

Preventing technical debt with the TAP framework for Technical Debt Aware Management

Marion Wiese^{a,*}, Paula Rachow^a, Matthias Riebisch^a, Julian Schwarze^b

^aUniversität Hamburg, Department of Informatics, Vogt-Kölln-Str.30, 22527 Hamburg

^bGruner + Jahr GmbH, Information Technology, Baumwall 11, 22459 Hamburg

Abstract

Context. Technical Debt (TD) is a metaphor for technical problems that are not visible to users and customers but hinder developers in their work, making future changes more difficult. TD is often incurred due to tight project deadlines and can make future changes more costly or impossible. Project Management usually focuses on customer benefits and pays less attention to their IT systems' internal quality. TD prevention should be preferred over TD repayment because subsequent refactoring and re-engineering are expensive.

Objective. This paper evaluates a framework focusing on both TD prevention and TD repayment in the context of agile-managed projects. The framework was developed and applied in an IT unit of a publishing house. The unique contribution of this framework is the integration of TD management into project management.

Method. The evaluation was performed as a comparative case study based on ticket statistics and two structured surveys. The surveys were conducted in the observed IT unit using the framework and a comparison unit not using the framework. The first survey targeted team members, the second one IT managers.

Results. The evaluation shows that in this IT unit the TAP framework led to a raised awareness for the incurrence of TD. Decisions to incur TD are intentional, and TD is repaid timelier. Unintentional TD incurred by unconscious decisions is prevented. Furthermore, better communication and better planning of the project pipeline can be observed.

Conclusions. We provide an insight into practitioners' ways to identify, monitor, prevent and repay TD. The presented framework includes a feasible method for TD prevention despite tight timelines by making TD repayment part of project management.

Keywords: Technical Debt, Project Management, Technical Debt Awareness, Technical Debt Repayment, Technical Debt Prevention, Technical Debt Backlog

1. INTRODUCTION

Software systems are subject to frequent changes due to new or changing requirements or changes in the environment. Hence, a central goal in the development of software systems should be to implement these systems in such a way that they are easily changeable over a long time. Many different causes limit this ease of changeability and should be addressed. Because these modifications usually provide no visible benefit to non-developers and are challenging to understand, they are often given a lower priority by business leaders. Cunningham introduced the metaphor of Technical Debt (TD) to make this problem better understandable to business managers [1].

Avgeriou et al. define TD as “a collection of design or implementation constructs that are expedient in the short

term, but set up a technical context that can make future changes more costly or impossible” [2]. In a technical metaphor for financial debt, a sub-optimal implementation or design is interpreted as debt. The resulting problems are interpreted as interest rates, refactoring as repayment, and refactoring cost as principal.

In 2016, Avgeriou et al. claimed about TD: “It’s the next big thing, and it’s messy” [3]. TD is a serious problem in practice, often caused by tight deadlines due to fast-changing requirements in the age of digitalization [4, 2, 5].

McConnell and Fowler differentiate between intentional and unintentional incurred TD [6, 7]. Intentional TDs result from a conscious decision, e.g., implementing a workaround to reach a set deadline. Unintentional TDs result from mistakes, e.g., poor design at the code level or architectural decisions that, over time, turn out to have been wrong.

TD prevention is stated as the preferable option for TD management by many practitioners [8, 9, 10]. In a long-term view, implementing the optimal solution right from the beginning is cheaper than incurring TD, especially when considering possible interest payments [11, 4].

*Corresponding author

Email addresses: marion.wiese@uni-hamburg.de (Marion Wiese), paula.rachow@uni-hamburg.de (Paula Rachow), matthias.riebisch@uni-hamburg.de (Matthias Riebisch), schwarze.julian@guj.de (Julian Schwarze)

TD prevention primarily targets unintentional TD. To enable TD prevention, two main aspects must be considered:

1. Stakeholders have to be aware of TD to make conscious decisions [12, 13, 14].
2. Stakeholders have to understand the causes of TD [4, 15].

TD Awareness. Awareness is self-evident for self-admitted TD, i.e., ToDo-Statements in the code, and TD found by tools. In practice, TD prevention is often focused on these kinds of TD [16, 10]. In research, the perception of design and architecture decisions and the TD incurred by these is less evident and less focused on.

TD backlogs and lists lead to more visibility, make TD explicit, and raise the perception of TD. This, in turn, raises the overall awareness for TD [17, 18, 19, 20].

The awareness leads to better communication between developers and IT management, which is a problem often mentioned regarding TD management [3, 21, 22, 15]. For example, in their work about architectural TD (ATD), Verdeccia et al. found “*difficulties in communicating the presence of ATD to the stakeholders of a software product*” [5].

Better communication and the overview of TD can, in turn, impact how IT managers plan the project pipeline, i.e., the timing of the project portfolio’s various projects. This could provide a solution to the impediment mentioned by Besker et al. [23]: “*If the managers are not aware of the amount of software development time the developers waste because of TD, they are consequently not able to react and take appropriate action regarding the wasted time.*”

TD Causes. The initially named cause for TD incurrence is tight project deadlines, as described in the paper of Cunningham [1]. Other causes have been identified by different research papers, e.g., bad design decisions, unavailability of a key person, neglected technical improvements, lack of education, communication issues, or problems related to project planning [17, 24, 25, 4]. Nevertheless, tight deadlines are identified by many research works as the most pressing cause [24, 26, 4, 5, 15, 21].

An example of the problem of tight deadlines is the German government’s VAT cut in the summer of 2020 as part of its response to the Corona crisis. The corresponding law was enacted only two days before it came into force. To reflect this change in the software systems, adjustments to these systems were necessary for many German IT units. These units were, therefore, subject to a tight schedule.

In many cases, this constant time pressure leads to a phenomenon in developer behavior. In preemptive obedience, the developers search for the fastest and easiest solution assuming a tight schedule, even when there is no time pressure. To step back and think about or discuss different solution options is not common practice. In a paper about cognitive biases and their antecedents, Borowa et al. call

this phenomenon communicational antecedents, i.e., “*constructive criticism and challenging the ideas underlying the decisions of others is not standard*” [22]. This may lead to the anchoring bias, i.e., choosing the first available option. In turn, this bias causes decisions to incur TDs that are not made intentionally. Regarding this topic, Besker et al. found that “*practitioners rarely base their decisions on pre-defined procedures or guidelines*” [20], and Lenarduzzi et al. state that “*decisions related to TD issues were often informal and ad-hoc*” [27].

There may be frequent situations where the sub-optimal but faster to implement solution must be chosen due to project deadlines. However, it should be a goal to make it the whole team’s conscious and intentional decision to choose the optimal or sub-optimal solution.

In this paper, we present and evaluate the TAP framework for **T**echnical debt **A**ware **P**roject management which was developed by practitioners. The TAP framework divides all tasks that are not visible to the customer into four categories. The developers create corresponding backlog tickets. One of these categories is the so-called TD ticket category. These tickets comprise only intentional TD and relate these tickets to the project incurring the TD. The repayment of these TD items is performed as part of the project after meeting the set deadline. Other parts of the TAP framework are a continuous time contingent for maintenance and maintenance projects similar to what McConnell described [6]. Maintenance in the TAP framework includes repayment of unintentional TD, adaption to new technologies, optimization of quality attributes. This way of handling TD is a new contribution to the State of the Art.

To the best of the authors’ notice, no studies integrate TD management into the project management process focusing on the time limit and including a solution to prevent TD despite tight deadlines that is feasible in an industrial environment. By TD tickets and their effects on decision-making and the project pipeline, this framework presents such a solution.

The contribution of this paper is the presentation and evaluation of the TAP framework and especially the TD tickets. The evaluation consists of the analysis of ticket statistics and two surveys addressing team members and IT managers, respectively.

The evaluation will show the TAP framework’s benefits as perceived by the IT unit using it:

- Communication between different stakeholders is optimized.
- Overall awareness for TD is raised.
- Decisions on TD incurrence are made consciously and intentionally.
- TD incurred by unintentional and unconscious decisions can be prevented.
- TD incurred by intentional and conscious decisions due to tight deadlines are repaid timely.

Additionally, a survey of the IT managers indicates that they can plan the project pipeline more realistically and make the customer understand delays due to TD repayment and prevention.

In the following subsection, we will give an overview of the related work. Subsequently, we will motivate and introduce the TAP framework in Section 3. Section 4 will present the case study design, including research questions (RQs) regarding the TAP framework’s effects, benefits, and drawbacks. The evaluation comprises ticket statistics and results of two surveys conducted in the observed IT unit and a comparison unit. The evaluation’s results will be presented in Section 5. In Section 6, we will discuss the results, answer the RQs, and describe the threats to validity. The paper ends with the conclusion and future work in Section 7.

2. RELATED WORK

In this section, we present the related work to our paper. For this purpose, we follow the paper’s title and describe the state of the art of TD prevention, TD awareness, TD in project management, and case studies for TD management frameworks or methods.

2.1. TD Prevention

TD Prevention is often mentioned in research but mostly as a **problem statement** [28, 8, 29]. In 2015, Li et al. [25] found TD prevention to be understudied. More research on how to prevent TD beyond raising TD awareness has been done since.

Many papers researching **prevention strategies** focus on inadvertent code debt and tools helping to find these debts, e.g., code smells, before incurrence [11, 10, 8]. Fewer papers focus on the prevention of TD on the design and architecture level, which are usually incurred during decision-making.

Morgenthaler et al. provided insights to the TD management of build debts at Google. They summarized their prevention strategies in three topics “Automation”, “Make it easy to do the right thing”, and “Make it hard to do the wrong thing” [12]. Yli-Huumo et al. found that minimum viable products can help prevent TD by focusing TD prevention on areas that are important for the customer and throwing away parts that have shown no real business value [30]. Preventing documentation debt during requirements engineering was the research focus of Charalampidou et al. They integrated requirements specifications into the IDE (integrated development environment) [31]. Lenarduzzi et al. suggested optimizing requirements and receiving early feedback from customers. These approaches improved the effort estimation and enhanced the quality by optimizing tests and documentation [27]. Yang et al. suggested for COTS (commercial off the shelf) - intensive systems to train the COTS assessment team and explore contracting options with COTS vendors

[32]. Another way to prevent TD is to reuse code which was mentioned in the work of Feitosa et al. [33]. In their comparison of encouraging, rewarding, forcing, and penalizing strategies for TD reduction, Besker et al. found some companies were using educational sessions on how to avoid TD [13]. Vogel-Heuser et al. advised using guidelines to prevent TD in embedded systems [29]. Borowa et al. presented some mechanisms to prevent cognitive biases that may lead to TD incurrence, such as “challenging all decisions”, “explicitly gathering information about alternatives”, “documenting and passing on knowledge”, “explicitly registering all accounts of TD” and “defining and recording who is responsible” [22].

The InsignTD project¹ is an international survey study. The survey participants often stated that their TD could have been prevented [4]. Various papers of this study uncovered TD causes and prevention strategies used by practitioners. The most commonly found prevention strategies are “well-defined scope/requirements”, “code evaluation/standardization”, “following well-defined project processes”, “well-defined architecture/design”, “TD awareness/management”, “adoption of good practices”, “improving tests/coverage”, “good communication”, and “reviews” [4, 28, 9, 14].

All these research results are valid aspects of TD prevention. However, compared to the **TAP framework** these aspects are mostly vague, i.e., details and information on how to achieve these aspects are missing. Furthermore, they do not provide a solution to the problem of tight deadlines.

2.2. TD Awareness

In 2012, Kruchten et al. already mentioned the importance of TD awareness: “*The first step is awareness: identifying debt and its causes*” [17]. In a survey with more than 1,800 participants, Ernst et al. found that 79% of the participants strongly agree that “*Lack of awareness of TD is a problem*” [26].

As with prevention, there are many papers mentioning TD awareness in the form of a **problem statement** [12, 34, 10, 29].

Papers on **how to raise awareness** are less common. Many research papers found that visualization helps to raise awareness for TD. An obvious way to visualize TD is to create a list of all known TD items, which is also part of the TAP framework. For example, Kruchten et al. provided a color scheme for a backlog including colors for new features, architectural features, defects, and TD [17]. Kaiser et al. presented the Code Christmas Tree, which is a visual presentation of the code coverage of a system. This visualization was hung in the hallway and resulted in discussions. By this, the visualization raised the awareness of different stakeholders [35]. Gupta et al. also found that so-called information radiators, i.e., visualizations that are

¹<http://www.td-survey.com/>

displayed where people can see it frequently, were a good way to raise awareness [36]. Baysal et al. as well as Treude et al. examined the positive effects dashboards have on developers' awareness of their projects' issues [37, 38]. Guo et al. presented a TD management approach including a list of TD, which increased the awareness of TD. In this case study, the project manager stressed the TD list's positive effect [19]. Eliasson et al. proposed a method to visualize dependencies in the automotive domain to make TD visible. They found their method to be useful for stakeholders to be aware of architectural TD [18]. Martini et al. proposed a method called AnaConDebt to identify whether and when Architecture TD should be refactored. They found this method is useful to raise awareness "*about the importance and the urgency of proactively refactoring ATD*" [39]. Baars et al. used a gamification approach to make developers aware of harmful patterns in their codebase by turning the patterns into monsters in Minecraft² [40]. Al Mamun et al. researched TD in self-driving miniature cars. They focused on code debt and proposed that tools detecting rules violations would also increase awareness [41].

Another way to raise awareness is to optimize communication, e.g., workshops and training sessions. The communication aspect is important for the TAP framework, too. Shrama et al. found the awareness of the refactoring's benefits was lacking. They reported on the strategy to conduct workshops on the refactoring's benefits and refactoring tools [34]. Letouzey et al. also mentioned an awareness workshop for top managers to introduce TD and the SQUALE³ method to them [42]. Tonin et al. [43] researched the effects of TD awareness by conducting a classroom study. The students were informed about the concept of TD and should use TD boards during a project. The effects of this were a changed attitude, more discussions, and more conscious incurrence of TD.

In addition to the presented ways to raise TD awareness, the **TAP framework** raises the awareness by assigning responsibilities for TD management which is an aspect not found in research to the best of the authors' notice.

2.3. TD management and project management

A project is characterized by a beginning and an end and thus has a deadline. This deadline is an often mentioned cause for TD, as shown in Section 1. One often mentioned **problem** for managing and preventing TD is related to project management [19, 4, 9, 14]. "*Lack of process integration*" is an obstacle mentioned in the work of Guo et al. Their case study provided some insights into the obstacles of adapting research methods to a practical environment [44]. Martini et al. found that the "*split of budget in project budget and maintenance budget boosts the accumulation of debt*" [24].

To **solve these problems** Ramasubbu et al. integrated TD management into quality management steps [45]. They created a TD register to store TD items in relation to their requirements and resulting defects. Furthermore, they considered design moves to implement requirements or repay TD and the risks associated with them. They conducted a cost-benefit analysis on this data and proposed a way to control TD. While their approach is similar to the TAP framework, their focus was on companies that already use quality assurance processes, and their approach did not focus on TD prevention. Other research papers address the management of a TD backlog, e.g. [46, 17, 47].

In contrast to the **TAP framework**, these papers do not focus on projects and the resulting timeline problems. The TAP framework tackles the deadline problem by integrating TD management into project management, i.e., integrating the repayment of a project's TD as part of the project itself.

2.4. Case Studies on TD Management

Regarding TD management, Besker et al. found that it "*promotes a culture within organizations in which developers are supported and appreciated for managing their TD*" and that it "*leads to a virtuous cycle where the right culture and TD prevention mechanisms reinforce each other*" [48].

This paper presents a comparative case study of a TD management framework developed by practitioners. To the best of the authors' notice, no other case studies compare an IT unit using a framework or method for managing TD with an IT unit not using a TD management strategy. For this reason, we compare our work to other case studies on **TD management frameworks and methods** but focus on confirmatory case studies and action researches.

In an action research, Yli-Huomo et al. developed a process for identification, documentation, and prioritization of TD. They focused on uncovering the risks and benefits related to these TD activities [49]. Guo et al. focused their case study on uncovering the costs of their TD management approach, which centers around a TD list and comprises five steps for measuring and prioritizing TD items. During the implementation of this approach in practice, they had to face various obstacles, which they presented in another paper [44]. Finally, they applied their approach not directly in practice but simulated it for past releases on actual company data. The Tracy framework for business-driven TD prioritization was developed by Reboucas de Almeida et al. They proposed this framework to prioritize TD repayment activities not only by technical necessities but also by the business value of the configuration item that holds the TD item [50]. The work of Ramasubbu et al. is similar to the TAP framework and was also validated by an industrial case study [45]. We already introduced it in the previous paragraph. Malakuti et al. described the TD management approach of a specific project and obstacles while introducing TD concepts to a

²<https://www.minecraft.net/>

³<http://www.squale.org/>

company, but they were still missing a working solution [51].

All these TD management strategies focus on the general management of TD without a relation to a specific time frame. In contrast, the central aspect of the **TAP framework**’s TD tickets is the integration of TD management into project management which includes the deadline aspect.

3. TAP FRAMEWORK

3.1. Motivation

The TAP framework was developed and established in an IT unit at the beginning of 2018. It was utilized ever since and is the subject of this case study. The focus of the IT unit that developed this framework is on systems that support the marketing of advertisements in print, online and mobile media, a volatile and competitive market.

Accordingly, tight timelines and frequently changing systems are the main cause for the TD of this IT unit. The tight timelines often meant that “*well-structured project planning*” or “*adopting good development practices*” as proposed by Freire et al. [28] is not sufficient.

In this IT unit, TD accumulation led to increasing cycle times, unnecessary errors in the systems, and developers’ discontent. The IT management was surprised by these problems because they were not aware of having incurred debts in any form. Ultimately, however, they recognized these problems and gave sufficient priority to dealing with TD. The IT unit including the management was willing to change its processes.

The main goals for this change were pre-defined by the IT unit’s manager: First, TD must be prevented in any case if this is possible without affecting the deadlines. Second, whenever a set deadline leads to TD, this TD shall be repaid timely. Third, the IT managers should have the ability to track the TD in their systems.

On this basis, a framework to help manage TD was established. Two factors of the TAP framework, in particular, were intended to support TD prevention: the overall raised awareness for TD incurrence and the integration of TD management into the project management process. A third factor, the overview of TD resulting from the framework adoption, helped the management track TD. The resulting TAP framework for TD management is presented below.

3.2. Development

The framework was developed in a working group of two managers and two solution architects of the IT unit. The development spanned five steps: (I) identification of the tasks that hinder the development of functional requirements, (II) identification of management and prevention processes for these tasks, i.e., ticket categories, (III) assignment of tasks to ticket categories, (IV) development and documentation of the overall framework, (V) establishment of the new processes in the IT unit.

Eleven hindering tasks were identified: (1) bug fixing, (2) regularly refactoring of code that may be in a bad condition due to causes that are not specified in detail, (3) workarounds which are made under the pressure of a deadline, (4) missing documentation or tests mostly due to time pressure, (5) implementation that stay incomplete because of changing business decisions, (6) temporary switches in the program for the distinction of new and old program parts, (7) adoption of new architecture approaches, (8) adapting new infrastructure like e.g. version upgrades, (9) innovation like proofs of concepts (POC) for new technologies, (10) implementation and refinement of technical monitoring tools, (11) performance optimization.

Four processing types were developed to handle these tasks, resulting in the corresponding ticket categories. In this framework, all other tasks are functional requirements including bugs and performance issues that are visible to the user. The assignment of the hindering tasks to these categories can be seen in Table 1.

Task	Scope	Category
(1) Bug fixing	not visible/ visible	MT FRT
(2) Refactoring	small big	MT MP
(3) Workaround		TDT
(4) Documentation/Test		TDT
(5) Incomplete implementation		TDT
(6) Temporary switch		DT
(7) Architecture change	small big	MT MP
(8) Infrastructure		MP
(9) POCs	small big	MT MP
(10) Monitoring		MT
(11) Performance	not visible visible	MT FRT

Table 1: Hindering tasks and their assigned ticket categories (MT - Maintenance Ticket, MP - Maintenance Project, TDT - TD Tickets, DT - Deconstruction Tickets, FRT - Functional Requirement Ticket)

3.3. Ticket Categories

The TAP framework results in four ticket categories for handling different types of TD-related tasks. It is essential to notice that the distinction of these categories is not because of the causes or theoretical classification schemes. The distinction refers to the way the tickets are to be processed. A fifth ticket category handles all functional requirements. The latter category is not presented in detail as it is not in the scope of this framework.

All tasks are recorded as tickets in a comprehensive project backlog and tagged and handled depending on their category. The framework includes guidelines for the category-specific recording and processing of these tickets. The person who has the greatest interest in completing the

tasks of a category is the one responsible for this category. This person may be an architect or a business analyst. Table 2 shows an overview of the ticket categories, the responsible person, and the processing conditions.

3.3.1. Maintenance

In the TAP framework, maintenance tickets are technically driven tickets that have no direct impact on the user’s perception of the system⁴. In particular, adapting systems technically, enhancing quality attributes, and repaying unintentional TD are parts of this category.

Recording: Every maintenance ticket must contain one sentence describing the ticket’s impact. The description must be understandable to business analysts and managers.

Processing: Ten percent of every sprint’s planned capacity is invested in maintenance tickets according to the architect’s decision.

Goal: The goal of the maintenance tickets is to repay the unintentionally incurred TD continually and to allocate time for other maintenance tasks, e.g., technical adaption.

Example: Examples of this category are the upgrade of a third-party library or the refactoring of poorly structured code.

3.3.2. Maintenance Project

All tasks belonging to the maintenance category but requiring more than five days of development time are treated as maintenance projects. This is based on the practice applied for functional requirements and business projects.

Recording: The architects prioritize maintenance projects and create a pipeline of these projects.

Processing: This pipeline is integrated into the overall project pipeline as defined and prioritized by the IT managers.

Goal: Maintenance projects aim to allocate enough time to perform maintenance tasks with a long duration or maintenance tasks requiring more organization, e.g., when more than one team is involved.

Example: An example is a version upgrade of a central database used by more than one application. Another example is the change of the system’s architecture, e.g., from a SOA to a microservice architecture.

3.3.3. Technical Debt

These so-called TD tickets only comprise intentional TD. These tickets describe tasks necessary to improve internal software quality during the implementation of a functional requirement, e.g., time for clean code, good design, or to comply with a specified architecture. These

tasks are not mandatory to implement the required functionality and can be implemented after a given deadline.

The team often discusses two or more solution options, one of which follows the standards and the specified architecture, and the others more or less deviate from the optimal solution. The latter, however, may be faster to implement. Hence, this category only refers to conscious decisions to incur TD, i.e., intentional TD.

This idea of TD follows the original introduction of TD by Cunningham [1].

Recording: In such cases, the team consciously decides to record two tickets: The functional requirement ticket describes the sub-optimal solution implemented before the deadline. With this ticket, the team incurs TD. The TD ticket describes the optimal solution and the tasks to repay the incurred TD. TD tickets are usually identified during estimation meetings when the team discusses the details of a functional requirement and estimates the effort for this requirement.

Processing: The uniqueness of the TAP framework is that the repayment of these intentionally incurred TD items is part of the project plan. No project can be finished before the TD tickets that have been incurred during this project are processed. The project plan does not end after reaching the deadline and keeping to the tight schedule. Instead, the development team and architects prioritize the TD tickets of this project in a prioritization meeting. The prioritization is discussed and decided by a majority vote and does not follow a strict procedure. In a subsequent project phase, the TD repayment phase, the developers process the project’s TD tickets alternately with the project’s remaining requirement tickets, as shown in Figure 1. Thus, the responsibility for the TD repayment is transferred to project management because to finish their project is in the project manager’s interest. In this way, the TAP framework integrates the management of intentional TD into the project management process.

Goal: First, evaluating different solution options in advance raises the overall awareness only to incur TD intentionally. Consequently, unintentionally incurred TD is prevented. Second, making the project managers responsible for the TD accumulated during their project decreases their willingness to incur TD. In other words, this creates a feedback loop, and both the team and the project manager change their behavior towards TD incurrence. Third, project managers and IT managers get an overview of accumulating TD of a particular project while still running. The IT managers can intervene or adjust the project plan and the following projects early on. Finally, by this approach, intentionally incurred TD is repaid timely.

Example: For the example from Section 1, this may mean that hard-coded VAT rates are adjusted in the code rather than refactored to a flexible solution to meet the schedule. As a result of this decision, the developers record the corresponding TD ticket to introduce a centralized tax variable. This TD will be repaid in an orderly manner in the “TD repayment phase” of the “VAT adjustment”

⁴We use the term “maintenance” as it is often used in agile organizations. However, in research, standardization, and sometimes in classical organizations, the implementation of new or changing functional requirements is also referred to as adaptive maintenance.

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

Table 2: Ticket categories, responsible persons, and time

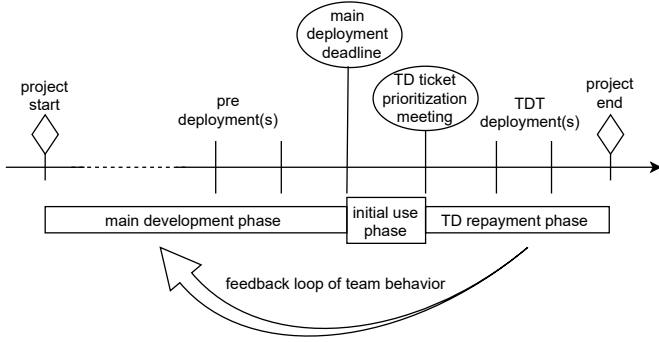


Figure 1: Project plan with included TD repayment phase

project and after the law comes into force. In contrast, discussion of the various solution options may show that the low effort makes centralization the better option, and TD are prevented from the beginning.

3.3.4. Deconstruction

A ticket of the deconstruction category is a special TD ticket that cannot be processed during the project. After implementing a new solution, the legacy code sometimes needs to be kept parallel for the time being. These tickets then comprise the deconstruction of this legacy code when it is no longer needed. They can also comprise the deletion of deprecated code parts after all consumers of these parts are detached. Therefore, this category only comprises intentional TD that cannot be repaid immediately, i.e., during the TD repayment phase of its project.

Recording: These tickets must contain additional information about when the deconstruction can take place.

Processing: The deconstruction ticket must be repaid as soon as possible as part of the contingent for functional requirements.

Goal: The primary purpose of these tickets is to avoid code cluttering and improve code comprehension.

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

3.4. Scope

The initial scope of this framework is the IT unit that developed the framework. Nevertheless, other units and companies may adopt the framework or at least relevant parts of it. Relevant parts are, in particular, the distinction in processing intentional TD as TD tickets and unintentional TD as part of maintenance tickets, the immediate recording of intentional TD, and the processing of intentional TD as part of the project that incurred them.

We think this framework will particularly help units that are subject to tremendous time pressure. The framework was used in an organizational setting of agile project management, including agile processes and project management elements, including project deadlines. Regardless, we expect the framework to help in all organizational environments where a timeline is relevant, i.e., project or release deadlines.

The framework may not be suitable for agile product development without timeline constraints. Additionally, the problem caused by accumulated TD must be large enough to make the initial effort to organize the backlog worthwhile.

4. CASE STUDY DESIGN

The goal of this research was the presentation and evaluation of the TAP framework. The framework's presentation took place in Section 3. The evaluation was based on the case study design, according to Runeson [52].

4.1. Research Questions

As described in Section 1, the two main problems concerning TD are preventing TD and dealing with TD despite having a tight schedule and fixed deadlines. The TAP framework provides a possible solution found and developed in practice for these problems. With subsequent research questions (RQ), this paper provided an evaluation of the TAP framework. We evaluated especially the framework's feasibility and effectiveness regarding the problems mentioned above.

(RQ 1) Framework Application

(RQ 1.1) Do practitioners find the TAP framework reasonable?

(RQ 1.2) Are the processes of the TAP framework feasible in practice?

This is the basis for all other evaluations as it is not reasonable to analyze the framework's impact further should it not be feasible in practice. We used a team survey of the observed IT unit's members and ticket statistics to evaluate these questions.

(RQ 2) TAP Framework's perceived Effects

(RQ 2.1) Is the use of the TAP framework associated with a raised awareness for the incurrence of TD?

(RQ 2.2) Is the use of the TAP framework associated with a more conscious incurrence of TD items?

(RQ 2.3) Is the use of the TAP framework associated with a better overview of TD items?

(RQ 2.4) Is the use of the TAP framework associated with a timelier repayment of TD items?

These aspects are the TAP framework's goals, as presented in Section 3. The evaluation of these questions shall show whether these goals were reached. To evaluate these questions, we used ticket statistics and surveyed the members of the observed and a comparison unit.

(RQ 3) TAP Framework's perceived Benefits and Drawbacks

(RQ 3.1) Do practitioners observe that TD can be prevented by the adoption of the TAP framework?

(RQ 3.2) Are there other benefits or drawbacks arising from the adoption of the TAP framework?

(RQ 3.3) Do practitioners find these benefits justify the additional effort?

The purpose of these questions is to gather the TAP framework's impact. The main goal of TD prevention is evaluated, as well as secondary benefits concerning team discussion, decision-making process, and generating an overview of TD and maintenance tasks. To answer these RQs, we used descriptive statistics, and correlations between the survey variables for effectiveness and benefits to support the descriptive statistics. Furthermore, we asked for the justification of the additional effort in relation to the benefits to substantiate the framework's feasibility.

(RQ 4) Management Perspective

(RQ 4.1) Do the managers have an overview of their IT systems' TD?

(RQ 4.2) What are the benefits of the TD overview from a management perspective?

The success of the TAP framework depends largely on the management's support as the additional effort must be

approved by them. Therefore, we evaluated the effect the framework and especially the generated TD overview has on the management.

4.2. Case and Units of Analysis

Following the definitions of Runeson [52], we conducted a comparative case study with two units of analysis - the observed IT unit that developed and adopted the TAP framework and the comparison unit that did not. The composition of the observed unit and the comparison unit are presented in Table 3. The comparison unit is slightly smaller but is led by the same unit manager and follows a similar organizational structure.

The focus of this case study was the evaluation of the framework's effects on the observed IT unit. The case study protocol, as well as all data collection and evaluation details and the data itself can be accessed in the additional material⁵.

Context. The IT units are part of a German publishing house with more than 9000 employees worldwide. This is an established company publishing content and advertisements in magazines, websites, and mobile apps. Both IT units follow a Scrum approach and develop and maintain software for internal use. They are both dealing with a substantial amount of legacy code.

Unit of analysis. The observed unit develops and evolves systems responsible for the marketing of advertisements. The main competitors are Google and Facebook. This leads to often changing systems due to new advertising formats and the merging of companies to get a bigger market share. Additionally, the changes are usually time-sensitive to be first in the market or adhere to the business year when merging companies.

The unit develops about 5-10 medium-large systems and the interfaces between these systems. The development tasks are organized in agile-managed projects. The unit comprises three organizational teams with two cross-cutting Scrum teams. The three organizational teams comprise:

- **Business Analysis:** The members are responsible for the requirements engineering and project management.
- **Development:** The members focus solely on the systems' development.
- **Operations and Support:** The members are responsible for customer communication, quality assurance, and basic server support.

Each Scrum team is responsible for specific tasks depending on the required technology's expertise. The business analysts are acting as product owners in the Scrum processes.

⁵<https://doi.org/10.5281/zenodo.5788222>

The unit of analysis was not chosen but given, as this IT unit developed the framework for itself.

Comparison Unit of Analysis. The comparison unit develops and evolves systems responsible for internal company tasks, like support for human resources, janitor services, company intranet, or canteen.

The unit develops about 20 small- to medium-size systems and their interfaces. Like the observed unit, the comparison unit uses agile projects for their development and has to reach project deadlines. On the one hand, in some of their projects, the time pressure is less stringent than in the observed unit. On the other hand, some of their projects have to adhere to legal requirements, including hard-set deadlines.

Furthermore, the IT management particularly encourages this unit to explore new technologies, leading to additional risk of TD and legacy code. The comparison unit consists of exactly one team, which is both the organizational team and the Scrum team. This team has one business analyst who is also the architect and the product owner in the Scrum process.

The comparison unit of analysis was chosen due to their availability as the manager of both units supports this study. Furthermore, both units follow a similar organizational approach, i.e., agile-managed projects.

4.3. Data Collection

To answer RQ 1.2 and RQ 2.4, we evaluated the recording and processing statistics for TD and maintenance tickets. RQs 1 to 3 were answered by the results of the team survey. Finally, a follow-up survey investigated the TAP framework’s effects on the IT managers, which answered RQs 4. Both surveys targeted participants from the observed unit and the comparison unit.

4.3.1. Ticket Statistics

We collected data from all project backlogs of the observed unit and identified the tickets under consideration by tags and epic affiliation. All tickets in the backlog were labeled according to their category at the time of data collection. 524 maintenance tickets and 141 TD tickets were identified.

In consultation with the teams’ architects, we validated the raw data and removed old or invalid tickets. Furthermore, we narrowed down the tickets to the period from January 2018 to March 2020. 236 maintenance tickets and 102 TD tickets remained for the evaluation.

We did not evaluate the statistics for maintenance projects and deconstruction tickets due to insufficient data, e.g., only ten deconstruction tickets exist between 2018 and 2021.

4.3.2. Team Survey

We chose the method of a survey to benefit from the input of all team members willing to share their experiences. For RQ’s 1 to 3, we asked corresponding questions in this questionnaire.

	observed unit		comparison unit	
	members	particip.	members	particip.
manager	2	n/a	2	n/a
architect	2	n/a	1 ^a	n/a
business analyst	8	6	1 ^a	1
developer	15	9	5	4
operations	5	2	0	0
sum	32	17	8	5
response rate		53%		62%

^aarchitect and business analyst are the same person

Table 3: Composition of units and team survey participants

Survey Participants. We asked all members of the observed and comparison IT unit to fill out the questionnaire. The comparison unit’s results allowed us to validate the descriptive statistics. The participants were asked which team they belong to, but due to works council regulations, they were not asked which role (e.g., architect, manager) they inhabit. The response rate to the survey is shown in Table 3.

Questionnaire Construction. To avoid misunderstandings, the questionnaire started with short information about the term TD. For statistical reasons, background information was queried, e.g., the participant’s team membership. The questionnaire was divided into two main parts: (I) the framework’s assessment and (II) the framework’s effects and benefits.

Part (I) showed the framework’s practical feasibility and provided answers to RQ 1.1 and RQ 1.2. For every ticket category, the same set of questions regarding the reasonableness and feasibility was asked. Each part started with short information about the ticket category. Part (I) was not filled out by the comparison unit as these participants did not use the framework.

Part (II) comprised five subsections with questions regarding:

- TD awareness (RQ 2.1)
- comparison of optimal and sub-optimal solutions (RQ 2.2)
- overview of TD (RQ 2.3)
- observed benefits of the comparison and overview (RQ 2.4, RQ 3.1 and RQ 3.2)
- justification of the additional effort (RQ 3.3)

All parts included a set of assertions the participants were asked to validate using the following Likert scale[53]: applies - rather applies - rather does not apply - does not apply - cannot answer. The *cannot answer* option was given as some questions could not be answered by all members of the unit. The participants were explicitly asked only to use this option in these cases.

Finally, two more open questions at the end allowed the participants to point out good parts of the TAP framework and parts needing improvement.

Two developers familiar with the TAP framework filled out a first questionnaire as a pilot test. Based on this, we reduced and focused the questions. One IT manager gave additional feedback on the optimized questionnaire’s construction, which we incorporated.

4.3.3. Follow-up Management Survey

During a first presentation of the evaluation results, the management mentioned that the framework provides a valuable overview of the TD and maintenance tasks. This perception contradicts what most of the survey participants assumed of their management (see Figure 4(b)). Since the advantages of TD management for IT managers are not often researched, we investigated this in more detail.

Survey Participants. The survey participants were only the three involved line managers: the unit manager of both units, the team manager of two teams of the observed unit, and the team manager of the comparison unit. All managers have personnel responsibilities for the teams involved and have a good and broad overview of the business goals. Due to the small sample size, we could not generalize the results. However, the evaluation provided qualitative input exploitable in further research.

Questionnaire Construction. We asked the managers four open questions via email to gather qualitative data. The questions were related to

- the overview they have of their system’s TD,
- the effects this has
 - on their project pipeline,
 - on TD repayment and prevention, and
 - on the communication with their customers.

4.4. Data Analysis

4.4.1. Ticket Statistics

To evaluate RQ 1.2 and RQ 2.4, we presented the number of tickets in terms of categories and time as descriptive statistics for the period from January 2018 to March 2020.

4.4.2. Team Survey

For the questionnaire’s evaluation, we used the statistics software SPSS⁶ for closed questions and the qualitative research software MaxQDA⁷ for open questions.

First, we created descriptive statistics of all closed questions for an exploratory data analysis. For simplification and analysis purposes, we dichotomized the answers for the questions of part (II). This means we summarized the answers *applies* and *rather applies* as well as *rather does not apply* and *does not apply*. We present all values as a percentage to align the output of the observed unit and

the smaller comparison unit. For the sake of clarity, we only show the *applies* answers in the following figures.

Second, we developed hypotheses for RQ 2 based on this exploration and evaluated the significance of the hypotheses. We used the Mann-Whitney U-Test [54] because it can be applied to not normally distributed data on an ordinal scale. We present only significant findings.

Third, we analyzed correlations between survey participants who incur TD consciously and the benefits they assess. We used the ϕ -coefficient of Pearson’s χ^2 -correlation [55, 56] for dichotomous variables and assumed a significant correlation from a significance level of 0.05, i.e., 5%. The effect size is interpreted as medium if it is higher than at 0.3 and as strong if higher than 0.6.

Last, two researchers used open coding to evaluate the open questions.

4.4.3. Follow-up Management Survey

Two researchers evaluated the management questionnaire’s results using open coding with MaxQDA. Due to the limited number of participants, we did not count codes. Instead, we provided some qualitative insights of the line managers of the observed and comparison unit.

5. RESULTS

5.1. Ticket Statistics

The timelines in Figures 2(a) and 2(b) show the development of ticket counts over time to answer RQ 1.2. (feasibility) and RQ 2.4 (timely repayment). The maintenance tickets were created and processed continually. The timeline for TD tickets shows a peak in incurred TD in spring 2019 due to a project with a tight deadline. No maintenance or TD tickets were processed in March 2019. Accordingly, the TD tickets as well as the maintenance tickets show a processing peak after the reached deadline at the end of April 2019. Initially, only the TD tickets with the highest priority were processed. From July to November 2019, the TD tickets with a lower priority were processed, meaning that this project was still not completed (see Figure 1). Other projects that started in parallel caused this long duration because development capacities had to be divided between the projects.

5.2. Team Survey

5.2.1. Assessment of the TAP Framework

Most survey participants agreed that recording and processing all four types of tickets are generally reasonable (Figures 3(a) and 3(d)). Most of them also agreed that the framework’s procedures for recording and processing are reasonable (Figures 3(b) and 3(e)). When asked whether the framework worked as intended, the agreement decreased, but most of the survey participants still stated that the procedures work well and are helpful (Figures 3(c) and 3(f)). For RQ 1.1 and RQ 1.2, these results mean the TAP framework is reasonable and feasible in practice.

⁶<https://www.ibm.com/de/de/analytics/spss-statistics-software>

⁷<https://www.maxqda.de/>

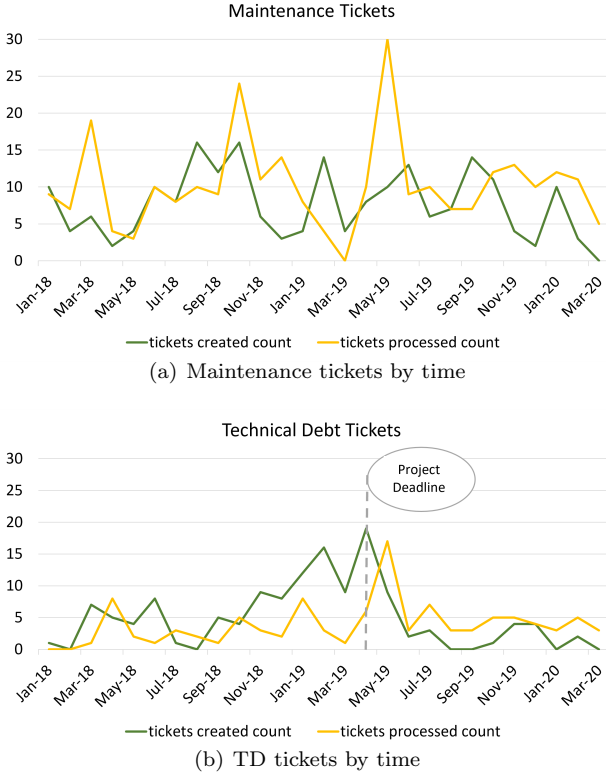


Figure 2: Ticket statistics with a project deadline at the end of April 2019

5.2.2. Perceived Effects of the TAP Framework

First, we asked survey participants whether they and their team typically recognize that they incurred TD items before or after their incurrence. The goal of this question was to identify whether the participants were aware of the incurrence to answer RQ 2.1. With more than 75.0% agreement, both units stated that they usually recognized it when TD items were incurred. The results are not significantly different.

Second, the participants were asked whether they compare different solution options and estimate respective implementation costs, principal, and interest rates for incurring TD. To answer RQ 2.2, we assumed a conscious and intentional decision to incur TD if the units compared different solution options. Regarding this question, an apparent difference between the units could be seen in Figure 4(a). 75.0% of the observed unit's participants agreed to compare sub-optimal and optimal (or more) solutions and to calculate implementation cost for this comparison. Only 20.0% of the participants agreed to compare options and estimate implementation costs for the comparison unit. Furthermore, many participants of the observed unit agreed that they estimate principal (46.7%) and interest costs (33.3%), while this was not the case for the comparison unit.

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

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

Table 4: Significance of H_R -hypothesis

esis (H_0):

$H_{R.1}$: The observed unit compares sub-optimal and optimal solutions more frequently than the comparison unit. ($H_{0.1}$: The observed unit does not show different behavior than the comparison unit in terms of comparing sub-optimal and optimal solutions.)

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

Table 4 shows the significance values of these hypotheses measured with the Mann-Whitney U-Test. The exact significance for $H_{R.2-4}$ hypotheses is below 0.05. Consequently, these hypotheses can be accepted as they are significant to the 5% significance level. This means the possibility these hypotheses are wrong is lower than 5%. The exact significance for $H_{R.1}$ hypothesis is 0.058 and a little higher than 5%. $H_{R.1}$ cannot be accepted. While these statistics give an additional indication on the difference between the units, we have to keep the low number of data points in mind.

In summary, this means the framework's adoption is associated with a more conscious comparison of different solutions options. In particular, the estimate of the respective costs of the solution options differs significantly between these two units.

Third, we asked whether the framework's adoption generated an overview of TD items to answer RQ 2.3. To evaluate this, we asked the participants which stakeholder has an overview of all TD. Figure 4(b) shows that in the observed IT unit, the architects had an overview of all TD (100% agreement). The customer was not expected to know the system's TD (11.8% agreement). In the case of the comparison unit, it was unclear which stakeholder had an overview of TD.

5.2.3. Perceived Benefits of the TAP Framework

To answer RQ 3.1 and 3.2, we asked the participants what benefits they expected (comparison unit) or assessed (observed unit) of the comparison of different solution options and, thus, the conscious incurrence of TD. Figure 4(c) shows that both units mostly agreed that this comparison could prevent TD. Especially the observed unit assessed that discussions were led more rational (82.4% agreement)

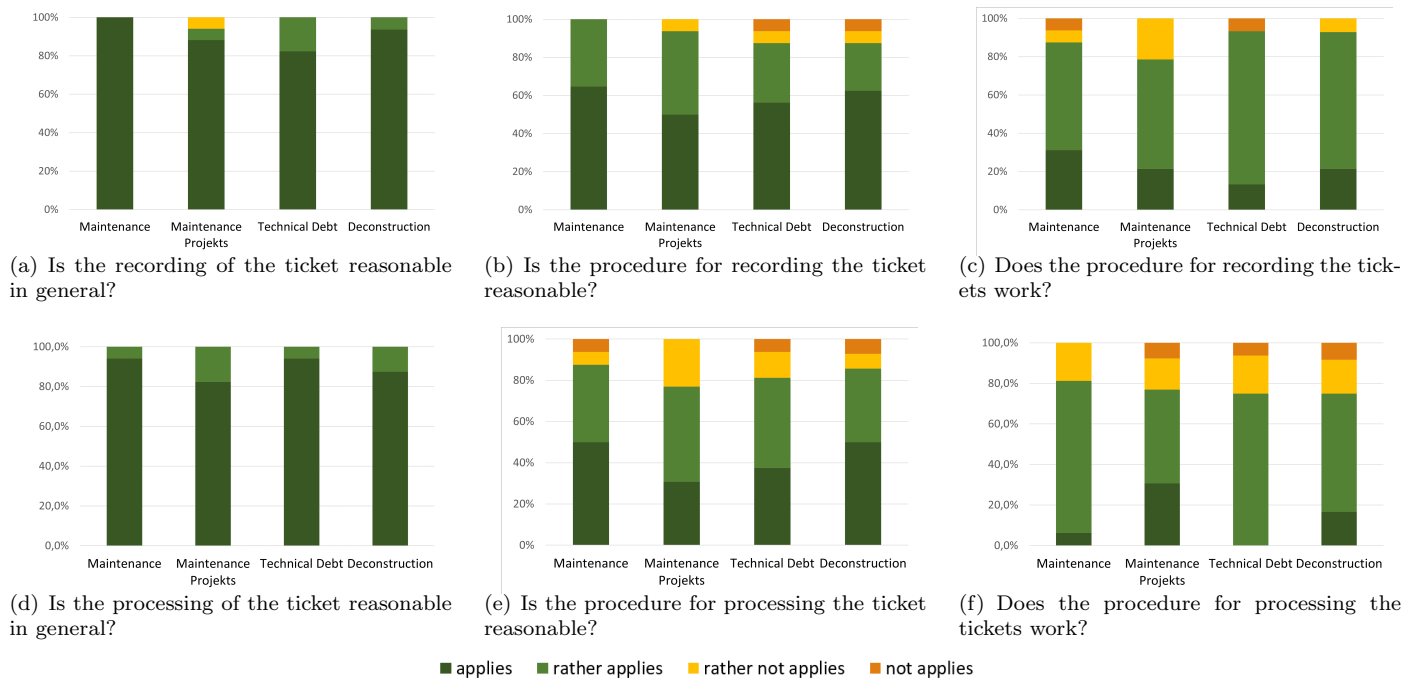


Figure 3: Appropriateness of the TAP Framework - Descriptive Statistics

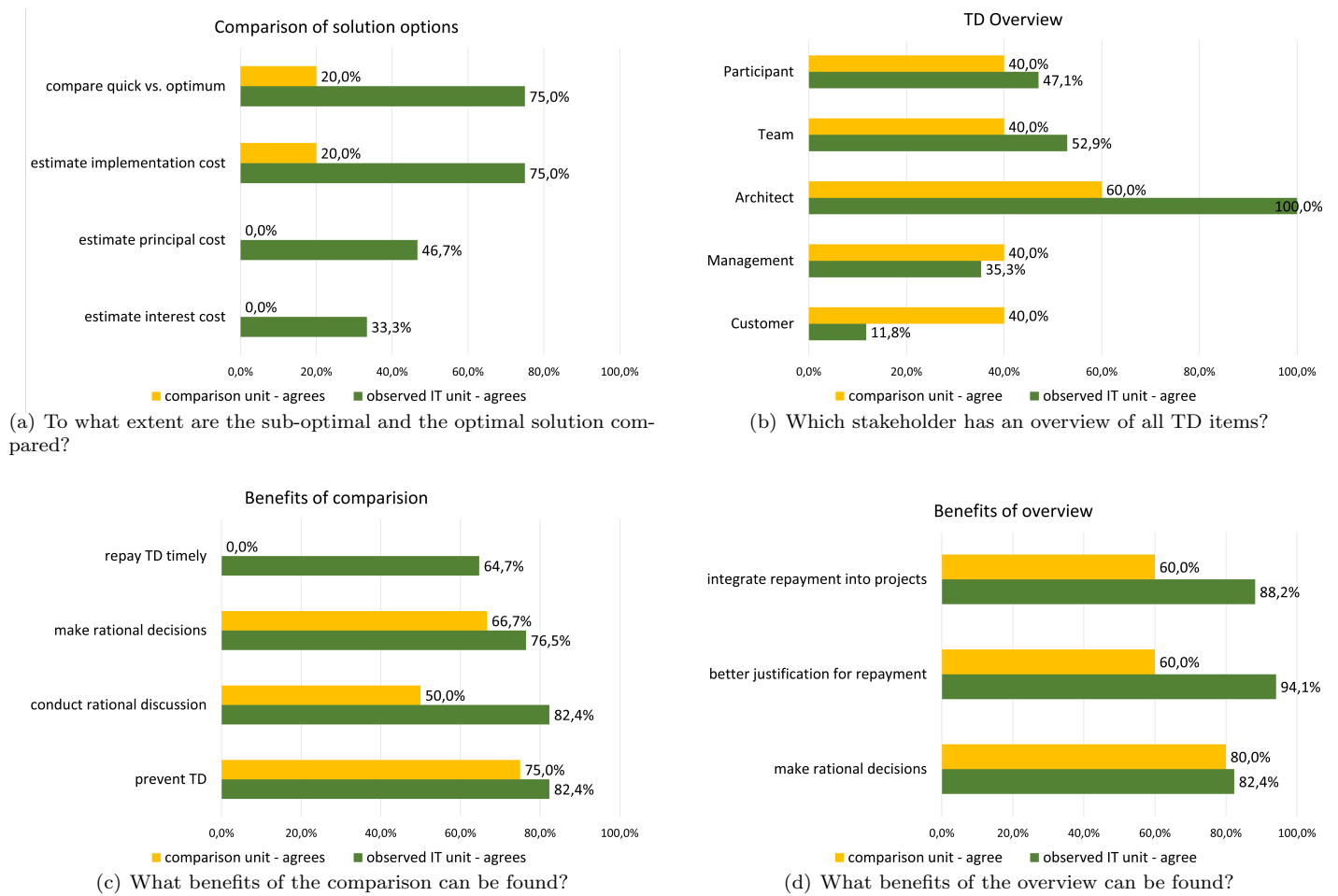


Figure 4: Usability, Effects, and Benefits of the TAP Framework - Descriptive Statistics

	TD prevention		rational discussions		rational decisions	
	ϕ -coeff	signif.	ϕ -coeff	signif.	ϕ -coeff	signif.
comparing solutions	0.419	0.061	0.303	0.176	0.623	0.007
estimating implementation costs	0.357	0.110	0.471	0.035	0.535	0.020
estimating principal costs	0.394	0.086	0.456	0.047	0.495	0.036
estimating interests costs	0.309	0.179	0.357	0.120	0.385	0.103

Table 5: Correlations of comparison and estimations vs. benefits
(Significant correlations are highlighted)

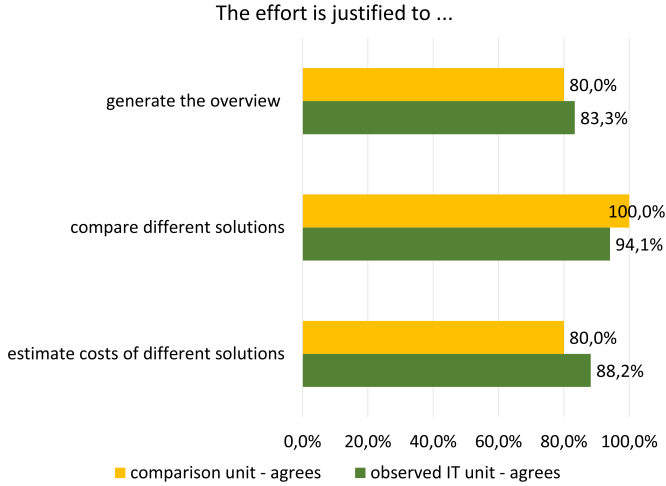


Figure 5: Do the survey participants think the extra effort for this is justified?

and more rational decisions were made (76.5% agreement). Timely repayment (RQ 2.3) of TD was not expected by the comparison unit but assessed by many participants of the observed unit (64.7% agreement). This difference may partly be due to a misunderstanding that the observed unit was unconsciously assessing the framework’s benefits and not just the comparison’s benefits. The units show no relevant differences in benefits, which means no hypotheses can be derived.

To answer RQ 3.2, the participants were asked what benefits they expected or observed from the generated overview of TD items. The observed unit’s participants noted that integrating the TD items into existing or upcoming projects is easier with an overview (88.2%). Furthermore, the overview gives a better justification for the repayment (94.1%) (see Figure 4(d)). Only 60% of the comparison unit expected the overview’s benefits, which is a noticeable smaller but still not significant amount. Both units expected or observed more rational decisions due to the generation of the overview.

Finally, to answer RQ 3.3, the participants were asked whether the benefits justified the effort for comparing and estimating the options and generating the TD overview. Both units mostly thought the effort was justified (>80.0% agreement). Again, the units show no apparent differences (see Figure 5).

5.2.4. Correlations between effects and benefits

As shown in the previous section, the main difference caused by the framework was that the observed team compared different solution options and estimated the costs for new TD tickets. We observed different benefits that may result from this comparison and estimations: more rational discussions, more rational decisions, and the prevention of TD. To support the observation and explore the possibility of a causal effect between the comparison and estimations and these benefits, we calculated the correlations between them. We also calculated the significance of these correlations and assumed a correlation if the significance was less than 0.05.

Table 5 shows significant correlations of medium strength between the perceived benefit of making more rational decisions and the observed effects of comparing solutions, estimating implementation costs, and estimating principal costs. The benefit of leading more rational discussions has significant correlations of medium strength to the estimation of implementation and principal costs.

Following our observations and the correlations, we can associate the comparison and estimations lead with more rational decisions. Furthermore, we associate the estimation of implementation and principal costs with more rational discussions. Both associations seem to be valid assumptions since they are also logically derivable.

The correlations do not support associations for TD prevention or the estimation of interest costs.

As we did not have a significant difference between the units related to the overview of TD, we did not calculate correlations for the overview’s perceived benefits.

5.2.5. Open Questions

Most comments were only made one time and are single opinions. Two points were mentioned three times each.

First, “TD is a topic” expresses that it was already helpful to talk about the topic of TD and not to ignore it. The code is related to the goal of raising TD awareness. One participant pointed out: “*I actually see the main advantage of conscious handling of the topic in the matter as such, because otherwise, it is a topic that is almost ignored in everyday development.*”

Second, “Sometimes TD is OK” shows that there were situations in which participants considered retaining TD reasonable, e.g., in legacy systems. Another participant

stated: “If a system is to be replaced, then I consider technical debts to be justifiable if they disappear after 1-2 years anyway”.

A third point was mentioned two times: “Evaluation is difficult and subjective”. This topic refers to the research topic of TD prioritization. The participants mentioned that it is sometimes hard to decide which TD should be repaid first and whether a particular item is TD at all, especially if the architectural rules seem to be open for interpretation. One participant phrased it as follows: “Even if the ‘teaching book’ says that the current implementation does not correspond to the architectural specifications, it still seems to be a personal opinion that decides on the fundamental correctness of the specification.” The other participant mentioned that TD is easy to define as a deviation from the norm or rules. However, it is “very difficult to evaluate this deviation and work out the importance of fixing this debt.”

One participant also mentioned that the “framework raises the understanding for each other”: “I’m thinking of greater domain understanding in development and greater technical understanding for business analysts.” Regarding the “more rational discussions”, one participant states: “If the process is consciously lived and ingrained in people’s minds, it reduces energy-sapping discussions about fixing technical debt.” Another participant summarizes these effects of the comparison: “The conscious handling of different possible solutions promotes cooperation, mutual understanding, and knowledge exchange in the team and thus generally contributes to the development of stable and accepted software.” Lastly, one participant acknowledges the “importance of TD prevention”: “The incurrence of debt should be reduced, not the process for working it off optimized.”

The following **drawbacks** were mentioned by the participants. One participant stated that the categories’ distinction and the separation of their responsibilities might lead to a separation of the team. “Although we keep saying that we are one team, we often stay separate and the different tickets in particular help make the differences between the team parts visible”. Another participant thought the framework “(slightly) contradicts the agile principle of providing a functioning solution with company value as quickly as possible. At the same time, however, it levels the stable ground so that we can continue to work on new requirements quickly and without production errors. The advantage outweighs”. One participant of the operations team finds that in their team TD is not a topic and that “This is a concept that takes place in the development cosmos.” He recommends that the operations team should be involved in the repayment of TD.

5.3. Follow-up Management Survey

In this survey, we asked the IT managers of the observed and comparison unit for their perspective on their units’ TD management. The unit manager responsible for

both units answered for each unit separately. Thus, we got four different responses from three participants:

- team manager of the observed unit (TMO), participant A
- team manager of the comparison unit (TMC), participant B
- unit manager of the observed unit (UMO), participant C
- unit manager of the comparison unit (UMC), participant C

Overview of TD. The TMO and UMO both stated they had an overview of their TD. Both managers relied on a weekly meeting with the architects to establish this overview. In these meetings, architects and IT managers reviewed the architecture, planned the evolution of the system and architecture, and discussed the TD and maintenance projects. The architects prepared the meeting by providing a prioritized pipeline of maintenance projects and an architectural evaluation of the TD items. The TMC and UMC did not have a specific mechanism to create an overview of their TD. Still, they remarked that they had a “holistic project pipeline that identifies technical debts as such and made it possible to plan corresponding topics in an integrative manner.” The unit manager, who experienced the strategies of both units, remarked regarding the comparison unit: “In terms of future operational responsibility, I would feel more comfortable if the technical debt in the project was managed continuously and transparently.” He stated his concern that “TD does not become clear until the handover of operations or even during ongoing operations.”

Project Pipeline. For the TMO and UMO, the overview had a direct impact on the planning of maintenance projects. It also helped to plan capacities and estimate project duration realistically. The UMO even mentioned the possibility that “new projects start later.” The TMC stated that TD “projects are given a further measurable criterion for their urgency.” The TMC and UMC further mentioned that it was still a problem to prioritize “between business requirements and technical necessity”, while the TMO or UMO did not mention this problem.

TD Repayment Activities. The TMO and UMO planned their TD repayment activities during their weekly meeting with the architects. During these meetings, it became apparent in which cases it was sensible just to accept the TD, e.g., when the system will be re-engineered soon. The UMO further mentioned the effect the visibility may have had for the teams: “The existing debts become visible, and in the end, developers and product owners have to decide whether further debts are justifiable or realistically repayable in the foreseeable future.” The TMC focused on TD awareness and implied that the overview led to enhanced awareness. He stated that “TD awareness leads to TD avoidance” and that TD awareness “supports the case for projects and measures to reduce this debt”.

Communication with Customers. The TMO and UMO said the pipeline has helped make effects visible, making it easier to communicate with customers. Project durations were better understood and accepted. The UMO mentioned that “*it also reflects the customers’ interest in a) being able to use the right features on time, but b) also to use a stable and future-proof system in the medium to the long term*”. He remarked that this also included communication with other stakeholders, e.g., higher management or users. The TMC saw the same benefits but pointed out that the understanding also depended on the customer’s technical know-how.

6. DISCUSSION

In this section, we will finally answer our RQs from Section 4.1 regarding the TAP framework developed by practitioners. We used different methods to answer our research questions. Table 6 gives an overview of which method and figure will be used to answer the respective questions.

To give a better overview of the results, we generated a variation of a cause-effect diagram (Figure 6) summarizing the effect, benefits, and drawbacks. The diagram is based on the qualitative and quantitative data found in this study. Moreover, we took logical deduction into account for some of the more detailed connections, e.g. the repayment phase leads to timely repayment. The literature review led to some connections that supports the qualitative data. In addition, we summarized the codes at a higher level to give a better overview. The following sections discuss these cause-effect mechanisms in detail and answer the RQs.

6.1. Application of the TAP Framework

RQ 1.1/RQ 1.2: *Do practitioners find the TAP framework reasonable? Are the processes of the TAP framework feasible in practice?*

Figures 3(a) to 3(f) shows little disagreement. Most team survey participants found the framework reasonable and thought it did work as intended. Only a few participants thought it must be optimized. The ticket statistics (see Section 5.1) indicate that the tickets were recorded and processed as intended. We conclude that the framework is feasible in practice and the following findings based on this framework are valid.

6.2. Perceived Effects of the TAP Framework

RQ 2.1: *Is the use of the TAP framework associated with a raised awareness for the incurrence of TD?*

On the one hand, 75% of the participants stated they were aware of taking on TD at the point in time they incur it. There was no difference between the observed and the comparison unit. On the other hand, the answers to the open questions support the finding that the awareness was

raised considerably (“TD is a topic”) and that the participants started to consider when to incur or keep TD and when not to (“Sometimes TD are OK”). In the management survey, the managers also imply that the awareness was raised. This might indicate that the awareness before introducing the framework was lower in the observed unit than in the comparison unit. However, we found out that the survey question on whether the participants are aware of the TD incurrence had great potential to be misunderstood. Furthermore, developers may be biased to think they always know when they incur TD.

RQ 2.2: *Is the use of the TAP framework associated with a more conscious incurrence of TD items?*

Figure 4(a) reveals that the observed unit showed a significantly different behavior than the comparison unit (Table 4). The observed unit’s members compared different solutions and their costs before deciding on one of them. They may still have incurred TD by choosing the sub-optimal solution, but the benefit was a consciously and intentionally made decision.

RQ 2.3: *Is the use of the TAP framework associated with a better overview of TD items?*

In Figure 4(b), it is striking that in the observed IT unit, the architects were the ones who had the overview of all TD items (100% agreement). The responsibility to keep an overview of all tasks rested with the architects. We could not observe such a clear responsibility in the comparison unit. A TD overview might also be created in the comparison unit by other means. However, the evaluation indicates that the clear responsibility of the architects, which led to the TD overview, was an additional benefit of the framework.

RQ 2.4: *Is the use of the TAP framework associated with a timelier repayment of TD items?*

If the unit’s members had to incur TD, they mostly paid it back timely, as seen in Figure 2(b). Additionally, Figure 4(c) shows that two-thirds of participants in the observed IT unit perceived they repaid TD timelier. Nevertheless, the unit did not repay all TD tickets during their respective projects, and one-third of participants were still dissatisfied with the timely repayment.

RQ 4.1: *Do the managers have an overview of their IT systems’ TD?*

The observed unit’s managers had an overview of their TD. The architects were tasked to a) manage the maintenance project pipeline and b) track TD and other maintenance tasks in the backlog. The managers got their overview primarily due to the weekly meeting with the architects. They profited from the architectural evaluation of these tasks. The comparison unit’s managers did not have a comprehensive overview of their TD.

6.3. Perceived Benefits and Drawbacks of the TAP Framework

RQ 3.1/RQ 3.2: *Do practitioners observe that TD can be prevented by the adoption of the TAP framework?*

RQ	Method	Results Section	Figure/Table
RQ 1.1 (TAP framework’s reasonability)	team survey	5.2.1	figs. 3(a), 3(b), 3(d) and 3(e)
RQ 1.2 (TAP framework’s feasibility)	team survey	5.2.1	figs. 3(c) and 3(f)
	tickets statistics	5.1	fig. 2
RQ 2.1 (raised awareness)	team survey	5.2.2	
		5.2.5	
	management survey	5.3	
RQ 2.2 (conscious incurrence)	team survey	5.2.2	fig. 4(a), tab. 4
		5.2.5	
RQ 2.3 (TD items’ overview)	team survey	5.2.2	fig. 4(b)
RQ 2.4 (timely repayment)	team survey	5.2.3	fig. 4(c)
		5.2.5	
	tickets statistics	5.1	fig. 2
RQ 3.1 (TD prevention)	team survey	5.2.3	fig. 4(c)
RQ 3.2 (benefits and drawbacks)	team survey	5.2.3	fig. 4(c), fig. 4(d)
		5.2.4	tab. 5
		5.2.5	
RQ 3.3 (effort’s justification)	team survey	5.2.3	fig. 5
RQ 4.1 (managers’ TD overview)	management survey	5.3	
RQ 4.2 (benefits of the managers’ TD overview)	management survey	5.3	

Table 6: Overview of methods and figures used to answer the respective questions and results section containing details

Are there other benefits or drawbacks arising from the adoption of the TAP framework?

The most apparent perceived effect of the framework was the conscious incurrence of TD by comparing different solution options. In Figure 4(c), we presented the benefits associated with comparing solution options.

To back up these findings, we evaluated the correlation between the perceived benefits and the comparison of solution options and the estimations. We assume that the comparison and estimations can be associated with more rational discussions and decisions following the significant correlations.

The correlations (Table 5) do not support associations for TD prevention or the estimation of interest costs. The missing correlations for interest costs may be since only one-third of the participants estimate these costs. The missing correlations for TD prevention lead to the assumption that the prevention of TD is maybe not directly caused by the comparison and estimations. Still, TD prevention was perceived as a benefit by 82.4% of the participants, and it was mentioned in the open questions. Furthermore, it logically follows that rational decision-making may be able to prevent TD.

The TD overview’s perceived benefits were more rational decisions and better planning of the project pipeline by justifying TD repayments and integrating TD repayments into projects (Figure 4(d)).

Finally, the open questions support that the awareness for TD was raised, and activities like prioritization were discussed throughout the unit. Furthermore, optimized

understanding and discussions were mentioned.

The researchers observed no particular drawbacks other than the (initial) effort to use the framework (see RQ 3.3). Therefore, we asked for drawbacks in open questions. One participant each mentioned the lack of adherence to agile processes and the possible separating affect the framework may have on the team. Finally, it was suggested to better involve the operations team.

RQ 3.3: *Do practitioners find these benefits justify the additional effort?*

Figure 5 shows that most participants of the team survey think that the perceived benefits justified the additional effort for comparison, cost estimation, and the generation of the overview. These results confirm the TAP framework’s usefulness.

RQ 4.2: *What are the benefits of the TD overview from a management perspective?* All IT managers agreed that TD items became visible and, thereby, the overall awareness for TD increased. In particular, the project pipeline’s management profited from the visibility and architectural evaluation of TD. The amount of TD could be used to measure the importance of projects. The project’s durations became discernible, and the managers could postpone subsequent projects if necessary. The managers noticed that decisions regarding the incurrence of TD were made more conscious and, by this, TD could be prevented. The communication between IT management and teams improved. The better overview led to better communication with the customers and a better acceptance of repayment measures. Ultimately, TD management has been in

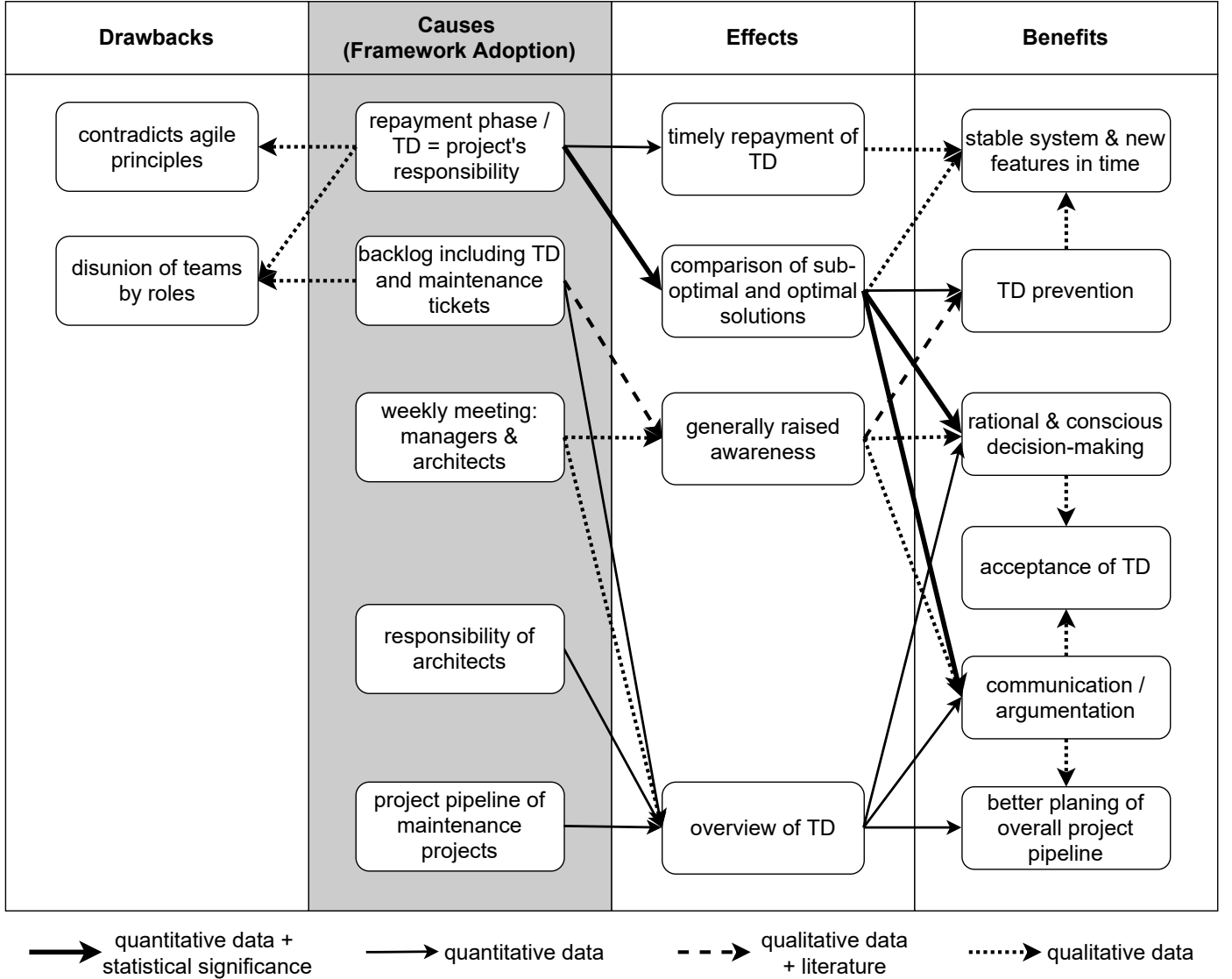


Figure 6: Causes, perceived effects, and perceived benefits and drawbacks (arrows show effect-direction) of the TAP framework

the customer's interest as it provides stable systems and new features in time.

6.4. Threats to Validity

Construct Validity. To enhance the construct validity, we questioned and compared two units, one of which did adopt the TAP framework and one did not use any method for managing TD. As presented in Section 4.2, the scope and size of the comparison unit differ from the observed unit. However, the organizational form, particularly the agile-managed projects with deadlines, is identical. Since the time limitation is the most important basis for a meaningful framework adoption, we assume that the comparison of units is valid.

To ensure comprehensibility and to optimize the questionnaire, we did pilot tests. It could be a problem that the team survey did not ask directly but only in open questions for drawbacks. Furthermore, the quantitative

survey's construction was not preceded by a qualitative survey but only by the researcher's observations. The open questions and the question about the effort's justification should have addressed parts of this problem. Additionally, we not only evaluated the questionnaire but also the ticket's statistics. This data triangulation improves the construct validity. For observer triangulation, two researchers independently coded the open questions of the team survey and the management survey. Only three managers answered the follow-up management survey. Thus, this survey only provides qualitative insights. However, the unit manager who answered this survey is the co-author of this paper and may be biased. Yet, this co-author was only involved in developing the framework and writing the framework section. He was not involved in the scientific evaluation. Still, he may be biased toward the positive impact of the framework. Two co-authors utterly uninvolved with the company enhance the validity

despite a possible author's bias.

Conclusion Validity. The paper was able to show statistically significant differences between the observed and the comparison unit. We used SPSS and standard statistic techniques to evaluate the findings' significance. Nonetheless, the sample was very small as it can only encompass the two IT units, and the staff survey had an overall response rate of 55%. We carefully considered the small sample size when interpreting the findings. We suggest supporting the findings and assumptions by further replicating this study to gain more data points.

Internal Validity. We observed valuable effects, but the same effects could have been observed in other IT units. This threat was minimized by the comparison with a unit not using the TAP framework. Nevertheless, just the management of TD could have led to better developer morale as presented by Besker et al. [48] and consequently to good ratings. This threat can and should be minimized by replicating the study in other constellations. As correlations do not directly lead to causalities, the mentioned interpretations of the correlations should be validated, e.g., by follow-up interviews.

External Validity. The TAP framework has been developed and tested in an industrial environment, and it has been in use for a long time. Both these facts are meaningful benefits of this research. Nevertheless, this was just one case study, and it may be biased. We suggest replicating the framework adoption and the study in other units and companies, economic sectors, and countries to confirm the findings. Furthermore, the comparison unit should adopt the framework, or the reasons for not adopting this should be evaluated. We suggest focusing on the most relevant parts which were already mentioned in Section 3.4: the distinction in processing intentional TD as TD tickets and unintentional TD as part of maintenance tickets, the immediate recording of intentional TD, and the processing of intentional TD as part of the project they were incurred in.

For this, we provided the framework information in this paper and the questionnaires and raw data online⁸.

7. CONCLUSION AND FUTURE WORK

In this paper, we present and evaluate the TAP framework developed and used in practice for a long time. The framework uses TD tickets for intentionally incurred TD to include TD management in the project management timeline. These TD tickets transfer the responsibility for TD incurred during a project to the same project's project management. The team records TD tickets at the time they incur the TD. This creates a feedback loop of team and project management behavior in the observed unit. They perceived the following benefits:

- The communication between different stakeholders, including IT management and customers, is optimized, and discussions between the unit members are led more rationally.
- The overall awareness for TD rises by making TD visible and providing an overview of TD.
- TD tickets raise the awareness for the decision of whether TD should be incurred and make these decisions conscious and intentional.
- TD items unintentionally incurred due to unconscious decisions can be prevented.
- TD items intentionally incurred during a project are part of the project and are paid back timelier.

Furthermore, TD tickets provide an in-time overview for the IT managers as these items are tracked during the project. The management survey indicates that this enables the managers to react and adjust the project pipeline if necessary.

While the idea of maintenance tickets and projects is not new, the idea of TD tickets for intentional TD and the proposed processing of these tickets as part of the project is a novel approach. To the best of the authors' knowledge, such an approach can not be found in the research literature so far.

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

For researchers, this paper provides insights into procedures for managing TD developed in the industry. The paper indicates the importance of providing a solution for managing and preventing TD under the pressure of tight deadlines. Furthermore, this research confirms that the responsibility for TD needs to be transferred to the whole unit. Developer teams cannot solve these problems alone. The TAP framework provides a solution for these problems.

For future work, we believe it could be helpful to supplement the framework with more formal processes for some TD activities, e.g., automated TD identification, TD measurement, or TD prioritization. In particular, approaches for TD prioritization may help optimize the repayment phase, e.g., a consequent distinction of potential and effective TD [46], as our research uncovered this to be a problem.

The effects of TD management on the planning of project pipelines is an intriguing aspect. Many underlying problems of TD management refer to communications issues between stakeholders, e.g., managers and developers. Hence,

⁸<https://doi.org/10.5281/zenodo.5788222>

we think that the perspectives of different stakeholders, e.g., line managers, on TD is another relevant research topic, which we would like to evaluate further.

At various points, we suggested replicating the study. However, this framework was designed as an actual frame to be adopted in whole or part. This is not a weakness of the framework, but takes into account the reality in which IT units' processes need to be adapted to their own needs. We can see this, e.g., in the ITIL (Information Technology Infrastructure Library) processes or agile development methods. We hope to find other units that are willing to adopt all or some of the relevant parts of the framework mentioned in Section 3.4. It would be interesting to evaluate which parts may lead to similar effects and benefits in other companies.

Acknowledgment

We would like to thank the observed and comparison unit of *Gruner+Jahr GmbH - Information Technology Department* for contributing to the framework, completing the questionnaires, and supporting this research.

References

- [1] W. Cunningham, The WyCash portfolio management system, in: Proceedings of the Conference on Object-Oriented Programming Systems, Languages, and Applications, OOPSLA, Vol. 2, 1992, pp. 29–30. doi:10.1145/157710.157715.
- [2] P. Avgeriou, P. Kruchten, I. Ozkaya, C. Seaman, Managing Technical Debt in Software Engineering, Dagstuhl Reports 6 (4) (2016) 110–138. doi:10.4230/DagRep.6.4.110.
- [3] P. Avgeriou, P. Kruchten, R. L. Nord, I. Ozkaya, C. Seaman, Reducing Friction in Software Development, IEEE Software 33 (1) (2016) 66–73. doi:10.1109/MS.2016.13.
- [4] N. Rios, R. O. Spínola, M. Mendonça, C. Seaman, The practitioners' point of view on the concept of technical debt and its causes and consequences: a design for a global family of industrial surveys and its first results from Brazil, Empirical Software Engineering 25 (5) (2020) 3216–3287. doi:10.1007/s10664-020-09832-9.
- [5] R. Verdecchia, P. Kruchten, P. Lago, I. Malavolta, Building and evaluating a theory of architectural technical debt in software-intensive systems, Journal of Systems and Software 176 (2021) 110925. doi:10.1016/j.jss.2021.110925.
- [6] S. McConnell, Managing technical debt, Construx Inc. (2008).
- [7] M. Fowler, Technical debt (2019). URL <https://martinfowler.com/bliki/TechnicalDebt.html>
- [8] D. Tsoukalas, D. Kehagias, M. Siavvas, A. Chatzigeorgiou, Technical debt forecasting: An empirical study on open-source repositories, Journal of Systems and Software 170 (2020) 110777. doi:https://doi.org/10.1016/j.jss.2020.110777.
- [9] S. Freire, N. Rios, B. Gutierrez, D. Torres, M. Mendonça, C. Izurieta, C. Seaman, R. O. Spínola, Surveying Software Practitioners on Technical Debt Payment Practices and Reasons for Not Paying off Debt Items, in: Proceedings of the Evaluation and Assessment in Software Engineering, EASE '20, Association for Computing Machinery, New York, NY, USA, 2020, pp. 210–219. doi:10.1145/3383219.3383241.
- [10] C. Apa, H. Jeronimo, L. M. Nascimento, D. Vallespir, G. H. Travassos, The perception and management of technical debt in software startups, Fundamentals of Software Startups: Essential Engineering and Business Aspects (2020) 61–78doi:10.1007/978-3-030-35983-6_4.
- [11] J. Yli-Huumo, A. Maglyas, K. Smolander, How do software development teams manage technical debt? – An empirical study, Journal of Systems and Software 120 (2016) 195–218. doi:https://doi.org/10.1016/j.jss.2016.05.018.
- [12] J. D. Morgenthaler, M. Gridnev, R. Sauciu, S. Bhansali, Searching for build debt: Experiences managing technical debt at Google, 2012 3rd International Workshop on Managing Technical Debt, MTD 2012 - Proceedings (2012) 1–6doi:10.1109/MTD.2012.6225994.
- [13] T. Besker, A. Martini, J. Bosch, Carrot and Stick Approaches When Managing Technical Debt, in: Proceedings of the 3rd International Conference on Technical Debt, TechDebt '20, Association for Computing Machinery, New York, NY, USA, 2020, pp. 21–30. doi:10.1145/3387906.3388619.
- [14] B. Pérez, C. Castellanos, D. Correal, N. Rios, S. Freire, R. Spínola, C. Seaman, C. Izurieta, Technical debt payment and prevention through the lenses of software architects, Information and Software Technology 140 (2021) 106692. doi:https://doi.org/10.1016/j.infsof.2021.106692.
- [15] S. S. Freire, N. Rios, B. Pérez, C. Castellanos, D. Correal, R. Ramač, V. Mandić, N. Taušan, G. López, A. Pacheco, D. Falelli, M. Mendonça, C. Izurieta, C. Seaman, R. Spínola, How Experience Impacts Practitioners' Perception of Causes and Effects of Technical Debt, in: 2021 IEEE/ACM 13th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE), 2021, pp. 21–30. doi:10.1109/CHASE52884.2021.00011.
- [16] J. C. Rocha, V. Zapalowski, I. Nunes, Understanding Technical Debt at the Code Level from the Perspective of Software Developers, in: Proceedings of the 31st Brazilian Symposium on Software Engineering, SBES'17, Association for Computing Machinery, New York, NY, USA, 2017, pp. 64–73. doi:10.1145/3131151.3131164.
- [17] P. Kruchten, R. L. Nord, I. Ozkaya, Technical debt: From metaphor to theory and practice, IEEE Software 29 (6) (2012) 18–21. doi:10.1109/MS.2012.167.
- [18] U. Eliasson, A. Martini, R. Kaufmann, S. Odeh, Identifying and visualizing Architectural Debt and its efficiency interest in the automotive domain: A case study, in: 2015 IEEE 7th International Workshop on Managing Technical Debt (MTD), 2015, pp. 33–40. doi:10.1109/MTD.2015.7332622.
- [19] Y. Guo, R. O. Spínola, C. Seaman, Exploring the costs of technical debt management – a case study, Empirical Software Engineering 21 (1) (2016) 159–182. doi:10.1007/s10664-014-9351-7.
- [20] T. Besker, A. Martini, J. Bosch, Technical Debt Triage in Backlog Management, in: 2019 IEEE/ACM International Conference on Technical Debt (TechDebt), TechDebt '19, IEEE Press, 2019, pp. 13–22. doi:10.1109/TechDebt.2019.00010.
- [21] M. Soliman, P. Avgeriou, Y. Li, Architectural design decisions that incur technical debt — An industrial case study, Information and Software Technology 139 (2021) 106669. doi:https://doi.org/10.1016/j.infsof.2021.106669.
- [22] K. Borowa, A. Zalewski, S. Kijas, The Influence of Cognitive Biases on Architectural Technical Debt, in: International Conference on Software Architecture (ICSA), 2021, pp. 115–125. doi:10.1109/ICSA51549.2021.00019.
- [23] T. Besker, J. Bosch, A. Martini, J. Bosch, Technical debt cripples software developer productivity: A longitudinal study on developers' daily software development work, in: Proceedings - International Conference on Software Engineering, ACM, 2018, pp. 105–114. doi:10.1145/3194164.3194178.
- [24] A. Martini, J. Bosch, M. Chaudron, Architecture Technical Debt: Understanding Causes and a Qualitative Model, in: 2014 40th EUROMICRO Conference on Software Engineering and Advanced Applications, 2014, pp. 85–92. doi:10.1109/SEAA.2014.65.
- [25] Z. Li, P. Avgeriou, P. Liang, A systematic mapping study on technical debt and its management, Journal of Systems and Software 101 (2015) 193–220. doi:10.1016/j.jss.2014.12.027.

- [26] N. A. Ernst, S. Bellomo, I. Ozkaya, R. L. Nord, I. Gorton, Measure it? Manage it? Ignore it? Software practitioners and technical debt, 10th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, ESEC/FSE 2015 - Proceedings (2015) 50–60doi:10.1145/2786805.2786848.
- [27] V. Lenarduzzi, T. Orava, N. Saarimäki, K. Systä, D. Taibi, K. Systä, D. Taibi, K. Systä, D. Taibi, An Empirical Study on Technical Debt in a Finnish SME, in: 2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), Vol. 2019-Septe, 2019, pp. 1–6. doi:10.1109/ESEM.2019.8870169.
- [28] S. Freire, N. Rios, M. Mendonça, D. Falessi, C. Seaman, C. Izurieta, R. O. Spínola, R. O. Spínola, Actions and impediments for technical debt prevention: Results from a global family of industrial surveys, in: Proceedings of the ACM Symposium on Applied Computing, SAC '20, Association for Computing Machinery, New York, NY, USA, 2020, pp. 1548–1555. doi:10.1145/3341105.3373912.
- [29] B. Vogel-Heuser, F. Bi, Interdisciplinary effects of technical debt in companies with mechatronic products — a qualitative study, Journal of Systems and Software 171 (2021) 110809. doi:https://doi.org/10.1016/j.jss.2020.110809.
- [30] J. Yli-Huumo, T. Rissanen, A. Maglyas, K. Smolander, L.-M. Sainio, The Relationship Between Business Model Experimentation and Technical Debt, Lecture Notes in Business Information Processing 210 (2015) 5–6. doi:10.1007/978-3-319-19593-3.
- [31] S. Charalampidou, A. Ampatzoglou, A. Chatzigeorgiou, N. Tsiroidis, Integrating traceability within the IDE to prevent requirements documentation debt, Proceedings - 44th Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2018 (2018) 421–428doi:10.1109/SEAA.2018.00075.
- [32] Y. Yang, R. Michel, J. Wade, D. Verma, M. Törngren, T. Alelyani, Towards a taxonomy of technical debt for COTS-intensive cyber physical systems, Procedia Computer Science 153 (2019) 108–117. doi:https://doi.org/10.1016/j.procs.2019.05.061.
- [33] D. Feitosa, A. Ampatzoglou, A. Gkortzis, S. Bibi, A. Chatzigeorgiou, CODE reuse in practice: Benefiting or harming technical debt, Journal of Systems and Software 167 (2020) 110618. doi:https://doi.org/10.1016/j.jss.2020.110618.
- [34] T. Sharma, G. Suryanarayana, G. Samarthiyam, Challenges to and Solutions for Refactoring Adoption: An Industrial Perspective, IEEE Software 32 (6) (2015) 44–51. doi:10.1109/MS.2015.105.
- [35] M. Kaiser, G. Royse, Selling the Investment to Pay Down Technical Debt: The Code Christmas Tree, in: 2011 Agile Conference, 2011, pp. 175–180. doi:10.1109/AGILE.2011.50.
- [36] R. K. Gupta, P. Manikreddy, S. Naik, K. Arya, Pragmatic approach for managing technical debt in legacy software project, in: Proceedings of the 9th India Software Engineering Conference, Vol. 18-20-Febr of ISEC '16, Association for Computing Machinery, New York, NY, USA, 2016, pp. 170–176. doi:10.1145/2856636.2856655.
- [37] O. Baysal, R. Holmes, M. W. Godfrey, Developer dashboards: The need for qualitative analytics, IEEE Software 30 (4) (2013) 46–52. doi:10.1109/MS.2013.66.
- [38] C. Treude, M. A. Storey, Awareness 2.0: Staying aware of projects, developers and tasks using dashboards and feeds, Proceedings - International Conference on Software Engineering 1 (2010) 365–374. doi:10.1145/1806799.1806854.
- [39] A. Martini, J. Bosch, An Empirically Developed Method to Aid Decisions on Architectural Technical Debt Refactoring: AnaConDebt, in: 2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C), 2016, pp. 31–40.
- [40] S. Baars, S. Meester, CodeArena: Inspecting and Improving Code Quality Metrics using Minecraft, in: 2019 IEEE/ACM International Conference on Technical Debt (TechDebt), 2019, pp. 68–70. doi:10.1109/TechDebt.2019.00023.
- [41] M. A. Al Mamun, C. Berger, J. J. Hansson, Explicating, Understanding, and Managing Technical Debt from Self-Driving Miniature Car Projects, in: 2014 Sixth International Workshop on Managing Technical Debt, Institute of Electrical and Electronics Engineers Inc., 2014, pp. 11–18. doi:10.1109/MTD.2014.15.
- [42] J.-L. Letouzey, M. Ilkiewicz, Managing Technical Debt with the SQALE Method, IEEE Software 29 (6) (2012) 44–51. doi:10.1109/MS.2012.129.
- [43] G. S. Tonin, A. Goldman, C. Seaman, D. Pina, Effects of Technical Debt Awareness: A Classroom Study, in: H. Baumeister, H. Lichter, M. Riebsch (Eds.), 18th International Conference, XP 2017, Vol. 283, 2017, pp. 84–100. doi:10.1007/978-3-319-57633-6_6.
- [44] Y. Guo, C. Seaman, F. Q. da Silva, F. Q.B. da Silva, F. Q. da Silva, F. Q.B. da Silva, Costs and obstacles encountered in technical debt management – A case study, Journal of Systems and Software 120 (2016) 156–169. doi:10.1016/j.jss.2016.07.008.
- [45] N. Ramasubbu, C. F. Kemerer, Integrating technical debt management and software quality management processes: A framework and field test, Test Engineering and Management 45 (7-8) (2019) 883. doi:10.1145/3180155.3182529.
- [46] K. Schmid, A formal approach to technical debt decision making, QoSA 2013 - Proceedings of the 9th International ACM Sigsoft Conference on the Quality of Software Architectures (2013) 153–162doi:10.1145/2465478.2465492.
- [47] F. Bachmann, R. L. Nord, I. Ozkaya, Architectural tactics to support rapid and agile stability, CrossTalk 25 (3) (2012) 20–25.
- [48] T. Besker, H. Ghanbari, A. Martini, J. Bosch, The influence of Technical Debt on software developer morale, Journal of Systems and Software 167 (2020) 110586. doi:https://doi.org/10.1016/j.jss.2020.110586.
- [49] J. Yli-Huumo, A. Maglyas, K. Smolander, J. Haller, H. Törnroos, Developing processes to increase technical debt visibility and manageability – An action research study in industry, in: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Vol. 10027 LNCS, 2016, pp. 368–378. doi:10.1007/978-3-319-49094-6_24.
- [50] R. Rebouças de Almeida, R. do Nascimento Ribeiro, C. Treude, U. Kulesza, Business-Driven Technical Debt Prioritization - An Industrial Case Study, in: 2021 IEEE/ACM International Conference on Technical Debt (TechDebt), 2021, pp. 74–83. arXiv:2010.09711, doi:10.1109/TechDebt52882.2021.00017.
- [51] S. Malakuti, S. Ostroumov, The quest for introducing technical debt management in a large-scale industrial company, in: A. Jansen, I. Malavolta, H. Muccini, I. Ozkaya, O. Zimmermann (Eds.), Software Architecture. ECSA 2020. Lecture Notes in Computer Science, Vol. 12292, Springer International Publishing, 2020, pp. 296–311. doi:10.1007/978-3-030-58923-3_20.
- [52] P. Runeson, M. Host, A. Rainer, B. Regnell, Case Study Research in Software Engineering, John Wiley and Sons Ltd, 2012. doi:10.1007/978-3-030-49392-9.
- [53] A. Joshi, S. Kale, S. Chandel, D. Pal, Likert Scale: Explored and Explained, British Journal of Applied Science & Technology 7 (4) (2015) 396–403. doi:10.9734/bjast/2015/14975.
- [54] H. B. Mann, D. R. Whitney, On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other, The Annals of Mathematical Statistics 18 (1) (1947) 50–60. doi:10.1214/aoms/1177730491.
- [55] J. Cohen, Statistical Power Analysis for the Behavioral Sciences, Lawrence Erlbaum Associates, New York, New York, USA, 1988.
- [56] K. Pearson, I. Mathematical contributions to the theory of evolution. —VII. On the correlation of characters not quantitatively measurable, Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character 195 (262-273) (1901) 1–47. doi:10.1098/rsta.1900.0022.