team agreed to try but immediately qualified the agreement: "we won't take 20 percent... basically all of the tickets on the backlog top of the backlog would need to be tech debt items because we get new SLA tickets that are dragged to the top like every few days" (focus group).

6.3.2 Capacity in the Observed Trial

Capacity was named at the very first session of the trial. The first refinement opened with the lead voicing concern that there was no time to handle TD; the team nonetheless wished to proceed and re-scheduled meetings to make space. After the refinement produced a top-of-backlog of roughly ten TD items, the next observable daily reverted to feature-dominant discussion; successive observed dailies all recorded SLA tickets as the dominant work stream, with one entry noting that "majority of tickets discussed today are SLA tickets" (observation). At the second refinement the lead summarised the pattern explicitly: "it's not valuable to refine [evaluate TD items] because there has been not enough time to pick up these explicit debt items. It's an important goal to make every fifth ticket a debt item, but the work has been very reactive, and developers have not been able to pick up items from the backlog" (observation, second refinement). One developer further reported that during the trial they had only been able to engage with SLA tickets and so had no input on TD repayment (observation, second refinement). On the same evidence, the framework as deployed did not produce a sustained 20% TD allocation, did not displace the SLA-dominant work stream that the lead summarised at the second refinement, and did not alter the team's reactive operating pattern over the observed weeks; the trial does not support the inference that these effects would have followed under different conditions, only that they did not occur in this trial.

6.3.3 The Same Constraint Across All Five Interviews

The post-implementation interviews returned to capacity as the dominant cause across the team. One developer characterised the team as "too small a team with too much to do, so it goes more to patching than to actual fixing" and located the framework's non-realisation in

the same place: "[the framework] didn't change much of anything. Partly probably from the lack of time" (post-implementation interview). The lead articulated the displacement mechanism in concrete terms: "for example [a developer] is fixing the login fault — it kind of overrode the technical debts we'd flagged through the framework. Then we said we'll do this and this and this, and that came past other technical debts that were more critical" (post-implementation interview). Another developer framed the trial as a "glued-on attempt" whose momentum the day-to-day "just ate away," noting that the team did not have ownership of the change because it had not arisen from an internally felt need (post-implementation interview). A third developer gave the temporal version: "I think one month is a very short time for us... easy to talk about, okay, now we're going to do these. And then immediately when I finished my previous bigger feature, another one pops up" (post-implementation interview). The remaining interviews — including the truncated interview, whose themes 1 and 2 were captured before recording loss — were consistent with these three readings; the truncation affected only later themes (4–5) and is flagged in those theme sections. Read as evidence about the framework rather than only about the surrounding capacity, the trial is consistent with the descriptive claim that the framework as deployed did not change much of anything in this team's day-to-day pattern, did not generate internally-felt ownership of the change, and did not survive the day-to-day pull of feature work that one developer described as having "just ate away" at the trial's momentum; whether a different deployment under different capacity conditions would have produced different effects is not adjudicable from this trial.

6.3.4 Dedicated Allocation for Bigger Items

A secondary mechanism reinforced the capacity ceiling: the team's diagnosis that the bigger TD items could not be addressed through normal backlog work and that time for them needed to be allocated separately, with the time justified to management. The lead described in the focus group the criterion that had been used over the preceding years to decide which large TD items got addressed — "is this something that would crash our platform in the next 12 months" — and the consequence: "things that take like several weeks to fix, those would need to be scheduled separately... assigning time in our roadmap that... now for this slot, let's say two months" (focus group). The Visuals project, an in-flight cross-cutting feature initiative, was characterised by the lead in the post-implementation interview as "kind of TD repayment — we have bad structures we're dismantling and putting into a different position, but it's not necessarily understood as that. It might be understood as new development" (post-implementation interview). The interview also noted that the Visuals project had at that point

taken over a year of calendar time. The lead's closing observation that "this framework is a tool for management, but business pressure from management can take focus from TD management to new feature development" (observation, second refinement) named the structural cause the framework could not itself address. The same utterance figures again in T4 (Section 6.6), where its second clause is read as the team's articulation of where the framework's underused value lies — one observation read for two purposes.

6.3.5 Survey-Side Counterpart

Both surveys probed perceived sufficiency of TD-allocated time on a five-point scale. The pre-survey mean was 2.4 (n=5) — the lowest cluster on the instrument, alongside "TD is documented consistently" and "TD is prioritized systematically." After the trial, the post-survey mean was 2.5 (n=4), an absolute shift of +0.10 — the smallest movement on any Likert item in the post-survey. Pre-survey open-ended responses had already named time and resources as the primary challenge: four of five respondents mentioned this directly (PREQ8, R1–R4). The post-survey open-ended responses returned the same answer: all three respondents to POSTQ9 named resourcing or capacity as a primary challenge, with respondent 2 noting "the work is reactive... systematic, more planned way to handle debt would be a better way... reduce context switching" (POSTQ9, R2). The directional flatness of the time-allocation item is consistent with the qualitative finding that the framework did not act on capacity. With n=5 pre and n=4 post, unmatched, the +0.10 result cannot independently confirm the qualitative conclusion, but the convergence of the lowest-rated baseline item, the smallest pre-post movement, and consistently named open-ended responses across both anonymous channels is consistent with the descriptive claim that capacity remained the binding constraint throughout the trial.

This theme primarily informs RQ3 by surfacing capacity as the dominant challenge encountered during the trial and constrains RQ1 by establishing the baseline condition under which the gap between intended and enacted application is described. The discussion in Chapter 7 returns to this constraint as the precondition for interpreting all other themes, including how partial application in Theme T2 should be read.

6.4 Theme T2 — From Calculation to Conversation

The team accepted the refined TDMF's concepts but applied its structural scaffolding while declining its principal-and-interest scoring (P/I scoring) step, using the framework as a discussion-and-signalling structure rather than as a prioritisation calculation.

The framework's P/I scoring step asks the team to estimate, for each candidate TD item, principal — the cost of remediating it now — and interest — the additional cost the team expects to incur from leaving it unremediated — and to prioritise items on the resulting ratio. Across the focus group's third phase, the first and second refinements, the post-survey open-ended responses, and all five interviews, participants described the framework as conceptually familiar and judged its mental model as broadly aligned with what the team already did. At the same time, every source recorded that the framework's principal-and-interest scoring step was skipped at the very first refinement, sustained as skipped throughout the trial, and retrospectively endorsed as having been the right call given the team's circumstances. Multiple alternative formulations to P/I scoring were articulated — in the focus group, by participants raising simpler one-number metrics, an urgent–meaningful binary, low-medium-high or must-should-nice-to-have classifications, win/lose spreadsheet columns, and cross-referencing TD items with planned features as a roadmap-relative lens; and at the first refinement, by the lead voicing effort-based rules of thumb such as "doable in under a week, let's take it". None of these was operationalised as the team's evaluation method during the trial. In actual practice, TD items were evaluated and prioritised case-by-case through team discussion of each item on its own merits, with no scoring or classification system applied. The framework that was actually applied was therefore the discussion-and-prioritise spine of the framework, stripped of its formal P/I scoring step.

6.4.1 Conditional Acceptance of the Mental Model

One developer in the focus group put the conceptual match plainly — in a session facilitated by the researcher who is also the framework's designer — "from my point of view it's pretty much what I've experienced in previous places... for me this framework pretty much is what we are doing, the question is that can we have a process to follow to actually ticket the issues and make them more visible to us also so that we don't forget that they are there" (focus group). The same alignment with prior expectations was echoed at the first refinement by a developer reflecting on the meeting just held: "this meeting was aligned with what I would

have expected. Once we have less TD items and new items, we can utilise the principal and interest" (observation, first refinement). In the post-implementation interviews, the same conditional acceptance returned — conditional in being expressed in counterfactual or future-tense form ("if only it had been used", "if we keep on working on it") and in not being accompanied by any sustained application of the framework's P/I scoring step during the trial. One developer gave the most direct formulation: "the framework felt entirely logical... if only it had been used, it would have been quite valid" (post-implementation interview). Another said the framework "fit in quite well, or at least it could fit in really well if we keep on working on it" and added that "we all agreed that it makes sense" (post-implementation interview). The remaining interview content was consistent in the same conditional register. Because the post-implementation interviews were conducted by the framework's author, these conditional acceptances are interpreted as compatible with — rather than as independent confirmation of — the qualitative reading. An alternative reading available on the same evidence is that the team accepted what was already isomorphic to its existing practice and declined the framework's novel P/I scoring step; the methodological basis for both readings is set out in Chapter 4's reflexivity subsection.

6.4.2 Decline of P/I Scoring

The decision to skip principal-and-interest scoring was made at the first refinement and was not subsequently reversed. The observation entry records the moment: "a dev asked if they should assign principal and interest to tickets. Team decided not to assign principal/interest for now — too many tickets, they need to handle most of the smaller items first before scoring" (observation, first refinement). The reasoning, articulated by another developer at the same meeting and recorded verbatim, was that "we have known debt items... we know we are so behind, we don't need to convince anyone (with principal/interest) that we need to pick these" (observation, first refinement). The reasoning recorded here places the deferred items on the deliberate side of Fowler's deliberate–inadvertent typology of TD [14]: the team is aware of the items and has chosen, given capacity constraints, not to address them rather than failing to recognise them. The same reasoning was sustained at the second refinement: "team acknowledged that ROI estimation might have been hard for them" (observation, second refinement). All five post-implementation interviews returned to the same position. The lead articulated the most extended version: "the principal-and-interest amount being assigned to tickets or debt items felt unnecessary, because in my opinion the more important thing is whether to fix or not. Sure, that helps with decision-making. But in my opinion, if a ticket

exists, the decision is already a bit baked in — 'this should be done'. Otherwise, the ticket isn't worth doing and can be removed. So, it's more 'when do we do it'" (post-implementation interview). Another developer gave the practical version: "we verbally assessed the items, but we didn't actually put any numbers on anything" (post-implementation interview). A third developer noted that in their context P/I evaluation was skipped because TD was "recognised pretty well" internally and the team had no need "to sell upward" (post-implementation interview); the remaining interviews echoed the same point briefly.

6.4.3 Resolution: Structural Layer Endorsed, P/I Scoring Declined

The endorsement and the partial application apply to different layers of the framework. The framework's structural layer — recurring discussion of TD as part of the regular team forum, the team's commitment to allocated TD time, and the recording of TD items in the centralised issue tracker — was endorsed because the team had already constructed something resembling it implicitly; the framework's contribution was to make that implicit structure explicit and shared. The framework's principal-and-interest scoring step was not partially applied so much as deliberately declined. The team articulated two conditions for meaningful P/I application — slack capacity to score items they have time to repay, and sufficient roadmap clarity to estimate interest meaningfully — and judged both as absent in their context. The team's reasoning was articulated explicitly at the first refinement: "we know we are so behind, we don't need to convince anyone with principal/interest that we need to pick these" (observation, first refinement). One developer's articulation in the focus group anticipated the same point — "it's really hard to estimate the interest if you don't know what we are going to build in the future... we don't have that like roadmap clarity to the part where we could make those decisions" (focus group) — and was endorsed at the second refinement by another developer noting that visibility into work two-to-six months ahead was insufficient to evaluate interest meaningfully (observation, second refinement). Read across the layers, the team adopted the discussion-and-prioritise spine of the framework while declining its scoring core, on grounds the team itself articulated coherently and consistently.

6.4.4 Alternatives Discussed but Not Operationalised

The team did not adopt any single replacement for P/I scoring. Several alternative formulations were articulated, all in discussion register and none implemented as a method. At the first refinement, the lead voiced effort-based rules of thumb during item-by-item

handling: "if you see something doable in under a week, let's then take it" and "if something is doable in under a day, let's take it" (observation, first refinement); for "really small TD items: don't put estimates" (observation). Another developer expressed a related preference: "getting rid of smaller debt items, even if they don't have big impact, but just to get them out of the way. I think we should do more of this, going through the backlog" (observation, first refinement). The focus group had earlier surfaced compatible alternatives. The lead proposed reducing scoring to a low-medium-high triple or to a binary: "we don't even have to assign three values, we have the is this urgent, is this meaningful... we can discuss these on their own merit" (focus group). Another developer proposed collapsing principal and interest into a single number — "how much work and how much work can be avoided, that's basically what it is... put into a one number" — and offered "frustration of you having... if this gets fixed, do I feel better" as a candidate metric (focus group). A third developer suggested an external spreadsheet with simple win and lose columns (focus group). The lead in interview consolidated the preference: "more meaningful is whether it's must do, should do, or nice to have. That's more meaningful in my view" (post-implementation interview). Each of these formulations was raised but none was selected, scored against, or used to rank items; the team's actual evaluation method, sustained throughout the trial, was case-by-case discussion of each item on its merits — confirmed across all five interviews and articulated by the lead in the focus group quote above ("we can discuss these on their own merit"). The case-by-case logic was broadly continuous with the team's pre-trial practice (described in Chapter 5); what the trial changed was the structural setting in which the discussion took place — the dedicated forum and the explicit allocation of TD time — not the evaluation logic itself.

6.4.5 The Team's Preferred Evaluation Lens

Several participants articulated cross-referencing TD with planned features as the most meaningful evaluation lens in principle — a roadmap-relative rather than ROI-relative way of thinking about prioritisation. One developer put it most fully in the focus group: "if any of those tech debt items that we identified would like make some other ticket that we are going to work on easier, then it might make sense that then the valuation kind of like changes... if it has a one, then it relates to like work we will be doing in this refactoring project" (focus group). The same proposal returned in the same developer's post-implementation interview at greater length: "it would be very useful: 'now we have this feature coming, things X, Y, Z down there are really things where the interest has now materialised. If we don't fix this now, all our work on this new feature is hellishly heavier and thus more expensive'" (post-

implementation interview). The lead's selection at the first refinement reflected an implicit and informal version of the same logic: "in tickets you think have a big impact, let's move it right away" (observation, first refinement). Like the alternatives discussed in Section 6.4.4, however, this lens did not become a method: there was no systematic cross-referencing, no scoring against the roadmap, and no procedure for surfacing roadmap-adjacent TD items. The lens was an articulated preference, not an implemented practice; the team's evaluation throughout the trial remained case-by-case discussion.

6.4.6 Forum and Willingness

The team committed to using the backlog refinement or in their words longer daily as the forum for the framework's identification, evaluation, and prioritisation activities (focus group; observation, first refinement). This commitment translated the focus group's articulation that the team did not have separate refinements — "it's the daily, just a long daily" (focus group) — into a proportionate operationalisation: the framework's forum was the meeting that already existed. The team also explicitly committed to trying the framework, in a remark whose relational register — addressed to the framework's designer in the room — is itself part of what the quote records: "for sure, for you my friend, a thousand times... I would. OK, so if I think we all can agree that we can try this out" (focus group). Within the trial, the lead noted at closing that "discussion arose... it's not directly because of the framework, but the very fact that the dev-topic was brought up — that has been useful" (post-implementation interview). One developer gave the simplest characterisation of what was realised: "at least it sparked discussion. About technical debt. Yeah, that was a good thing" (post-implementation interview).

This theme primarily informs RQ1, by surfacing the gap between the framework's intended end-to-end application and the structural-only enactment that the team produced, and informs RQ2, by capturing the team's perception of the framework as conceptually valid and conditionally useful but bureaucratically excessive in its formal P/I scoring step. Chapter 7 returns to this structural-versus-P/I-scoring distinction as the central interpretive frame for the relationship between the refined TDMF as theory and as practice in this case.

6.5 Theme T3 — Visibility, not Identification, is the Gap

Identification of TD was never the bottleneck; the analysis reads the gap the framework addresses in this team as one of visibility — to other team members, to the lead's view of the backlog, and to the statistics the team uses to characterise its own work.

The team's TD knowledge was already extensive at baseline. The pre-survey item probing identification scored 3.4 — the highest-rated baseline item — and four of the five respondents to PREQ8 named time and resources, not identification, as the primary challenge. In the focus group, the lead formulated the position bluntly: the system has "20% to 40% of the backlog [as] debt items, so we don't need to spend time to find more." What appears across observations and interviews instead is a triple visibility gap. First, team members held fragmented, individual pictures of which TD items existed and which had been triaged, with prioritisation done by leads (the tech lead together with the CTO) without team-level visibility. Second, the lead's view of the backlog was effectively confined to its top, leaving items lower in the backlog "in the void" and forcing developers to guess whether items they encountered in the middle of the backlog were sanctioned for pickup. Third, and most striking quantitatively, the same trial period yielded three different TD-repayment rates depending on tagging boundary — 2.1%, 13%, or over 50% — because the team's working definition of TD did not match the explicit tagging on tickets in the issue tracker. The post-survey item on transparency of TD decisions registered the largest directional shift of any Likert item in the instrument: 2.6 to 3.75 (+1.15), reported here as a directional indicator only. Read alongside the qualitative pattern, the shift is consistent with the descriptive claim that visibility — not identification — is what the framework supplied, and what the team needed. The TDM literature has typically treated identification as a non-trivial problem [4]; the case-specific finding here — that identification was saturated and visibility was the gap — is therefore a contrast with the literature's modal framing, not a general claim about TD identification across teams.

6.5.1 Identification as a Saturated Baseline

The lead articulated the position in the focus group: "we have debt everywhere in the system, we have already 20% of the backlog to 40% of the backlog is debt items, so we don't need to spend time to find more debt items that they are right at our eyes" (focus group). One developer said the same in summary form at the trial's commitment moment: "for now, it's we

won't be needing to do much identification. We would just need to evaluate existing... and also kind of decide in the longer daily" (focus group). Another developer reinforced the point in interview: "pretty much all of the tickets we put on the backlog have decent value... they are already good items to work on... it doesn't really matter which ones we pick" (post-implementation interview). A third developer, less directly involved in TD selection during the trial, gave the inverse evidence: "over the years there's certainly accumulated quite a lot, because I have a feeling that no attention has been paid to it... there's lots of undocumented" (post-implementation interview). This is consistent with — not contradictory to — the team's claim of saturated identification: identification had occurred, but the documentation step that would render identifications visible to others had not always followed. The pre-to-post shift on the identification item, 3.4 → 3.25, was the only Likert item to register a decline. With $n_{post} = 4$, a 0.15 movement represents at most one respondent's single-step change and cannot be reliably interpreted; the data are equally consistent with sample noise, with item ambiguity, and with a substantive reading that the framework's visibility work made participants more aware of un-identified TD they had not previously realised was there, slightly lowering self-rated identification confidence. The downward movement is returned to in Section 6.8 and Chapter 7.

6.5.2 Documentation as the Visibility-Generating Mechanism

Three developers articulated the documentation-as-visibility argument across the post-implementation interviews. One put it most fully: "the documentation... it's not the framework's most useful part except in the sense that then it brings visibility — it doesn't stay just devs knowing 'here's a bit of this and that'. The fact that it gets documented makes it a visible part for others too. And then maybe it would force more toward fixing" (post-implementation interview). Another characterised the same dynamic in its negative form: "technical debt is somehow invisible — very invisible. The only thing that's visible is bugs. We're entirely at the mercy of opinions when devs come saying things are problematic" (post-implementation interview). The focus group had already established the existing documentation landscape as scattered: a shared technical-state document, individual issue-tracker tickets, audit findings, and automated dependency-vulnerability scanner outputs, with no single canonical source (focus group). One participant observed that "there are at least two documents in our drive regarding tech debt and legacy" (focus group), and the same fragmentation surfaced in the team's view of itself: "I don't think we have a common understanding of all the tech debt items and also the prioritisation" (focus group).

6.5.3 Tickets as the Gating Mechanism

The team's working norm was that visible TD work is ticketed TD work, with all the consequences that follow. The researcher's paraphrase, ratified by one developer in the focus group, articulated the norm: "nothing gets done if it doesn't get to the issue tracker" (focus group). Another developer gave the asymmetric pattern: "if it's from a customer, then I create a ticket... currently we are overwhelmed by the amount of stuff, so we cannot add anything more" (focus group); and "now I don't know, no one else is going to have time for it. So, if I create the database maintenance ticket, I could also just do it immediately or just forget about it" (focus group). The first refinement gave a direct enactment of the norm: when one developer raised a TD item — an unused admin feature that could be removed — the lead asked, "is there any ticket?" (observation, first refinement). Friction in the issue tracker (slow loading, "rows you don't see literally don't exist" in the new UI) reinforced the asymmetry — items that customers would not chase were not worth the cost of ticket creation (focus group). The team rejected a separate dedicated TD list outside the issue tracker on the grounds of historical experience with shadow lists: "the [issue tracker] backlog has been huge and there's been a lot of stuff we basically know that we won't ever have time to do... if we haven't done it in a year, we won't probably do it" (focus group). The lead formalised the rejection: "if you have separate list, it's the same that if you have multiple backlog, which one determines the priority... I would keep it in one place" (focus group). The proposed sustainable practice was brain-dump-and-prune: write items down informally, review every two weeks, drop what will not be done within one to two months (focus group).

6.5.4 Top-of-backlog as Priority Signal — and its Limits

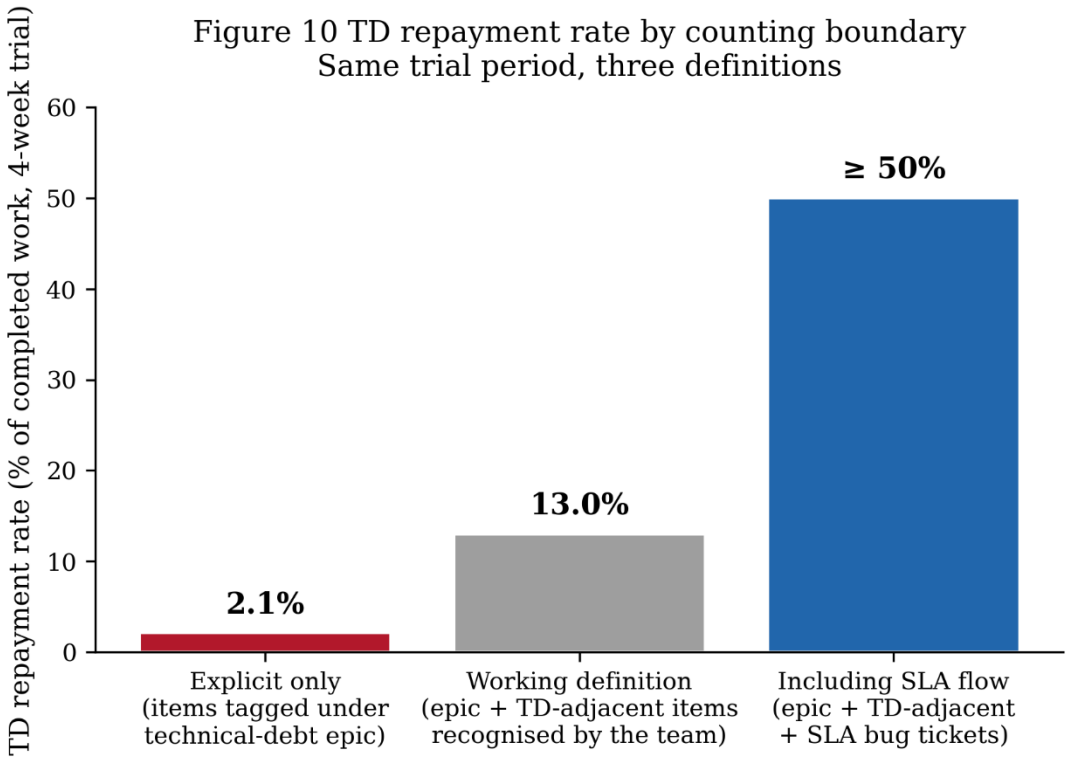
The first refinement operationalised top-of-backlog ordering — standard Scrum backlog management rather than a framework-specified mechanism — as the team's priority-signal mechanism for TD items, used in lieu of the framework's ROI prioritisation step: the lead physically moved 10 TD items to the top, ratified the rule "every fifth item is a TD item," and stated "in tickets you think have a big impact, let's move it right away" (observation, first refinement). The mechanism worked at the personal level for one developer: "it gets closer to the top of the backlog. So, when I pick my next task, I was way more likely to pick a tech debt item than before" (post-implementation interview). The mechanism remained operational structurally throughout the trial: at one observed daily, "top of the backlog has 3-5 prioritised tech debt items" (observation). The mechanism also surfaced a backlog-usage mismatch that

had previously remained implicit. The lead described the mismatch in the focus group: "in my head, it was like, hey, if you have time, you can take some DB maintenance ticket, but that was not the case with [the same developer] because he respects the priority of the backlog... I haven't communicated that too well" (focus group). Another developer said in the same exchange: "I never look at this" — the regular backlog. In the post-implementation interview, the lead confirmed the mismatch had remained partially unresolved: "devs may have thought 'because I haven't prioritised them to top of backlog, they aren't to be done'. So, communication has remained one-sided. Both clearly assume the opposite, and then nothing happens" (post-implementation interview). A residual visibility gap surfaced in an observation: a large-scope ticket entered development without the tech lead seeing the ticket; it had been approved by other management people and some developers (observation). Even with the top-of-backlog signal in place, governance could be circumvented when approval flowed through alternative channels.

6.5.5 The 2.1% / 13% / 50% Repayment-Rate Divergence

The most striking quantitative pattern within this theme — a 24-fold spread between the team's narrowest and broadest definitions of TD-repayment work — was named at the second refinement. The tech lead aggregated TD-repayment metrics through filtered queries on the team's issue tracker and reported three figures for the same trial period. Two explicit debt items had been repaid in the last four weeks under the technical-debt epic, yielding 2.1%. Manually expanding the count to include items the team thought of as TD-equivalent but had not marked under the technical-debt epic yielded 13%. Including SLA tickets — most of which the team treated as TD-equivalent because the bugs they fixed were caused by suboptimal implementations — pushed the figure over 50% (observation, second refinement). The lead immediately acknowledged the cause: "[I] might have failed to explicitly mark tickets under the tech debt epic" (observation, second refinement). The same 50% figure was named also in the anonymous post-survey by one of the four respondents (R2): "when looking from [the issue tracker], we use (depending on point of view) roughly half of the time / resources to technical debt, which is pretty much" (POSTQ9, R2). In interview, the lead gave the same number again: "from [issue-tracker] stats — if you'd convert ticket counts to time use, roughly half of time goes to some kind of debt repayment or bug fixing" (post-implementation interview). The three figures are not three counts of a single phenomenon but

three differently-defined operationalisations of TD-related work: 2.1% measures intentional, explicitly-tagged TD repayment; 13% measures repayment under the team's working definition, including items the team treats as TD even when not formally tagged; and over 50% measures debt-induced reactive work, in which TD is addressed through SLA-driven symptom-handling rather than through systematic remediation. The 24-fold spread between the explicit-tagged count and the working-definition count is therefore primarily a measurement-validity issue, not a tagging-hygiene one: the team's working definition of TD has no fixed boundary against debt-induced reactive work (an earlier observation had foreshadowed this — "team thinks of SLA bugs as debt items but they are not explicitly marked under the technical debt epic"), and the explicit tagging in the issue tracker captures only a subset of what the team itself counts as TD work. R2's hedging in the anonymous response — "depending on point of view" — is itself recognition of this definitional ambiguity, not corroboration of a single repayment rate; the convergence across sources is on the divergence itself, not on any one figure. Figure 10 visualises the resulting divergence: the same trial period yielded TD-repayment rates an order of magnitude apart depending only on which definitional boundary was applied.



Aggregated by the lead at the closing reflection (20.4); independently corroborated by POSTQ9 R2.

Figure 10. TD repayment rate over the four-week trial, calculated three ways from the same dataset. Aggregated by the lead at the second refinement; the upper bound was independently corroborated by an anonymous post-survey response (POSTQ9, R2).

One developer's post-implementation reflection drew the operational implication: "we'd have to be better at tracking how much tech debt we actually do, because if half of my week's work is reacting to problems in Slack, many of those don't have tickets at all" (post-implementation interview).

6.5.6 Survey-Side Counterpart

The post-survey item probing whether decisions about TD are transparent moved from a pre-mean of 2.6 to a post-mean of 3.75, an absolute shift of +1.15. This was the largest pre-to-post directional movement on any Likert item in the post-survey. The post-survey medians moved from 3 to 4. The pre-survey distribution was 0-2-3-0-0; the post-survey distribution was 0-1-0-2-1, with one respondent at the top of the scale where there had been none at baseline. The dispersion widened alongside the upward shift (std 0.55 → 1.26): responses spread further apart, so the directional movement registered alongside greater team-level divergence rather than uniform improvement. The open-ended responses to POSTQ8 (improvements observed) all three named visibility or discussion as the primary improvement: "it opened discussions of

TD. The team is more conscious of TD" (POSTQ8, R1); "it brought the technical debt more visible / to discussions in team level, at least temporarily" (POSTQ8, R2); "it made, at least me, more aware of our handling of the tech debt" (POSTQ8, R3). R2's hedge — "at least temporarily" — is itself a sustainability caveat: the visibility effect captured here is what the team perceived during the four-week observation window, not a durable change the trial can confirm. Given n=5 pre and n=4 post, unmatched, the +1.15 cannot independently confirm the qualitative conclusion; reported here as descriptive complementary evidence, the shift is consistent with the qualitative reading that visibility was the framework's principal delivered value during the trial.

This theme primarily informs RQ1 by characterising the framework's actual delivered value in this trial — visibility rather than identification — and informs RQ3 by surfacing the principal benefit the team experienced during the trial, alongside the marking-gap finding that simultaneously functions as a benefit and a measurement challenge. Chapter 7 returns to the visibility framing and the 2.1%/13%/50% divergence as the basis for adaptation recommendations directed at TD-tagging conventions and at the framework's measurement aspirations.

6.6 Theme T4 — The Framework's Untapped Potential is Upward-Facing

A consistent pattern across interviews, observations, and the anonymous post-survey was that participants identified the framework's most valuable potential not in coordinating work within the team, but in communicating upward — to the tech lead, the CTO, and the wider organization. This reframing was not present at the start of the trial; it was articulated as the team accumulated experience attempting to apply the framework against a backdrop of reactive work and unchanging capacity. By the second refinement, the team's articulated conclusion was that the framework had relatively little untapped value to deliver as a team-internal coordination tool but considerable untapped value as an upward-translating instrument — a vehicle for converting developers' tacit awareness of accumulated debt into a form that organizational decision-makers could act on. This pattern, present across observations, four of five interviews, and an anonymous post-survey response, cuts across the framework-as-designed assumption that TD management is principally an intra-team activity and constitutes the team's most consistent articulation of where they perceived the framework's underused value to lie. No upward communication of TD information from the team to management actually took place during the trial; the theme is therefore one of

articulated potential rather than demonstrated function, and the supporting participant statements are predominantly conditional and prospective in form. The supporting codes (visible in Figure 11) describe a stable role-based and pressure-based mechanism that operated alongside the framework throughout the trial, mostly outside the framework's reach. The relationship of this articulation to TDM literature on stakeholder perspectives on technical debt — particularly practitioner-survey work on TD perceptions across IT roles [24] — is taken up in Chapter 7, alongside the question of how the framework should be adapted in light of it.

6.6.1 Cross-Source Agreement

A summative articulation came at the end of the trial. In the second refinement, after the team had reviewed the previous four weeks' work and the lead had presented the metric-divergence finding (treated in Section 6.5.5), the team then discussed whether refinement of TD items should continue. Acknowledging again that their work had been very reactive, the lead summarised: "A framework can't address this, but management needs to step in and provide bandwidth with resources or removing pressure from new feature development. This framework is a tool for management, but business pressure from management can take the focus from TD management to new feature development." (observation, second refinement). A second developer immediately added that there had been no time for TD repayment and no direction from management to allocate time for it (observation, second refinement). The articulation was recorded in the field notes as the team's own integrative statement; because the field notes were authored by the researcher, who also facilitated the refinement, the researcher's position and its bearing on observation-based citation are discussed in Chapter 4's reflexivity subsection.

The same conclusion appeared independently in the interviews. One developer in a post-implementation interview constructed an extended argument that the framework, as deployed in this team, risked becoming "a fancy scrapbook of what debt we have" unless the change came from above: "the change should perhaps come from above — and we're given a task and a mandate to execute it as clearly as possible. So, we'd have a communication channel forward, or upward". The same developer elaborated that without an organizational mandate the framework "is experienced as extra load — some 'spring carnival exercises' that have to be done if no benefit is felt obtainable" (post-implementation interview). This was not framed as a critique of the framework's logic; the framework was explicitly described as "very light and

quite flexible" with "few things set in stone" (post-implementation interview). The critique was specifically about its silence on the upward channel: "the framework could take a stronger stance specifically on resourcing. Now it functions as a basis for resourcing more than as a definer" (post-implementation interview).

Another developer articulated the same point in a different idiom: the framework's documentation, if it were used, would have its primary value in being made visible to people outside the team — "in our company, made visible to the CTO and possibly even to tech lead, that 'here is a stack of stuff we are doing nothing about'. Which would probably give them a kick in the rear" (post-implementation interview). The same developer noted that "all that infrastructure already exists" — the team already had monthly meetings where such aggregations could be presented — and the gap was specifically the assembly and upward presentation step (post-implementation interview).

A third developer framed the same gap differently again: "Frameworks tell you 'this is how it should be done, this is the best way' and then after everyone has learned about it it's like 'okay but how do we actually do this' and then you hire a consultant. The framework doesn't tell you how you can apply it." The same developer described this as "vital information missing from the framework" and tied the missing element to the same upward channel the others had named — communicating the case for TD management to roles outside the developer team, including roles whose decisions about feature pressure and resourcing constrained what the team could do.

6.6.2 Earlier Traces and Anonymous Corroboration

The focus group provided early traces of the same reframing, though it was less developed there. One developer in the focus group had already characterised principal/interest scoring as "mostly used to convey the urgency to external stakeholders" rather than as an internal team tool, and the lead in the focus group had supported a related distinction: when a debt item was small, "people outside this team aren't interested," but for items measured in months "people outside of this team are also interested." What changed between focus group and interviews was the framing's centrality. In the focus group it appeared as an aside about one component of the framework; by the post-trial interviews it had become the team's articulated statement about the framework as a whole.

The anonymous post-survey corroborated this pattern through an independent channel. POSTQ9, R3, responding to the open-ended question on framework limitations, wrote that the framework "is not sufficiently opinionated on the amount of tech debt that is maintainable. Like the 'national debt' counter is just a number, but the risks going with that amount of debt are very real. The tech debt framework should have some guiding principle telling us, and the management, what is acceptable and what amount will end up crushing the system unless handled" (POSTQ9, R3; emphasis in conceptual reading). Because POSTQ9 was anonymous and written, this contribution is less plausibly attributable to interview rapport or focus-group dynamics than the interview material; it remains, however, a single anonymous response (one of n=4) and is read as one observation alongside the interview and observation evidence rather than as an independent confirming channel. POSTQ8 responses (R1, R2, R3) did not name upward communication as a realised improvement, but they did consistently name *visibility* and *awareness* as the framework's primary contribution (see Section 6.5 on T3) — a finding compatible with the upward-facing reframing in the sense that the visibility the framework produced was not yet wired into a communication channel that reached management. Read on a less-charitable interpretation, the same participant statements function as critiques of the framework as deployed: what is articulated here as upward-facing potential is equivalently an observation that the framework had not delivered upward value during the trial. Both readings are available on the evidence.

6.6.3 External Pressure as Existing Driver

The upward-facing reframing has a counterpart on the work-driving side. Across the trial, the work that actually progressed on TD items was largely driven not by the framework but by external pressure streams: yearly external audits with a service-level commitment to fix all medium-or-higher findings, security and accessibility pentests, automated vulnerability scanners, and customer-reported issues. The lead in the focus group described this as the team's standing pattern. The framework did not change this stream; it operated alongside it. In observations, the pattern was visible across the trial: one observation noted "a lot of security audit findings" alongside the SLA flow; another recorded one developer fixing an audit finding; the second refinement mentioned new items documented from the accessibility audit. The lead in the post-implementation interview made the relationship explicit: "Pentest came, those are now prioritised, get done immediately, recheck a week later... Accessibility tests came in critical or high, I pulled those up. So, it didn't come from the framework, it came from elsewhere." Another developer in a post-implementation interview recounted the trial's

most concrete instance: the 20% TD allocation "didn't really work as expected" because pentest tickets consumed roughly a week to a fortnight of TD-coded time. POSTQ9, R2 echoed the same picture: "We have multiple ongoing new feature development projects at the same time which causes us to have limited possibilities to handle the technical debt or the debt payment feels slow."

External pressure was thus the dominant driver of actually-addressed TD throughout the trial, just as it had been before the trial. The framework did not displace this stream. It also did not contest it: the team treated external-pressure-driven work as legitimately TD-relevant, and the lead's metric divergence at the second refinement (2.1% / 13% / over 50%, see Section 6.5.5) explicitly broadened to include SLA tickets when computing the upper bound. The implication for the upward-facing reframing is that the team is already responsive to external pressure; what the framework lacked, in their account, was a mechanism to translate the team's internal awareness of debt into a comparable form of external pressure — that is, a way to make TD legible upward in roughly the way a pentest finding or an audit recommendation is already legible upward. This is the gap that POSTQ9, R3's "acceptable-debt threshold" proposal addresses, and it is the gap that one developer's "kick in the rear" framing presupposes. The lead's "debt sanctuary for developers" framing at the first refinement (observation, first refinement) provided one existing intra-team mechanism for absorbing pressure, but the team did not describe it as sufficient on its own — it absorbed pressure rather than translating debt upward into the form the team understood as actionable.

This theme primarily informs RQ4 — how the refined TDMF should be adapted in light of trial findings — and bears on RQ1, in that the framework as designed treats TD management as principally an intra-team activity and the team's articulated reframing identifies a register the framework does not yet address. It also bears on RQ2 as the most consistent participant perception about where the framework's underused value lies.

6.7 Theme T5 — The Substrate: Legacy Code and Team Structure

The four themes presented above all describe how the framework was, was not, or might be applied. The fifth theme is of a different kind. It describes the substrate against which the framework operated — the accumulated technical and organizational state that shaped what the framework could do in this team and on this codebase. The substrate is not a finding about the framework's application; it is read across sources as recurring background context every time framework application was discussed. Without it, the other four themes are easy to

misread as universal claims. With it, they become findings rooted in this team's specific conditions: a long-lived legacy codebase with structural data-model debt, code that lacks a discoverable logical structure, and a team that, in practical terms, is parallel one-person sub-teams rather than a single team with shared work. This theme is therefore presented descriptively as a precondition rather than as a finding about framework application in the narrow sense; at the same time, the substrate is itself evidence about the framework, in that it names the conditions under which the framework was tested and therefore the conditions under which its potential effects were neither demonstrated nor falsified — they were simply not given the conditions to manifest. Its interpretation as a moderating condition is taken up in Chapter 7; its bearing on what the trial can claim about the framework is addressed alongside other limitations in Chapter 8. The supporting codes (visible in Figure 11) together describe a substrate that was not contested by any participant or any source in this case; whether that consensus is itself a feature of the substrate or an artefact of the focus-group setting is one of the reflexive limitations addressed in Chapter 8.

6.7.1 Codebase Substrate

The team described the codebase as carrying structural debt at the data-model level, not only at the implementation level. One developer characterised it concisely in the focus group: "It doesn't have a logical structure, it's not something that you can learn, it's more like you just have to remember" (focus group). The same passage continued: "It's really hard to step into the code base and start being productive... it's hard to wrap your head around." The same developer elaborated on the naming layer in particular: "We have base and we have layout. We have layout in the code, which is the base, and I don't even know what the layout is" (focus group). The compound effect — incremental extension without redesign, naming that misled rather than guided, hidden side effects in functions whose names suggested narrower behaviour — was described by two developers together: the pre-check function "performs rendering of the top navigation bar... it shouldn't modify rows in the database... the naming is confusing" (focus group).

The structural data-model layer was the substrate's load-bearing concern — what the architectural-debt literature would label as TD with architectural roots [21]. One developer in the focus group gave the example of the participants table, which "causes problems like everywhere," and of "translation tables that aren't translation tables — they are like the actual entity, which is stored in three rows." The lead in the focus group extended the picture: "One

key entity that is missing is the enrollment entity, and that's why our API gives three different response formats... it's really hard for the API user to understand what to expect as a response." The integration consequence was described in the same passage as a concrete cost: "We have multiple cronjobs running — one for [each CRM integration]... It might calculate the hashes for, let's say, 200,000 participants and then make three calls to [the CRM]" (the lead, focus group). The missing central save pipeline, in other words, surfaced as recurring runtime overhead at the integration boundary, not just as an abstract structural complaint.

Structural ossification — current structures having become hard to remove — was articulated by the lead in the same focus-group passage: "We have accepted that it is like this and because if we try to address that issue we need to basically explode the whole system" (focus group). In Fowler's typology [14], the substrate's structural debt sits largely on the inadvertent side: the structural state was inherited rather than chosen by the current team, and the team's relation to it is one of accommodation rather than active accumulation. One developer in the post-implementation interview returned to the structural layer and connected it to ongoing cost: "The lack of abstractions — we have no participant-type concept in the data; there are no table structures, just stuff scattered in question-and-answer fields... So those debts' repayment — we keep paying it all the time. Every time we have to dig through how these things actually work" (post-implementation interview).

The substrate's effect on daily development was visible in observations as well as in interviews. In one observation, a developer reported difficulty understanding how an assigned bug ticket should work, and how the corresponding code worked; another developer noted that they too were still trying to understand it (observation). Immediately afterwards, the lead and the same developer stayed after the meeting to use AI-assisted code analysis to investigate the implementation; the analysis had previously been unsuccessful, and was now partially successful, with the chat history shared between them (observation). Even AI-assisted analysis, in other words, did not reliably surface the code's behaviour. Another developer, in a post-implementation interview, gave a different example of the substrate's daily cost: the admin panel "sometimes... takes a minute to load this page" because it dumps an entire table without filtering into an HTML table and then runs a JavaScript datatables library on top. The same developer characterised the issue as "severe tech debt but no one reports it," because users had become habituated to the slowness. The substrate, in this

account, had degraded the team's signal channels as well as its codebase: behaviour bad enough to qualify as severe debt produced no tickets because users no longer expected the system to be otherwise.

6.7.2 Team-Structural Substrate

Alongside the codebase substrate, the team described an organizational substrate that constrained how the framework's team-level mechanics could function. The framework, as designed, presupposes a team that refines work together — that brings TD items into a shared forum, evaluates them collectively, and arrives at shared priorities. The team did not describe themselves as that kind of team. One developer in the focus group put the structure plainly: "We have one person teams inside the team... everybody's working on everything," and "we don't have refinements as a team... it's a long daily, just a long daily." The lead, in a post-implementation interview, echoed and extended this account: "We have three projects running, each with one dev. And then [a colleague] does SLA cases. So, everyone is in their own silo... refinements — we don't really have refinements as a team — more handover sessions." The lead characterised the operating mode as "managed chaos" rather than Scrum or Kanban. Another developer, in a post-implementation interview, described the same fragmentation from the developer side, framed in terms of capacity: "[a colleague] has been hammering at the parent-company integration. He's been more or less stuck on that and hasn't been able to do other things. Too few hands."

The fragmentation had material consequences during the trial. At the first refinement, one developer remarked at the close of the refinement: "Fine with me. Going through tickets, not that much related to me, I felt useless" (observation, first refinement). At the second refinement, another developer noted that they had only been focusing on SLA tickets and "didn't have any input when it comes to TD repayment" (observation, second refinement). Both observations reflected the same underlying structure: when each developer's work sat in a different system area and on a different project, a shared refinement of TD items in the longer daily was, by design, only partially shared — most items belonged practically to one person rather than to the team. Another observation that a large-scope ticket entered development without the lead seeing it (observation) added a related governance facet to the same structure: ticket flow could route around the lead because the lead was not, in practice, a single coordinating point through which all work passed. Read as evidence about the framework rather than only about the substrate, these observations are consistent with the

descriptive claim that the framework as deployed did not establish a shared refinement forum in which TD items belonged to the team rather than to individuals, and did not produce a governance pathway through which a large-scope ticket would reliably surface to the lead before entering development; whether a different team-structural substrate would have permitted these effects is not adjudicable from this trial.

The framework's adaptation in this team — the longer daily as a TD forum, the team's enactment of the framework's prioritisation intent via standard Scrum backlog ordering (top-of-backlog), and the discussion-and-prioritise spine described in Section 6.4 (Theme T2) — can be read against this team-structural substrate as the operationalisation feasible under this team's specific substrate conditions; whether a different operationalisation would have been possible under different conditions is a counterfactual the trial cannot adjudicate. The framework's fuller operationalisation, with collective principal/interest scoring and shared evaluation across the team, would have required a kind of team-level coordination this team did not, on its own account, have. This is a descriptive observation; whether the framework should accommodate teams of this kind, and how, is a Chapter 7 question.

This theme is contextual to all four research questions. It is most directly relevant to RQ1, where the practice-versus-theory gap is shaped by what the substrate forces, and to RQ4, where any adaptation of the framework will need to acknowledge that the substrate against which it is applied may vary substantially across teams and codebases. The same conditions also bound what the trial can claim about the framework: where the substrate dominated, the framework's potential effects were neither demonstrated nor falsified in this case, and the substrate is therefore evidence both about the conditions under which the framework operated and about the limits of what this trial can adjudicate regarding the framework itself.

6.8 Synthesis

The five themes presented in Sections 6.3–6.7 describe distinct facets of a single trial. T1 (capacity), T2 (calculation→conversation), and T5 (substrate) name the conditions under which the framework was deployed: acute time-pressure, conceptual acceptance with declined P/I scoring, and an operational ceiling set by a substrate of legacy code and team-structural fragmentation. Within those conditions, the cross-source argument most strongly carried across sources runs as a single chain: the team's clearest articulated improvement was a transparency gain in how TD decisions were made (Section 6.5.6; PREQ4 → POSTQ4 moved 2.6 → 3.75, a directional movement of +1.15 against a widening of dispersion from

std 0.55 to 1.26, the largest pre/post shift on any Likert item), but that transparency made visible a tagging gap so wide that the team's TD-repayment rate could be reported as 2.1%, 13%, or over 50% depending only on the counting boundary used (Section 6.5.5), and the team's reflective response to that visibility was not a request for more internal coordination but a call to translate the resulting picture upward to management (Section 6.6.1). The chain the analysis constructs is that the visibility gain is consistent with surfacing a measurement artefact, the artefact is consistent with making the framework's internal limit visible, and the team's articulated remedy is read as a call for a channel that did not yet exist. The chain is presented as a suggestive analytical reading rather than a sequence of established causal claims; three "consistent with" links chained together do not produce a stronger conclusion than any single link affords.

Each link in the chain is multiply sourced: the transparency gain by the post-survey Likert change, observations of the top-of-backlog signal being operationalised, and the three POSTQ8 free-text responses; the marking-gap divergence by the second-refinement observation in which the lead aggregated three different repayment rates from the same data, and by POSTQ9, R2's independent post-survey corroboration; the upward-facing recommendation by the second refinement, four of five post-trial interviews, and POSTQ9, R3's anonymous proposal of an "acceptable-debt threshold" mechanism. Because POSTQ9 was anonymous and written, this response is less plausibly attributable to interview rapport or focus-group dynamics than the interview material; it remains, however, a single anonymous response (one of n=4) and is read as one observation alongside the interview and observation evidence rather than as an independent confirming channel.

Figure 11 shows the relative weight of each theme in the qualitative coding, broken down by source. Theme T2 (calculation→conversation) and Theme T1 (capacity ceiling) carry the densest interview presence, reflecting the centrality of framework-application talk and capacity talk to the post-implementation reflective sessions; Theme T3 (visibility-not-identification) is the most evenly distributed across the three qualitative sources, and Theme T4 (upward-facing) is the smallest in absolute terms but has the most balanced source

contribution.

Figure 11 Code-tag distribution per theme by qualitative source

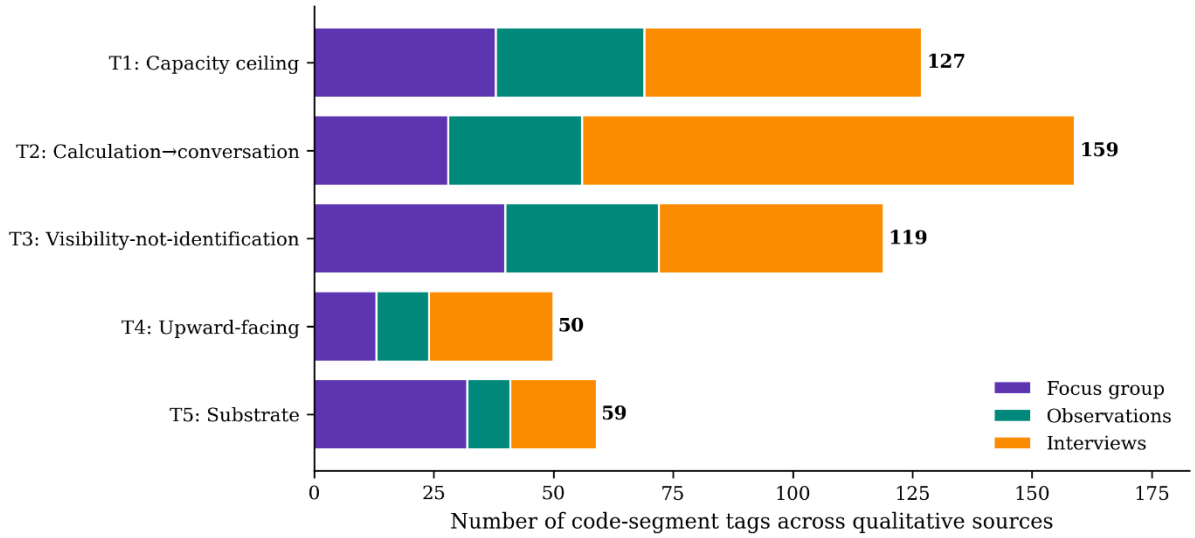


Figure 10. Distribution of code-segment tags per theme across the three qualitative sources (focus group, observations, interviews). Numbers at the right of each bar give the total tag count.

The themes also interlock in a smaller, more local way. T2's adoption of the longer daily as the framework's forum — which produced T3's visibility gain — is shaped by T5's team-structural substrate (the team had no genuine team-level refinement to begin with, so the longer daily was the maximal forum available). T3's marking-gap divergence is amplified by T5's data-model debt (the TD/bug boundary is partly a consequence of the codebase forcing fixes that look like bug work). T4's external-pressure-as-existing-driver finding (Section 6.6.3) is the obverse of T1's capacity ceiling: the work that did get done on TD during the trial was largely driven by the same external pressure streams that had been driving it before the trial, and the framework operated alongside this stream rather than redirecting it. None of these connections is necessary in the sense of being entailed by the framework; all of them are contingent on the trial's specific conditions, which is why T5 is presented as a precondition rather than as a finding about framework application.

Taken together, the case-study evidence presented in this chapter supports the descriptive picture that the refined TDMF was applied selectively in this team — adopted in its discussion-and-allocation form, declined in its P/I scoring form, and articulated by the team as having underused upward-facing potential — under conditions of acute capacity constraint and a substrate of legacy code and team-structural fragmentation. What would have falsified this picture in this trial: a sustained 20% TD allocation that displaced feature work; a documented operationalisation of P/I scoring or a substitute that the team adopted as a

method; an upward communication event in which the framework's outputs reached management and produced a resource decision; or a period of TD repayment that the framework, rather than external pressure, drove. None of these occurred during the four-week observation window. Chapter 7 takes these findings into the four research questions in turn — RQ1 primarily through T2's structural-versus-P/I-scoring distinction with T1 and T5 as moderating conditions; RQ2 through T2 and T3 descriptively and T4 prescriptively; RQ3 through T3 (benefits) and T1 (challenges) with Section 6.5.5's marking-gap finding occupying both sides; and RQ4 through T4 primarily and T5 secondarily. Limitations bearing on the analysis — the n=5 / n=4 unmatched survey samples, the single-team case design, the truncation of one interview's improvement-and-reflection content, and the insider-researcher position — are addressed in Chapter 8 alongside the directions for future research.

7 Discussion

This chapter answers the four research questions of Section 1.3 in turn, drawing on the five themes presented in Chapter 6 and the cross-source synthesis of Section 6.8. Section 7.1 addresses RQ1 — the difference between the framework's intended theoretical application and its actual implementation — primarily through Theme T2's structural-versus-P/I-scoring distinction with Themes T1 and T5 as moderating conditions. Section 7.2 addresses RQ2 — the team's perception and experience of the framework — through Themes T2 and T3 descriptively and Theme T4 prescriptively. Section 7.3 addresses RQ3 — benefits and challenges — through Theme T3 (benefits) and Theme T1 (challenges), with the marking-gap finding of Section 6.5.5 occupying both sides. Section 7.4 addresses RQ4 — how the framework should be adapted to better support Agile projects — through Theme T4 primarily and Theme T5 secondarily. Section 7.5 reflects on what the case-study findings imply for long-lived Agile projects more generally. Limitations bearing on these answers and directions for future research are taken up in Chapter 8, alongside the summary of findings and contributions.

7.1 RQ1 — Intended Theoretical Application versus Practice

RQ1 asked how the proposed application of the refined TDMF works in practice in an Agile software project, and what differences emerge between the intended theoretical application and its actual implementation. The case-study evidence supports a single descriptive answer: the framework was applied selectively. Read against the case data, the refined TDMF can be analytically partitioned into a structural layer (recurring TD discussion forum, allocated time, and recording of TD items in the centralised backlog) and a calculative layer (the ROI prioritisation step set out in Section 3.5.4, using principal-and-interest estimates without the interest-probability weighting of the original TDMF). A fourth designed element — visualisation and monitoring (Section 3.5.6) — was not adopted as a team practice during the trial and is therefore not represented in either layer. The team's adoption of top-of-backlog signalling as a priority mechanism is standard Scrum backlog management rather than the framework's specified ROI prioritisation mechanism; in this case it functions as the team's enactment of the framework's prioritisation intent in lieu of the calculative step. This decomposition is imposed by the present analysis — Chapter 3 organizes the framework around design principles rather than around two layers — but the case team's selective adoption maps cleanly onto it. The structural layer was set up at kick-off as designed;

sustained enactment was partial across each element — the 20% TD allocation did not sustain under the capacity ceiling documented in Theme T1, marking discipline was inconsistent (Section 6.5.5), and TD-focused discussion in observed practice was concentrated in the two refinement sessions of the four-week trial. The calculative layer, which Chapter 6 refers to as principal-and-interest scoring (P/I scoring), was declined from the first refinement and was not subsequently reinstated.

The structural layer comprises the recurring TD discussion forum (operationalised as the longer Monday daily), the team-level commitment to a roughly 20% TD allocation, and the recording of TD items in the centralised issue tracker (with the marking-gap caveat documented in Section 6.5.5). Each was recognisable in observed practice across the four-week trial. The closing reflection and four of five post-implementation interviews described these elements positively; one team member did not respond to the post-survey, and POSTQ9 contained mixed material including critical responses on resourcing and on the framework's limits. A developer's articulation in the focus group — that the framework was "pretty much what we are doing" — is read as endorsement of this structural layer rather than as confirmation that the framework as a whole had been applied.

The calculative step was declined at the kick-off refinement on the team's prior assessment, not as a within-trial retreat from a capacity ceiling that subsequently emerged. The framework's ROI prioritisation step presupposes two enabling conditions: capacity to score and act on items the team has time to repay, and roadmap clarity sufficient to estimate interest meaningfully relative to upcoming feature work. Theme T1 documents that the first condition was assessed by the team as absent at kick-off — and the team's reports across observations, interviews, and POSTQ9 free-text describe it as remaining absent across the four-week trial under sustained SLA pressure and feature crowding. Theme T5 documents that the second condition was further constrained by the substrate against which the framework operated: data-model debt that the team identified as systemic and architecturally rooted in the sense Kazman et al. characterise [21], and team-structural fragmentation in which TD items disproportionately concerned only one or two members of a five-person team. Under those conditions, the team's reasoning — "we know we are so behind, we don't need to convince anyone with principal/interest that we need to pick these" — is read not as misunderstanding the framework but as a reasoned prior assessment that the calculative step did not earn its overhead in this team's conditions; the in-trial capacity ceiling that emerged across the four-week trial was consistent with that prior assessment but was not its basis.

The gap between intended and enacted application is therefore principally located at the calculative layer and is conditional on capacity and substrate, not on a global rejection of the framework. Reading theoretical proposal against enacted practice in this case yields a more specific picture than "framework adopted" or "framework rejected": the framework's structural scaffolding was tractable and was kept; its calculative scaffolding required preconditions the team did not have, and the team's articulated alternatives — case-by-case roadmap-relative discussion — were not operationalised as a formal evaluation method. The pattern of declining or restructuring the calculative step in case-study applications of TDM in Agile contexts has parallels in prior reports [17], [25], and the case-team's experience of time pressure and feature deadlines as primary drivers of TD persistence is consistent with industry survey evidence that time pressure and deadlines are the most frequently reported causes of TD across organizations [24].

7.2 RQ2 — Team Perception of the Framework

RQ2 asked how development team members perceive and experience the framework as part of their Agile processes. The case-study evidence supports a three-part description of perception: conceptually accepted, operationally selective, and prescriptively reframed.

The team accepted the framework's conceptual content. Theme T2 records broadly positive reception of the mental model the framework encodes among respondents who answered. The structural elements of the framework — discussion forum, allocated time, and recording of TD items in the centralised backlog — were perceived as already implicit in the team's existing practice; the framework's contribution was to make that implicit structure explicit and shared. In the team's idiom, the framework named what they were already trying to do.

The framework was at the same time perceived as bureaucratically excessive in its formal scoring step. Participants did not characterise principal-and-interest scoring as wrong in principle; they characterised it as not earning its overhead under their conditions. The team's preferred evaluation lens, articulated across the focus group and interviews, was not numerical scoring but roadmap-relative cross-referencing — judging TD items against upcoming feature work and against each item's own merits, in discussion register. This articulation is consistent with Jaspan and Green's framing of TD as best understood through what engineers themselves report rather than through metric proxies alone [13], and the team's response is read as an articulated alternative — though one the team did not operationalise as a formal evaluation method during the trial — rather than as a dismissal.

The most prescriptive element of the team's perception was Theme T4: the team identified the framework's most valuable potential not in coordinating work within the team, but in communicating upward — to the tech lead, the CTO, and the wider organization. This reframing carried across observations and four of the five post-implementation interviews active for this code category. An anonymous post-survey response (POSTQ9, R3, one of n=4 anonymous responses) independently called for an "acceptable debt threshold" mechanism and is read here as one corroborating observation alongside the interview and observation evidence rather than as an independent confirming channel — a hedge consistent with Section 6.6.2. The interview-mediated convergence supports a reading of T4 as the team's own conclusion rather than as solely an artefact of facilitator presence (Section 4.5), without claiming that facilitator effects have been eliminated. Set against the literature on managers' perspective on TDM [22], the team's reframing identifies a register that intra-team frameworks have historically under-served: the framework's outputs as a basis for upward communication and resource argumentation, not only for within-team coordination.

Across these three elements — conceptual acceptance, operationally selective application, and prescriptive upward-facing reframing — the team's perception of the framework is internally coherent. The team accepted the framework as a mental model, applied the parts whose preconditions they had, and articulated the part whose value they did not yet realise. The perception of the framework as bureaucratic in its calculative step and as untapped in its upward-facing potential should be read together: both reflect the team's assessment of where the framework's design and the team's conditions align, and where they do not.

7.3 RQ3 — Benefits and Challenges

RQ3 asked what benefits and challenges arise from applying the framework in an Agile project. The case-study evidence places benefits and challenges on either side of a single axis — visibility — that links Theme T3 and Theme T1 and is partially mediated by the marking-gap finding of Section 6.5.5.

The principal benefit was visibility. The qualitative material across the recurring TD discussion forum, the team-level commitment to allocated time, and the routine recording of TD items in the issue tracker describes the team gaining a shared, observable picture of what TD it was carrying and how it was prioritising; the team's adoption of top-of-backlog ordering for TD items added a visible priority signal via standard Scrum backlog management. The post-survey item probing whether decisions about TD are transparent moved from a pre-mean

of 2.6 to a post-mean of 3.75 — a directional indicator of +1.15 against a widening of dispersion (std 0.55 to 1.26); the medians moved from 3 to 4 (Section 6.5.6). With n=5/n=4 unmatched, this is read as suggestive complementary evidence relative to the qualitative findings rather than as a confirmed effect (Section 8.3). The benefit is visibility-of-the-existing rather than identification-of-the-new, which Theme T3 names as the gap the framework actually addressed — identification was already saturated in the case team before the trial. Industry survey evidence reports substantial practitioner experience with TD identification and management [24], consistent with — though not equivalent to — the case-team's already-extensive baseline identification.

The principal challenges were capacity and marking discipline. Theme T1 documents the capacity ceiling: SLA-bound issues and feature crowding consumed the slack the framework would have used for sustained TD repayment. The pre-survey identified time and resource constraints as the dominant challenge; the post-survey did not contradict this; the closing reflection, the post-implementation interviews, and the open-ended POSTQ8 responses converged on the same constraint. Two readings of this finding apply in parallel: the framework made the team's relationship to capacity more legible to itself (a benefit), and capacity remained the operating constraint on what the framework could deliver during the trial (a challenge the framework did not alter).

The marking-gap finding of Section 6.5.5 sits on both sides. The same trial that produced the directional transparency shift also made visible a measurement artefact: the team's TD-repayment rate could be reported as 2.1%, 13%, or over 50% depending on the boundary chosen between three differently-defined operationalisations of the same data. As a benefit, the divergence is itself a transparency outcome — the team would not have produced or noticed it without the framework's discussion forum bringing measurement into routine view. As a challenge, the divergence indicates that the framework's measurement aspirations are conditional on a definitional convention the trial did not establish — what counts as a TD-repayment event, where the TD-versus-bug boundary is drawn, what feature work counts as incidentally repaying TD. This dual character is consistent with Freire et al.'s identification of monitoring-related impediments as a recognised category of TDM pitfall in Agile projects [27].

The benefits-and-challenges balance can therefore be summarised as visibility gained, capacity unchanged, and measurement made legible-but-incomplete.

7.4 RQ4 — Adapting the Framework for Agile Projects

RQ4 asked how the framework should be adapted to better support Agile projects. Theme T4 informs this question primarily, Theme T5 informs it secondarily, and Theme T3's marking-gap finding informs the measurement element. Four adaptation directions emerge from the case data; the recommendations stated below are subject to the limitations set out in Section 8.3.

First, the framework's outputs should be designed for upward communication, not only for intra-team coordination. The team's reframing carried across observations and four of the five post-implementation interviews whose recordings covered this theme; an anonymous post-survey response (POSTQ9, R3, one observation alongside the interview and observation evidence per Section 6.6.2) independently called for an "acceptable debt threshold" mechanism. Together, these sources identify a register the framework as designed underspecifies. A practical adaptation suggested by the case data is the addition of an upward-facing artefact to the framework — for instance, a periodic TD summary intended for management consumption that names not only what TD exists but what the team judges to be a tolerable upper bound and what the cost of staying above it is. Because the case team did not produce, prototype, or test such an artefact during the trial, this recommendation rests on the team's articulated potential rather than on observed evidence of effect; it is therefore offered as a hypothesis for further empirical evaluation, not as a finding established by this trial. The direction is consistent with the literature gap identified by Wiese and Borowa, who report that TDM frameworks have historically addressed the team perspective more thoroughly than the management perspective [22].

Second, the framework's calculative step (i.e., the ROI prioritisation step using principal-and-interest estimates, set out in Section 3.5.4) should be presented as conditional, not as default. The case team declined principal-and-interest scoring at kick-off because its preconditions — capacity slack to score and act, roadmap clarity to estimate interest — were assessed as absent. A revised framework would benefit from explicitly naming the conditions under which the ROI prioritisation step earns its overhead and the conditions under which structural-only adoption is sufficient. The team's preferred evaluation lens — roadmap-relative cross-referencing in discussion register — should be recognised as an articulated alternative the trial did not formally operationalise, rather than as an incomplete attempt to apply the framework. This would situate the calculative step within a broader family of

evaluation approaches whose applicability depends on context, consistent with the systematic mapping of TD prioritisation methods, which catalogues multiple mechanics without singling one out [4].

Third, the framework should be substrate-aware. Theme T5 documents that the framework's team-level mechanics were shaped by an organizational substrate — TD items disproportionately concerning one or two members of a fragmented team, fragmenting the team-level refinement the framework presupposes — and a codebase substrate of data-model debt the team identified as architecturally rooted [21]. A revised framework should acknowledge that the same set of practices will land differently depending on substrate, and that adaptation may include identifying which substrate conditions can be addressed by team-level practice, which require organization-level intervention, and which can only be worked around. Recognising the substrate as part of the framework's operating context — not as an obstacle external to it — is consistent with the broader conclusion that TD's architectural visibility shapes what team-level TDM can do [12], and with reports that management-level conditions frame what teams can accomplish in TDM [22].

Fourth, the framework's measurement aspirations are conditional on a definitional convention the case team did not establish. The 2.1% / 13% / 50% repayment-rate divergence (Section 6.5.5) is the most evocative empirical illustration of this in the trial: identical four-week data produced three plausible TD-repayment rates depending on the chosen boundary between TD-repayment, bug-fix, and feature work. A revised framework would benefit from including a defined-once tagging convention — what counts as a TD-tagged item, what counts as a TD-repayment event, where the TD-versus-bug boundary is drawn, and how incidental TD repayment during feature work is or is not counted — without which the framework's monitoring outputs are not interpretable across teams or across reporting periods. This recommendation directly addresses the measurement-aspiration gap that Theme T3's marking-gap finding made visible.

Taken together, the four adaptations point in two directions. Adaptations one through three converge on a common reading: read in this case, the framework's leverage was greater as a structuring scaffolding for shared practice than as a calculative engine for prioritisation, and its contribution can be increased by adding outward-facing artefacts and substrate-aware framing rather than by adding calculative machinery. The fourth adaptation — definitional conventions for measurement — is independent of this reading: it addresses an infrastructural

precondition for any monitoring claim the framework could support, regardless of whether the calculative step is retained or declined. These adaptations are bounded by the four falsifiers Chapter 6 set out, none of which occurred during the four-week observation: (i) a sustained 20% TD allocation displacing feature work; (ii) a documented operationalisation of the ROI prioritisation step (or a comparable formal substitute adopted as a written method); (iii) an upward communication event in which the framework's outputs reached management and produced a resource decision; (iv) a period of TD repayment that the framework rather than external pressure drove. Any of these occurring would falsify the adaptations proposed above. Falsifiers (i)–(iv) bear most directly on the structural-versus-calculative reading; the measurement-tagging adaptation has its own implicit falsifier — a defined-once convention failing to reduce the 2.1%/13%/50% divergence in subsequent trials.

7.5 Implications for Long-Lived Agile Projects

The case team's situation exemplifies one combination of conditions sometimes encountered in long-lived Agile projects: accumulated technical debt the team identified as architecturally rooted [21], sustained external SLA pressure, feature roadmaps that compete continuously with TD-repayment work, and a fragmented team-structural substrate (Theme T5). Each condition shaped what the framework could do in this trial, and the readings below are accordingly conditional on conditions resembling these. Four readings can be carefully drawn from the case-study evidence; each is offered as a conjecture inviting falsification rather than as an established pattern. Three of the four readings parallel the adaptations identified in Section 7.4; the measurement-tagging adaptation (Section 7.4 fourth) is project-internal infrastructure rather than a generalisable implication and is not repeated below, and a visibility reading is offered here as an interpretation rather than as an actionable adaptation since the visibility benefit is treated as an outcome of the framework's structural elements rather than a separate adaptation.

A first reading, conditional on conditions resembling the case team's, is that the calculative step (the ROI prioritisation step using principal-and-interest estimates) is conditional rather than default. The case team's decision to decline principal-and-interest scoring is read in this study as conditional on absent capacity slack and absent roadmap clarity. For long-lived projects in similar conditions, frameworks that present numerical prioritisation as a default mechanism may impose preconditions that are not met. A more useful framing — under these conditions — is to present the calculative step as one of several evaluation modes whose

applicability depends on the project's current capacity-and-roadmap conditions. In conditions resembling the case team's, structural-only adoption — discussion forum, signalling, allocated time, and roadmap-relative cross-referencing in discussion register — may be sufficient and may earn more compliance than a calculative mechanism the team's conditions do not support. This reading would be falsified by a comparable team adopting and sustaining numerical prioritisation under similar conditions.

A second reading is that visibility, not identification, is where the framework added value in the case team. Theme T3 documents that identification was a saturated baseline before the trial; the framework's contribution was to give the team a shared, observable picture of what they were carrying and how they were prioritising. The +1.15 directional shift on the transparency item is consistent with this reading without confirming it (n=5/n=4 unmatched, with the dispersion widening from std 0.55 to 1.26 cautioning against uniform-improvement readings; see Section 8.3). For long-lived projects where TD is generally already extensive and known, the practical implication is that frameworks may benefit from optimising for visibility-of-the-existing rather than for identification-of-the-new — shifting the centre of gravity from identification machinery toward documentation, signalling, and forum practices that make existing TD legible at a cadence the team can sustain. This reading would be falsified by a comparable trial in which identification mechanisms produced new-TD discoveries the team treated as material.

A third reading is that upward-facing artefacts are an underdeveloped leverage point. The team's most strongly articulated unrealised potential for the framework was upward-facing — translation of the team's TD picture into a form management could act on. The anonymous post-survey response (POSTQ9, R3, one observation alongside the interview and observation evidence per Section 6.6.2) calling for an "acceptable debt threshold" mechanism is one concrete instantiation of this. In long-lived projects where capacity decisions are made above the team but the data on TD is held within the team, framework designs that include outward-facing artefacts — periodic summaries, threshold proposals, escalation paths — may address a gap that intra-team mechanics cannot close. This reading is consistent with Wiese and Borowa's identification of the manager-perspective gap in TDM literature [22], and would be falsified by a trial in which a comparable team produced and used such artefacts without the predicted resource-decision effect.

A fourth reading is that substrate is part of a framework's operating context, not external to it. Theme T5 frames the codebase substrate (data-model debt the team identified as architecturally rooted) and the team-structural substrate (TD items disproportionately concerning a subset of a fragmented team) as conditions shaping what the framework can do, not as obstacles outside its scope. For long-lived projects, this implies that a framework's adaptation cannot be specified independently of its substrate. The same set of practices may land differently in a team with shared codebase ownership than in one with fragmented ownership, and in a codebase with isolatable structural debt than in one whose architectural roots are systemic. This reading would be falsified by a comparable framework producing comparable effects across substantively different substrates.

Set against the prior work this thesis builds upon, the readings above sharpen rather than overturn earlier framing. Guo and Seaman's TDM line of work [2], [6] — including the explicit portfolio mechanic of [6] — proposed numerical decision-support mechanics; the case team's experience suggests these mechanics can be conditional rather than default in long-lived Agile projects. The thesis's own bachelor's-thesis precursor [7] inherited the calculative step from Guo and Seaman's portfolio formulation [6] without empirically testing its preconditions; the case-study evidence in the present work suggests that the calculative step — analytically separated in Section 7.1 — is one component whose enabling conditions need to be made explicit. The present case-study findings identify upward-facing artefacts, definitional conventions for measurement, and substrate-aware framing as concrete adaptation directions for subsequent investigation.

These four implications are drawn from a single team's experience over four weeks. They are offered as carefully argued readings of the case-study evidence rather than as established patterns; their further evaluation in additional projects, teams, and substrates is among the directions for future research that Chapter 8 returns to.

8 Conclusions

This chapter concludes the thesis. Section 8.1 summarises the empirical findings that answer the four research questions of Section 1.3. Section 8.2 names the contributions the work makes to the technical-debt-management literature. Section 8.3 sets out the limitations bearing on these findings, and Section 8.4 identifies directions for future research, including the empirical falsification conditions presented in Chapters 6 and 7.

8.1 Summary of Findings

This thesis evaluated a refined Technical Debt Management Framework (TDMF) — set out in Chapter 3 — in a single Agile software development team over a four-week trial. The case study combined a focus group, ten observed team meetings, five post-implementation interviews, and pre- and post-implementation Likert questionnaires (n=5 / n=4 unmatched), analysed through reflexive thematic analysis [29] and group-level descriptive statistics. Five themes were constructed from the qualitative data; quantitative results were reported as suggestive complementary evidence consistent with the AI-assisted analysis workflow described in Section 4.4 and Appendix 4.

The descriptive answer to RQ1 is that the framework was applied selectively. The framework's structural layer — recurring TD discussion forum, allocated time, and recording of TD items in the centralised backlog — was set up at kick-off as designed; sustained enactment was partial across each element (the 20% allocation did not sustain under capacity pressure, marking was inconsistent, and TD-focused discussion was concentrated in the two refinement sessions of the four-week trial). The framework's visualisation and monitoring design (Section 3.5.6) was not adopted as a team practice during the trial and is therefore not represented in this enactment account. The team additionally enacted the framework's prioritisation intent through standard Scrum backlog ordering (top-of-backlog signalling), in lieu of the framework's ROI prioritisation step. Its calculative layer — the ROI prioritisation step (using principal-and-interest estimates without the interest-probability weighting of the original TDMF) — was declined at the kick-off refinement on the team's prior assessment that its preconditions were not met, and was not subsequently reinstated. The gap between intended and enacted application is therefore conditional on capacity (Theme T1) and on the substrate against which the framework operated (Theme T5), not on a global rejection of the framework.

In response to RQ2, the team's perception was three-part: the framework's mental model received broadly positive reception among respondents who answered; the calculative step was perceived as bureaucratically excessive in the team's conditions; and the framework's most prescriptive value was identified as upward-facing — translation of the team's TD picture into a form management could act on (Theme T4).

In response to RQ3, the principal benefit was visibility — a directional +1.15 shift on the post-survey transparency item, read as suggestive complementary evidence rather than as a confirmed effect (n=5/n=4 unmatched, with dispersion widening from std 0.55 to 1.26). The principal challenges were capacity and marking discipline, with the 2.1% / 13% / 50% repayment-rate divergence (Section 6.5.5) sitting on both sides as benefit (transparency outcome) and challenge (definitional convention absent).

In response to RQ4, four adaptation directions emerged: the framework's outputs should be designed for upward communication; the calculative step should be presented as conditional rather than default; the framework should be substrate-aware; and the framework's measurement aspirations are conditional on a definitional convention the trial did not establish. The first three converge on a common reading; the fourth is independent and addresses an infrastructural precondition.

A central caveat applies. Three of the chapter's positive findings (Themes T2, T3, T4) rest substantially on interview material elicited by the framework's designer in person. Section 8.3 acknowledges that facilitator-presence effects on positive readings cannot be ruled out, since co-worker rapport with the framework's designer would predict inflation rather than deflation; the partial mitigations available — sustained dissent on the calculative step across all qualitative channels, and the corroborating anonymous post-survey observation on T4 — apply asymmetrically. The robust contributions of this case study are accordingly its negative findings (declined calculative step, capacity ceiling, marking gap); the positive readings are reported with this caveat in place.

8.2 Contributions

The thesis contributes empirically, conceptually, practically, and procedurally. The conceptual and practical contributions are offered subject to the asymmetric-mitigation caveat of Section 8.1, where they rest on positive findings (Themes T2, T3, or T4).

The empirical contribution is a case-study evidence base on how the refined TDMF behaves in conditions characteristic of long-lived Agile projects: acute SLA pressure, feature-roadmap competition, and a substrate of architectural-root data-model debt with team-structural fragmentation. Of these, time pressure and feature-roadmap competition are well-documented in industry surveys [24]; the substrate and team-structural conditions are case-specific. Such combinations are underrepresented in published case studies of TDM application; the four-week trial documented in Chapters 5 and 6 is contributed as one such case.

The conceptual contribution comprises two analytic moves offered for further empirical evaluation. The first is the structural-versus-calculative layer partition of the refined TDMF (Section 7.1), which separates the framework's discussion-and-signalling scaffolding from its numerical prioritisation step. The decomposition is analyst-imposed rather than a property of the framework as designed in Chapter 3; the case team's selective adoption pattern maps cleanly onto it and motivates its further use in subsequent empirical work. The second is the visibility-not-identification framing (Section 7.3): the case-team's identification of TD was already saturated before the trial, and the framework's contribution was to surface what the team was carrying rather than discover new instances. This framing departs from the identification-first sequencing implicit in many TDM activity catalogues [4] and is offered as a hypothesis worth empirical comparison in projects with comparable maturity.

The practical contribution is the four adaptation directions of Section 7.4 — upward-facing artefacts, conditional calculative step, substrate-aware framing, and definitional measurement conventions — together with their case-study grounding. Each is grounded in the trial evidence but is offered as a hypothesis for subsequent empirical evaluation rather than as a finding established by this trial.

The methodological contribution is a worked example of an AI-assisted analysis workflow under researcher control (Section 4.4 and Appendix 4), in which an AI assistant proposed initial codes, computed descriptive statistics, and generated cross-source summaries while the researcher retained interpretive authority over coding decisions, theme construction, and analytical claims. The workflow is reported in sufficient detail that subsequent researchers can adopt, adapt, or critique the boundaries between AI-assisted and researcher-conducted analysis. The contribution is procedural: the underlying reflexive thematic analysis framework [29] is unchanged; what is contributed is a documented integration of AI assistance with reflexive thematic analysis.

The empirical and practical contributions add team-level case-study depth on the refined TDMF as enacted process. The conceptual and methodological contributions are offered in the same hedged register: as readings the case study supports rather than as claims it has independently established.

8.3 Limitations

This study adopts an exploratory case study approach to examine the practical applicability of a refined TDMF in a real-world Agile context. The findings should be interpreted with certain limitations in mind. First, the refined TDMF evaluated in this study was designed by the researcher based on prior literature and earlier work. To address potential confirmation bias, the study explicitly focused on identifying deviations, constraints, and mismatches between the intended theoretical application and actual practice. The findings therefore reflect not only successful aspects of the framework adoption but also limitations, challenges, and areas requiring further adaptation. These considerations support the credibility and transparency of the research while acknowledging its contextual and exploratory nature.

Second, the researcher's insider role in the organization may introduce bias. This risk was mitigated through multiple measures, including voluntary participation, anonymization of participants, the use of several data collection methods, and methodological triangulation across focus groups, observations, interviews, and questionnaires. Rather than viewing the insider position solely as a limitation, it also enabled rich contextual understanding access to day-to-day practices that would be difficult to obtain through external observation role.

Within this insider position, all five post-implementation interviews — which carry substantial interpretive load for Themes T2, T3, and T4 — were conducted by the researcher in the dual role of framework designer and analyst (Section 4.5). The partial mitigations documented in Section 4.5 apply asymmetrically: sustained dissent on the calculative step across the focus group, observations, and post-implementation interviews indicates that scepticism toward the framework's central mechanic was reportable to its designer in person, and the corroborating anonymous post-survey observation on Theme T4 noted in Section 7.2 reduces the alternative reading that this finding is solely an artefact of facilitator presence. These mitigations bear on the negative findings (declined P/I, capacity ceiling, marking gap) and on T4 specifically; they do not rule out facilitator effects on the positive readings (T2 conceptual acceptance, T3 visibility), where co-worker rapport with the framework's designer would predict inflation rather than deflation. T2 and T3 carry partial cross-channel

triangulation through POSTQ8 free-text responses on visibility and POSTQ9, R1 on the challenge of choosing the next item, though both channels share the trial-design context.

Third, the pre-implementation questionnaire received five responses (n=5) and the post-implementation questionnaire received four (n=4, response rate 80% after a one-week reminder cycle). Both surveys collected response timestamps but no respondent identifiers and paired pre-post analysis at the individual level is therefore not possible. All quantitative results in Chapter 6 are reported as group-level descriptive statistics — means, medians, standard deviations, and response distributions — consistent with the AI-Assisted Analysis Workflow described in Section 4.4 and Appendix 4, which explicitly excluded inferential testing on these sample sizes. Directional changes between pre and post means, including the largest observed shift of +1.15 on the transparency item (Section 6.5.6), are read as suggestive complementary evidence relative to the qualitative findings rather than as confirmed effects.

Fourth, one of the five post-implementation interviews was truncated by a recording failure after the interview themes covering overall experience and perceived usefulness had been captured but before the improvement-and-reflection portion began (Section 5.3.2). Cross-source analyses for Theme T3 (Section 6.5), Theme T4 (Section 6.6), and Theme T5 (Section 6.7) therefore draw on four of the five informants active for those code categories; the truncated interview's content for the earlier themes was retained and contributes to Theme T1 and Theme T2. The absent content cannot be inferred from the present content; cross-source convergence for T3, T4, and T5 is therefore established on a four-informant interview base rather than five and is read with that limit in mind.

Fifth, the trial spanned a single four-week observation window. Within-trial effects can be reported and have been reported in Chapter 6, but whether the team's selective adoption pattern would consolidate, drift, or revert under a longer observation horizon — whether the calculative step would be reinstated as capacity opened, whether the upward-facing potential would be realised through artefact production, or whether the visibility gain would persist — is not adjudicable from this design. Findings should therefore be read as describing how the framework worked in this team during these four weeks, not as predictions of how it would work over a longer horizon, in different teams, or under different conditions.

Finally, the study focuses on a single software development team, which limits statistical generalizability. However, the aim of this study is not statistical generalization but providing empirically grounded insights into how a theoretically proposed framework is interpreted,

adapted, and enacted in practice. The detailed description of the organizational context, team structure, and development practices support transferability to similar Agile projects.

8.4 Directions for Future Research

The case-study evidence supports five lines of subsequent inquiry.

The first is falsification trials. Each of the readings in Sections 7.4 and 7.5 is offered with explicit falsifying conditions. Comparable trials in which a sustained 20% TD allocation displaces feature work, the ROI prioritisation step is operationalised in documented form (or a comparable formal substitute adopted as a written method), framework outputs reach management and produce a resource decision, or TD repayment is framework-driven rather than externally driven would falsify the corresponding adaptations. Each is independent of the others and can be tested in separate trials. The measurement-tagging adaptation has its own implicit falsifier — a defined-once convention failing to reduce the 2.1% / 13% / 50% divergence in subsequent trials.

The second is multi-team and substrate-conditional studies. The substrate-aware adaptation direction predicts that the same framework will land differently in teams with shared codebase ownership versus fragmented ownership, and in codebases with isolatable structural debt versus systemic architectural roots [21]. Comparative studies across substrates — selected to vary on these dimensions — would test the strength of the substrate dependence and identify which framework practices are most substrate-sensitive. Such work would also speak to the conclusion that TD's architectural visibility shapes what team-level TDM can do [12].

The third is longitudinal studies. The four-week observation window of this case study cannot adjudicate sustainability or drift. Longer-term studies — six months, twelve months, or repeated across release cycles — would examine whether the team's selective adoption pattern consolidates, drifts, or reverts as capacity changes; whether the calculative step is reinstated when its preconditions return; and whether the visibility gain persists.

The fourth is upward-facing artefact prototyping. The case team did not produce, prototype, or test the upward-facing artefact (Section 7.4 first adaptation) the team articulated they wished existed. Prototyping such an artefact — for instance, a periodic TD summary with tolerable-upper-bound and cost-of-staying-above-threshold information — and evaluating it in trials with management as the intended audience would test whether the articulated

potential converts into observed effect on resource decisions. This direction connects with Wiese and Borowa's manager-perspective programme [22].

The fifth is empirical work on definitional measurement conventions. The 2.1% / 13% / 50% repayment-rate divergence (Section 6.5.5) made visible a measurement artefact whose resolution is not internal to the framework. Empirical work that defines, applies, and evaluates a candidate convention — what counts as a TD-tagged item, what counts as a TD-repayment event, where the TD-versus-bug boundary is drawn — would test whether convergent monitoring outputs become possible across teams and across reporting periods. Such work would also inform the framework's measurement aspirations, which are currently underspecified.

Across all five directions, the present case study is offered as one empirical anchor rather than as a definitive evaluation. The strength of subsequent inquiry will depend on how cumulatively its readings are tested against trials that vary the conditions identified here as material. The work this thesis contributes is one such anchor for a programme of empirical research the field of technical debt management continues to need [4], [24].

References

[1] W. Cunningham, 'The WyCash portfolio management system', in *Addendum to the proceedings on Object-oriented programming systems, languages, and applications (Addendum)*, in OOPSLA '92. New York, NY, USA: Association for Computing Machinery, Dec. 1992, pp. 29–30. doi: 10.1145/157709.157715.

[2] Y. Guo and C. Seaman, 'MEASURING AND MONITORING TECHNICAL DEBT', Jan. 2016, Accessed: Oct. 04, 2025. [Online]. Available: http://hdl.handle.net/11603/15618

[3] J. Holvitie, 'Technical Debt in Software Development : Examining Premises and Overcoming Implementation for Efficient Management', Apr. 2017, Accessed: Nov. 16, 2025. [Online]. Available: https://www.utupub.fi/handle/10024/134086

[4] Z. Li, P. Avgeriou, and P. Liang, 'A systematic mapping study on technical debt and its management', *J. Syst. Softw.*, vol. 101, pp. 193–220, Mar. 2015, doi: 10.1016/j.jss.2014.12.027.

[5] A. Aldaeej, A. Nguyen-Duc, and V. Gupta, 'A Lean Approach of Managing Technical Debt in Agile Software Projects – A Proposal and Empirical Evaluation', in *Agile Processes in Software Engineering and Extreme Programming*, C. J. Stettina, J. Garbajosa, and P. Kruchten, Eds, Cham: Springer Nature Switzerland, 2023, pp. 67–76. doi: 10.1007/978-3-031-33976-9_5.

[6] Y. Guo and C. Seaman, 'A portfolio approach to technical debt management', in *Proceedings of the 2nd Workshop on Managing Technical Debt*, Waikiki, Honolulu HI USA: ACM, May 2011, pp. 31–34. doi: 10.1145/1985362.1985370.

[7] M. Järvinen, 'Teknisen velan hallitseminen Scrumissa', University of Turku, Turku, 2023.

[8] F. Shull, J. Singer, and D. I. K. Sjøberg, Eds, *Guide to Advanced Empirical Software Engineering*. London: Springer, 2008. doi: 10.1007/978-1-84800-044-5.

[9] C. Wohlin and A. Aurum, 'Towards a decision-making structure for selecting a research design in empirical software engineering', *Empir. Softw. Eng.*, vol. 20, no. 6, pp. 1427–1455, Dec. 2015, doi: 10.1007/s10664-014-9319-7.

[10] Kent Beck *et al.*, 'Manifesto for Agile Software Development'. Accessed: Oct. 26, 2025. [Online]. Available: https://agilemanifesto.org/iso/en/manifesto.html

[11] Ken Schwaber and Jeff Sutherland, 'Scrum Guide | Scrum Guides'. Accessed: Oct. 26, 2025. [Online]. Available: https://scrumguides.org/scrum-guide.html

[12] P. Kruchten, R. L. Nord, and I. Ozkaya, 'Technical Debt: From Metaphor to Theory and Practice', *IEEE Softw.*, vol. 29, no. 6, pp. 18–21, Nov. 2012, doi: 10.1109/MS.2012.167.

[13] C. Jaspan and C. Green, 'Defining, Measuring, and Managing Technical Debt', *IEEE Softw.*, vol. 40, no. 3, pp. 15–19, May 2023, doi: 10.1109/MS.2023.3242137.

[14] M. Fowler, 'Technical Debt Quadrant', martinfowler.com. Accessed: Oct. 29, 2025. [Online]. Available: https://martinfowler.com/bliki/TechnicalDebtQuadrant.html

[15] J.-L. Letouzey and M. Ilkiewicz, 'Managing Technical Debt with the SQALE Method', *IEEE Softw.*, vol. 29, no. 6, pp. 44–51, Nov. 2012, doi: 10.1109/MS.2012.129.

[16] N. Davis, 'Driving Quality Improvement and Reducing Technical Debt with the Definition of Done', in *2013 Agile Conference*, Aug. 2013, pp. 164–168. doi: 10.1109/AGILE.2013.21.

[17] F. Oliveira, A. Goldman, and V. Santos, 'Managing Technical Debt in Software Projects Using Scrum: An Action Research', in *2015 Agile Conference*, Aug. 2015, pp. 50–59. doi: 10.1109/Agile.2015.7.

[18] J. Perera, E. Tempero, Y.-C. Tu, and K. Blincoe, 'Quantifying Technical Debt: A Systematic Mapping Study and a Conceptual Model', Mar. 12, 2023, *arXiv*: arXiv:2303.06535. doi: 10.48550/arXiv.2303.06535.

[19] N. Zazworka, C. Seaman, and F. Shull, 'Prioritizing design debt investment opportunities', in *Proceedings of the 2nd Workshop on Managing Technical Debt*, in MTD '11. New York, NY, USA: Association for Computing Machinery, May 2011, pp. 39–42. doi: 10.1145/1985362.1985372.

[20] W. Snipes, B. Robinson, Y. Guo, and C. Seaman, 'Defining the decision factors for managing defects: A technical debt perspective', in *2012 Third International Workshop on Managing Technical Debt (MTD)*, Jun. 2012, pp. 54–60. doi: 10.1109/MTD.2012.6226001.

[21] R. Kazman *et al.*, ‘A Case Study in Locating the Architectural Roots of Technical Debt’, in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, May 2015, pp. 179–188. doi: 10.1109/ICSE.2015.146.
[22] M. Wiese and K. Borowa, ‘IT managers’ perspective on Technical Debt Management’, *J. Syst. Softw.*, vol. 202, p. 111700, Aug. 2023, doi: 10.1016/j.jss.2023.111700.
[23] P. S. Medeiros dos Santos, A. Varella, C. R. Dantas, and D. B. Borges, ‘Visualizing and Managing Technical Debt in Agile Development: An Experience Report’, in *Agile Processes in Software Engineering and Extreme Programming, Xp 2013*, H. Baumeister and B. Weber, Eds, in Lecture Notes in Business Information Processing, vol. 149. Berlin: Springer-Verlag Berlin, 2013, pp. 121–134.
[24] R. Ramač *et al.*, ‘Prevalence, common causes and effects of technical debt: Results from a family of surveys with the IT industry’, *J. Syst. Softw.*, vol. 184, p. 111114, Feb. 2022, doi: 10.1016/j.jss.2021.111114.
[25] Z. Codabux and B. Williams, ‘Managing technical debt: an industrial case study’, in *Proceedings of the 4th International Workshop on Managing Technical Debt*, in MTD ’13. San Francisco, California: IEEE Press, May 2013, pp. 8–15.
[26] G. M. Q. M. Archela, A. C. V. Melo, and V. L. Gava, ‘Technical Debt Management in Agile Context: A new framework and case study in a large financial institution’, in *Proceedings of the 39th ACM/SIGAPP Symposium on Applied Computing*, in SAC ’24. New York, NY, USA: Association for Computing Machinery, May 2024, pp. 826–833. doi: 10.1145/3605098.3635946.
[27] S. Freire *et al.*, ‘Pitfalls and Solutions for Technical Debt Management in Agile Software Projects’, *IEEE Softw.*, vol. 38, no. 6, pp. 42–49, Nov. 2021, doi: 10.1109/MS.2021.3101990.
[28] P. Jarvinen, ‘Research Questions Guiding Selection of an Appropriate Research Method’, *ECIS 2000 Proc.*, Jan. 2000, [Online]. Available: https://aisel.aisnet.org/ecis2000/26
[29] V. Braun and V. Clarke, ‘Reflecting on reflexive thematic analysis’, *Qual. Res. Sport Exerc. Health*, vol. 11, no. 4, pp. 589–597, Aug. 2019, doi: 10.1080/2159676X.2019.1628806.

Appendices

Appendix 1 Focus Group Guide

The purpose of the focus group is to examine how the theoretically proposed refined TDMF aligns with the team's current practices, and to identify context-specific adaptations needed for practical application.

Phase 1: Current State of Technical Debt Management

FG1.1 How would you describe the way technical debt is currently identified in this project?

FG1.2 How is technical debt currently documented or tracked, if at all?

FG1.3 How are decisions made about whether and when technical debt should be addressed?

FG1.4 What challenges do you experience in managing technical debt in your day-to-day work?

Follow-up questions if needed:

FG1.F1 Are these practices explicit or informal?

FG1.F2 Who is typically involved in these decisions?

Phase 2: Presentation of the Refined TDMF

Framework walkthrough, no questions required.

The researcher presents the original TDMF by Seaman and Guo, including the four main artefacts (TDL, identification, evaluation, decision making).

The researcher continues to present the refined TDMF, including:

- Design principles underlying the framework
- Core activities (identification, evaluation, decision making) and how they are mapped to Agile artefacts
- Roles and responsibilities

- Example case of managing a TDI from identification to repayment

Participants are encouraged to ask clarifying questions.

Phase 3: Alignment and Contextualization

FG3.1 Which parts of the proposed framework feel intuitive or already familiar?

FG3.2 Which parts feel unclear, unrealistic, or difficult to apply in your current context?

FG3.3 Which events or practices in your team could realistically support?
FG3.3a Technical Debt identification?
FG3.3b Technical Debt evaluation?
FG3.3c Technical Debt decision making, prioritization and planning?

FG3.4 What, if anything, would need to change for the refined TDMF to work in your team?

FG3.5 Are there parts of the framework that should be simplified, emphasized, or omitted?

FG3.6 Based on this discussion, how should the refined TDMF be applied in this project?

FG3.7 Can we agree on a preliminary plan for integrating the framework into your existing process?

Appendix 2 Interview Guide

The purpose of the interview is to capture individual participant's perceptions and experiences of using the refined TDMF as part of their Agile development process.

Theme 1: Overall Experience (RQ2)

INT1.1 How would you describe your overall experience using the refined TDMF?

INT1.2 To what extend did the framework fit your way of working?

Theme 2: Perceived Usefulness (RQ2, RQ3)

INT2.1 Did the framework help you better understand or manage technical debt? How?

INT2.2 Which parts of the framework did you find most useful?

INT2.3 Were there any parts of the framework that felt unnecessary or difficult to use?

Theme 3: Impact on Day-to-Day Work (RQ2, RQ3)

INT3.1 Did using the framework change how technical debt was discussed or prioritized in the team?

INT3.2 Did it affect how you plan or carry out development work?

Theme 4: Challenges and Limitations (RQ3)

INT4.1 What challenges did you experience when applying the framework?

INT4.2 Were there situations where the framework did not work well?

Theme 5: Improvement and Reflection (RQ4)

INT5.1 How could the framework we improved to better support teams like yours?

INT5.2 Would you recommend using this framework in similar projects? Why or why not?

Appendix 3 Questionnaires

Pre-implementation Questionnaire

Participants were asked to indicate their level of agreement on a 5-point Likert scale.

PREQ1 Technical debt is clearly identified in this project.

PREQ2 Technical debt is documented in a consistent and visible manner.

PREQ3 Technical debt is discussed regularly within the team.

PREQ4 Decisions about addressing technical debt are transparent.

PREQ5 Technical debt is prioritized in a systematic way.

PREQ6 There is sufficient time allocated to addressing technical debt.

PREQ7 I feel that technical debt is managed well in the team.

Open ended question:

PREQ8 What do you see as the main challenges in managing technical debt in this project?

Post-implementation questionnaire

Using the same 5-point Likert scale:

POSTQ1 Technical debt is clearly identified in this project.

POSTQ2 Technical debt is documented in a consistent and visible manner.

POSTQ3 Technical debt is discussed regularly within the team.

POSTQ4 Decisions about addressing technical debt are transparent.

POSTQ5 Technical debt is prioritized in a systematic way.

POSTQ6 There is sufficient time allocated to addressing technical debt.

POSTQ7 I feel that technical debt is managed well in the team.

Open ended questions:

POSTQ8 In what ways, if any, did the refined TDMF improve technical debt management?

POSTQ9 What challenges or limitations did you experience when using the refined TDMF?

Appendix 4 AI-Assisted Analysis Workflow

This appendix describes the workflow used for AI-assisted data analysis in this study, summarizing how Claude Code by Anthropic was used as a research assistant during the analysis of the questionnaire and qualitative data.

Tool description

Claude Code is a command-line tool developed by Anthropic that provides an interactive AI assistant powered by the Claude large language model. The version used in this research was powered by the Claude Opus 4.7 model with 1M context. The tool operates on the researcher's local file system and can read data files, write and execute analysis scripts, and process textual data within a single context window of approximately one million tokens.

Outputs from the tool are non-deterministic; running the same prompt may produce slightly different results.

Positioning

Claude Code was used as a research assistant, not an analyst. The AI proposed, processed, and generated outputs; the researcher reviewed, validated, modified, or rejected those outputs and made all interpretive decisions and analytical claims.

Quantitative analysis workflow

Claude Code was used as a research assistant, not an analyst. The AI proposed, processed, and generated outputs; the researcher reviewed, validated, modified, or rejected those outputs and made all interpretive decisions and analytical claims.

For each questionnaire round (pre- and post-implementation):

1. The researcher validated the data file and specified the analyses to be performed.
2. Claude Code generated and executed a Python script using pandas and numpy to compute per-item means, medians, standard deviations, minimum and maximum values, and response distributions.
3. The researcher reviewed the outputs, spot-checking calculations and verifying the data parsing.
4. The researcher interpreted the results in the context of the case study.

For the pre/post comparison, Claude Code generated change metrics across both rounds; the researcher interpreted observed shifts considering qualitative findings, avoiding causal claims given the small sample size.

Qualitative analysis workflow

The qualitative analysis followed the thematic analysis approach described in Section 4.4.1, with AI assistance in selected phases:

Familiarization (researcher only, no AI): The researcher read each transcript without AI involvement to develop own initial impressions before any AI-generated patterns could introduce anchoring bias.

Initial coding (AI proposes, researcher validates): For each transcript, Claude Code was prompted to propose initial codes with definitions and line-level references to the source text. The researcher reviewed every proposed code against the source, accepting, modifying, or rejecting each. New codes were added based on the researcher's reading and contextual knowledge. For interview transcripts coded after the focus group, Claude Code applied the existing codebook and proposed new codes where needed; all assignments were validated by the researcher.

Theme development (researcher-led, AI-supported): The researcher developed themes from the validated codes. Claude Code suggested possible groupings and assisted with mechanical tasks such as searching across transcripts and re-sorting coded segments under revised themes.

Writing (researcher only): Analytical narratives, theme definitions, and interpretive claims were written by the researcher without AI assistance.

Observation note analysis

Field notes from the observation period were organized and indexed with assistance from Claude Code. The researcher coded the notes against the existing codebook; Claude Code assisted with searching for cross-references between notes and other data sources. All coding decisions and interpretations were made by the researcher.

Triangulation

Claude Code generated structured summaries of findings from each data source. The researcher identified convergences and divergences across sources and wrote the integrated triangulation narrative.

Triangulation

Claude Code generated structured summaries of findings from each data source. The researcher identified convergences and divergences across sources and wrote the integrated triangulation narrative.

Limitations

Claude Code does not have access to the contextual knowledge that the researcher acquired through direct involvement with the case organization.

Outputs are non-deterministic, meaning the same prompt may produce slightly different results on different runs. The researcher's contextual knowledge served as a quality check on AI outputs.