Submission date: 23/09/2022
Resubmission date: 20/07/2023
Camera ready submission date: 25/04/2024

1st round notification: 16/11/2022
2nd round notification: 17/04/2024
Available online: 29/04/2024
Publication date: 29/04/2024

Section: Regular article

A Method to Support Documentation Technical Debt Management

Leonardo Mendes[1], **Cristina Cerdeiral**[2], **Gleison Santos**[1]

[1]Programa de Pós-Graduação em Informática – Universidade Federal do Estado do Rio de Janeiro (UNIRIO)
Rio de Janeiro, RJ – Brazil

[2]Nest Wealth
Toronto, ON – Brazil

{leonardo.cabral, gleison.santos}@uniriotec.br,
cristina.cerdeiral@gmail.com

Abstract. *Documentation technical debt typically is related to non-existent, inconsistent, outdated, unnecessary, or incomplete documentation. Such problems may lead to negative consequences to the software product development and maintenance endeavors, for instance, difficulties identifying and removing defects and increased cost and effort to deploy new versions. We followed Design Science Research principles to create DOTED, a method for managing Documentation Technical Debt on software products. We present a case study to evaluate DOTED viability in supporting the management of this type of technical debt. The study participants highlighted its benefits and reported the intention of future use. We also found that DOTED may be more effective in fostering technical debt repayment if used early and continuously on the product's development lifecycle.*

Keywords. *Documentation Technical Debt; Technical Debt Management; Technical Debt; Software Documentation; Design Science Research*

1. Introduction

Documentation is an integral part of any software development process [Zhi et al. 2015]. Documentation technical debt refers to non-existent, inadequate, incomplete, inconsistent, or unnecessary documentation, although it is often confused with requirements technical debt [Soares et al. 2015]. Incurring technical debt is inevitable to any organization, considering that assuming a debt may be, at first, favorable to increase the teams' productivity or their rate to deliver new releases [Alves et al. 2016]. Not knowing what technical debt is, its causes, its consequences, how to avoid it, and how to manage it can cause significant damage to the organizations. Thus, there is a need to support software organizations in managing technical debts. Briefly, technical debt management consists of identifying, estimating, and deciding on existing debts in a project [Zazworka et al. 2013].

Cite as: Mendes, L., Cerdeiral, C. & Santos, G. (2024). A Method to Support Documentation Technical Debt Management. iSys: Revista Brasileira de Sistemas de Informação (Brazilian Journal of Information Systems), Vol 17(1), pp-pp. 5:1-5:28 DOI: 10.5753/isys.2024.2904

Studies on how software project organizations deal with technical debt in their work environments have become more frequent [Alves et al. 2016, Li et al. 2015, Besker et al. 2022, Silva et al. 2019]. In general, technical debt is usually managed implicitly by software project organizations [Li et al. 2015, Alves et al. 2016]. However, to avoid future interest repayments and increase project effectiveness, debt management techniques should be structured and conscious. Having a technical debt management strategy can significantly impact the number of technical debt items related to the software [Besker et al. 2022]. Among the benefits obtained with the effective management of technical debt are reduced maintenance costs, increased maintenance productivity, and reduced unexpected events, leading to a better estimation of costs and risks in projects [Seaman and Guo 2011].

This paper presents DOTED (*DO*cumentation *TE*chnical *D*ebt) method, which supports organizations developing software products to manage documentation technical debt. DOTED provides support artifacts to help create a plan for managing technical debts, identifying, measuring, and resolving debts, and monitoring defined mitigation actions. Moreover, it presents a knowledge base with causes and consequences of documentation technical debt and best practices. To create DOTED, we used a research strategy based on Design Science Research [Wieringa 2014, Hevner 2007]. We also present the case study results to evaluate DOTED viability in supporting documentation technical debt management of a software product. The team involved was able to execute the method at an advanced stage of product system development. The participants highlighted the benefits of DOTED use on the organization and its adequate support regarding its purpose. We had evidence of perception of the utility, ease of use, and intention of future use. We improved the method description due to the participants' feedback.

The paper is structured as follows: Section 2 shows the related work, Section 3 presents the research method, Section 4 describes the DOTED method, Section 5 shows its evaluation, and Section 6 points our final considerations.

2. Related Work

Li et al. [Li et al. 2015] executed a mapping study to identify and analyze research on technical debt and its management. According to them, the activities related to technical debt management are Identification, Measurement, Prioritization, Prevention, Monitoring, Repayment, Documentation, and Communication. On the other hand, Seaman and Guo [Seaman and Guo 2011] do not include Monitoring and Communication activities, while Sandberg et al. [Sandberg et al. 2015] do not include Documentation and Communication. Although Li et al. [Li et al. 2015] identified many studies describing models, methods, practices, and tools focused on specific activities, none focused on DTD management nor addressed all activities related to technical debt management.

Identifying technical debt (TD) is not trivial since it can be present in different artifacts of the same software product [Zazworka et al. 2013]. Although automated strategies point to items (mainly related to source code, as in [Charalampidou et al. 2018]) that need improvement from the developers' point of view, they still do not make it clear whether they can identify essential items for other stakeholders [Alves et al. 2016]. In this case, although more time-consuming, manual TD identification practices are more accu-

rate in identifying debts, such as problems in the project or product documentation, that could not be determined by automated techniques. To support the identification activity, we provide a knowledge base on causes that assist DOTED users in manually identifying DTD. We encapsulated it in an auxiliary artifact named *DOTED Guide* (see Section 4.1 and Appendix A).

Regarding TD estimates or measurement, considering that technical debts are closely linked to software quality, quality metrics can measure debt. Technical debt can also be measured through the effort to pay it, estimating the time and effort needed to correct the possible problem [Alves et al. 2016, Snipes et al. 2012]. Other estimating approaches include calculation models, i.e., through mathematical formulas or models [Stochel et al. 2012, Seaman and Guo 2011], and human estimation, i.e., according to experience and expertise [Seaman and Guo 2011, Guo and Seaman 2011]. Strategies to prioritize which technical debts to pay have been discussed over the years [Brown et al. 2010]. For example, focusing on paying off core items first (i.e., concentrating on paying off debts that could cause significant future damage) [Alves et al. 2016], items with higher cost/benefit ratios [Zazworka et al. 2011], or items incurring higher interest [Seaman and Guo 2011]. Strategies for TD repayment include refactoring [Buschmann 2011], automation [Letouzey 2012], rewriting [Buschmann 2011], and bug fixing [Gat 2010]. DOTED's approach to DTD estimating combines human estimation with a basic calculation model considering the DTD impact and the probability of its interest. The *DOTED Guide* also presents effects of DTD that might be useful to estimate its impacts and consequences. Prioritization is also accomplished by human judgment considering the DTD estimate and the cost associated with its removal. The decisions to accept or pay the DTD and how to handle it are at the discretion of DOTED executors.

TD monitoring watches the changes in the cost and benefit of unresolved TD over time, TD representation/documentation provides a way to represent and codify TD in a uniform manner addressing the concerns of particular stakeholders, and TD communication makes identified TD visible to stakeholders so that it can be discussed and further managed [Li et al. 2015]. DOTED adopts the planned check strategy [Bohnet and Döllner 2011] for monitoring, in which the identified TD is regularly measured and tracked for changes. Li et al. [Li et al. 2015] cite the most common fields for TD documentation/representation: ID, description, location, responsible/author, type, and description. DOTED encapsulates the location into the description and substitutes the responsible/author for the person/role that identified the TD item. It also documents the cost of repaying the TD item and the probability that the interest of the TD item needs to be repaid. According to Li et al. [Li et al. 2015], the most common approaches to support TD communication are TD dashboards [Power 2013] and backlog [dos Santos et al. 2013]. The latter is also the approach implemented by DOTED.

Some DT management strategies concentrate efforts on preventing technical debts during software construction [Alves et al. 2016] since the efforts required to pay the interest resulting from the debts acquired are more significant than the efforts undertaken to prevent them. For instance, Charalampidou et al. [Charalampidou et al. 2018] propose a tool-based approach for preventing documentation TD during requirements engineering by enabling the real-time creation of traces between requirements and code. Other

examples include practices to help prevent non-intentional debts [Yli-Huumo et al. 2016], such as having standards for code comments to make them legible and modifiable [Green and Ledgard 2011]. DOTED does not present an activity for explicit TD prevention. Instead, it relies on human factors analysis [Li et al. 2015, Michael Golden 2010], in which a culture that minimizes the unintentional TD caused by human factors is cultivated. Support for this type of analysis is based on the use of the *DOTED Guide*, which presents best practices that can indirectly support the prevention of DTD on other documents created during the product lifecycle (and not only on the source code or requirements) if applied, especially, in the early phases of the product lifecycle.

3. Research Method

The Design Science Paradigm seeks to create innovations that define the ideas, practices, technical capabilities, and products through which the analysis, design, implementation, management, and use of information systems can be effectively and efficiently accomplished [Hevner et al. 2004]. Design Science Research (DSR) focuses on solving a practical problem in a specific context through an artifact, consequently generating new scientific knowledge [Hevner et al. 2004, Wieringa 2014]. We follow the theoretical line of Hevner [Hevner 2007], which consists of the association of three cycles of related activities: *Relevance Cycle*, *Design Cycle*, and *Rigor Cycle*. We decided to use DSR as it allows us to incrementally design and evaluate an artifact, acquiring and creating knowledge along the way (which led to *DOTED Guide*) and how to handle it.

The *Relevance Cycle* involves defining the problem to be addressed, the research requirements, and the criteria for evaluating the research results. In this cycle, we performed a informal literature review on software documentation, technical debt, and technical debt management. The studied problem involves organizations' difficulties in managing problems with software documentation. We chose to address documentation technical debt based on (i) the low number of publications explicitly dealing with Documentation Technical Debt (as per [Alves et al. 2016]), (ii) the potential high impact of DTD in long product development cycles due to lack of adequate document management and lack of update of specification documents during the maintenance phase of a product, (iii) the first author's practical experience on development projects suffering from poor documentation. Thus, we defined the following research question: *"How to support organizations that develop software products in the management of documentation technical debts?"* As a result, we proposed an artifact representing the method described in this paper: DOTED (Documentation Technical Debt).

We outlined five requirements for DOTED: *(R1)* Supporting the main activities associated with managing technical debt; *(R2)* Supporting organizations to manage documentation technical debt originated from all software lifecycle phases; *(R3)* Supporting the management of documentation technical debt at any point in the software lifecycle, from requirements elicitation to maintenance; *(R4)* Providing artifacts to help organizations identify and record documentation technical debts; *(R5)* Providing a knowledge base on documentation technical debt to foster the method usage.

Such requirements were defined based on an ad-hoc literature review. According to Li et al. [Li et al. 2015], the activities related to technical debt management are Iden-

tification, Measurement, Prioritization, Prevention, Monitoring, Repayment, Documentation, and Communication. DOTED supports all of them but Prevention (R1). Moreover, we used the studies in [Li et al. 2015] to inspire how DOTED addresses these activities. According to Alves et al. [Alves et al. 2016] and Silva et al. [Silva et al. 2019], software organizations may incur technical debt at any stage of development, which may manifest itself in different ways. So, it is necessary to manage the documentation debt in each phase of the software lifecycle (R2 and R3). Ernst et al. [Ernst et al. 2015] further reinforce that artifacts are fundamental components to be applied in any technical debt management strategy (R4). Based on the assumption that software engineers lack knowledge about technical debt, and also documentation technical debt, concepts and influence on software projects [Alves et al. 2016, Silva et al. 2019] we propose a knowledge base of causes, consequences, and best practices associated with documentation technical debt to enhance the team ability to identify possible sources of debts, identify possible effects that may help estimating their impacts, and best practices that can inspire how to treat them (R5).

We defined three acceptance criteria to evaluate DOTED: viability, usefulness, and compliance with the established requirements.

The *Design Cycle* is related to constructing and evaluating the created artifact. In this work, the artifact is the DOTED method. The construction of DOTED is based on a literature review and qualitative study of the technical debt of documentation that gave rise to the *DOTED Guide* [Mendes et al. 2019] (see Section 4.1.1). The DOTED evaluation was based on a participative case study (see Section 5).

The *Rigor Cycle* refers to using and generating knowledge. Rigor is achieved by appropriately using foundations and methodologies from a knowledge base grounding the research and adding knowledge generated by the study to contribute to the growing knowledge base [Hevner 2007]. The main foundations we used were the literature on technical debt, qualitative analysis, and case study guidelines [Baskerville 1997, Runeson et al. 2012], and TAM (Technology Acceptance Model) [Davis 1989]. Thus, this cycle also includes research contributions to the knowledge base: DOTED itself, the case study to evaluate it in the industry, and the *DOTED Guide*. Also important to note is that the *DOTED Guide* source material [Mendes et al. 2019] was utilized as a reference for the Theoretical Framework of Documentation Debt outlined in [Rios et al. 2020].

According to Hevner [Hevner 2007], the Relevance, Design, and Rigor cycles are closely related. Also, many times their execution is intertwined. To create DOTED, we executed three iterations combining them. The *first iteration* focus was the execution of *Relevance Cycle*. During that, we also gathered necessary knowledge on technical debt, contributing to the foundations described in the *Rigor Cycle*. During this iteration, we realized the importance of easing the identification of possible documentation technical debts. Therefore, the *second iteration* aimed to create *DOTED Guide* (see Section 4.1.1). To make it, we performed activities related to the *Rigor Cycle* (i.e., identifying knowledge about conducting thematic analyses) and *Design Cycle* (i.e., *DOTED Guide* creation itself). In the *third iteration*, we proposed the DOTED method and evaluated it. Therefore, we executed the *Design Cycle* and the *Rigor Cycle*, not only identifying and using pre-

existing knowledge (such as the guidelines to execute case studies and how to use TAM [Davis 1989], but also making explicit the knowledge acquired by using DOTED in the industry.

4. The DOTED Method

DOTED aims to support organizations that develop software products in managing documentation technical debt. We created DOTED to be agnostic of the software development process, i.e., we expect it to be used with traditional, hybrid, or agile processes. It can be used during a (long) software development or maintenance project (as defined by PMBOK [PMI 2017]) or without creating projects at all should the organization has a dedicated team handling the product evolution continuously.

To run DOTED, two main parts are foreseen: the *Mentor* and the *Team*. The Mentor is directly involved in all phases and is responsible for supporting and guiding the team during DOTED execution. The Mentor must know project management and software development practices and the concepts involved in technical debt management. The Team is made up of professionals who design and develop the software. A product may involve one or more teams whose members may have different technical knowledge (e.g., design, development, testing, etc.). Team participation occurs at all stages but is paramount while identifying, measuring, and resolving documentation technical debts.

Next, we describe DOTED primary documents (Section 4.1) and phases (Section 4.2). We also discuss how DOTED should be iteratively executed (Section 4.3).

4.1. DOTED Artifacts

The main artifacts used by DOTED are described below. The *DOTED Guide* can be seen in Appendix A.

4.1.1. DOTED Guide

The *DOTED Guide* (which can be seen in Appendix A) is a knowledge base with possible DTD causes (Table 5) and effects (Table 4), and best practices to avoid them (Table 6).

It was built based on three primary sources. Due to the lack of information on the challenges, sources, or effects of Documentation TD, we executed the study based on the Thematic Analysis [Cruzes and Dyba 2011] of semi-structured interviews with a product development team [Mendes et al. 2019]. We later expanded it using sources identified in an ad-hoc review on software product documentation problems and best practices (e.g., [Clements et al. 2003, Martini and Bosch 2015, IEEE 1998, Charalampidou et al. 2018, Bourque et al. 2014, Melo et al. 2016]) and technical debt [Rios et al. 2018] as we did not find many publications focused on Documentation TD. We coded both findings using the same approach. Then, the feedback and the results from the case study (executed in a different context/team) allowed us to expand *DOTED Guide* content once again.

Each organization may adapt and build on *DOTED Guide* based on its own (existing or acquired) knowledge. Its content is used during the creation of the *DTDManagementPlan* and *DTDResolutionPlan* and must be fed back after each execution cycle.

4.1.2. DTDManagementPlan

The *Documentation Technical Debt Management Plan*, or *DTDManagementPlan*, is used to record general data about the product, including the expected documentation and the staff involved in the management of documentation technical debts (DTD). This document must describe (i) the name of the software product for which the method is performed, (ii) the development process used by the organization, (iii) the teams responsible for the product and the execution of DOTED, (iv) both used and expected documentation for the product, (v) DTD causes and effects, and (vi) best practices that can mitigate the DTD occurrence.

4.1.3. DTDBacklog

The *Record of Technical Debts*, or *DTDBacklog*, maintains the history of the identified DTDs, in addition to the measurement of criticality and the benefit of removing the DTDs. For each DTD identified, the following must be informed: a unique identifier, causes of the debt, the associated knowledge area (i.e., Project Management, Requirements, Design, Construction, Tests, Maintenance), the person responsible and the date of identification, other related DTDs. The following must be informed to control the debt repayment: priority, status (i.e., Pending, In Progress, Paid, Accepted), completion date, and useful notes. Debts are measured using the *Criticality Matrix* and the *Benefit Removal Matrix*. Optionally, the Mentor and Team may want to act preventively and only record debt causes, aiming to eliminate them and, thus, avoid new DTD occurrences.

The *Criticality Matrix* (Fig. 1) is generated by multiplying (i) the probability that an identified DTD, if not paid, will require extra effort in the future to be resolved and (ii) the estimated intensity of its impact. The first is to assess the probability of interest, that is, the need to make extra efforts to pay the debt in the future. For each DTD, the team should record the probability of interest occurring, classifying it as High (i.e., the chance of interest occurring on the debt is high and frequently happens), Medium (i.e., occasional probability of interest occurring), or Low (i.e., little or almost no chance of interest occurring). Then, the impact of debts on the product must be assessed as High (i.e., severe consequences to the product, causing partial or permanent damage), Medium (i.e., momentary consequences to the product that can be corrected at any time, without severe damage), or Low (i.e., almost imperceptible consequences to the product that can be easily corrected).

		Impact		
		Low	Medium	High
Probability	High	Medium	High	High
	Medium	Low	Medium	High
	Low	Low	Low	Medium

Figura 1. DTD Criticality Matrix (i.e. Probability x Impact)

The *Benefit Removal Matrix* (Fig. 2) is generated from the probability x impact

of the debt assessment (obtained from the *Criticality Matrix* and the estimated cost of debts). The estimated cost of each documentation technical debt must be recorded. Costs can be considered *High* (i.e., greater than 20% of the total available time or resources), *Medium* (i.e., from 10% to 20% of the total available time or resources), or *Low* (i.e., less than 10% of the available time or resources). The presented values can be modified and adapted to the organization's reality in which the DOTED is executed. In the end, the DTD resolution benefits are classified as *High*, *Medium*, or *Low*. Values associated with related DTD are also presented to support decision-making.

		Cost		
		Low	Medium	High
Criticality	High	High	High	Medium
	Medium	High	Medium	Low
	Low	Medium	Low	Low

Figura 2. DTD Benefit Removal Matrix (i.e., Criticality x Cost)

For both matrices, we opted for the ordinal scale depicted in Fig. 1 and 2 to quantify the DTD impact, the probability of interest occurring, and the cost of removing DTD. Nevertheless, the organization can adapt the artifact to provide numerical values instead. The team must use their experience to estimate the values and historical data gathered during product development and maintenance.

4.1.4. DTDResolutionPlan

The *Documentation Technical Debt Resolution Plan*, or *DTDResolutionPlan*, is organized by resolution cycles. The DTDs to be resolved within a predefined period are recorded for each cycle. The proposed solutions to be applied to each documentation technical debt, the estimated hours (total and per working day) to resolve the DTDs, the number of hours spent, other related DTDs, and the resolution situation (i.e., *Pending*, *In Progress*, *Solved / Paid*, *Not Solved* and *Accepted*).

4.2. DOTED Phases and Activities

To improve the teams' ability to deal with DTD with little experience and assistance, we described DOTED phases and outcomes, explicitly representing all tasks as a small process to guide and ease their execution correctly. That way, we provide detailed descriptions of activities to clarify what to do (i.e., as a process) and how to do it (i.e., as a method). Based on our experience, we believe a detailed 'execution guide' is important so project members can follow a method properly and, that way, minimize possible difficulties you draw attention to.

As follows, we describe DOTED phases.

4.2.1. Creating or Reviewing the Documentation Technical Debt Management Plan

Fig. 3 presents DOTED first phase. In this phase, the Mentor details the product context for which the method will be executed and identifies the product team and the team responsible for managing technical debts. The documentation technical debt items to be managed are determined based on existing internal organizational references (such as documents and personal experience) and the *DOTED Guide*. This phase main result is the *DTDManagementPlan*, which is created the first time the phase is executed and is reviewed at each execution cycle end as the documentation approach should be constantly reviewed to evaluate if it is adding value the way it is.

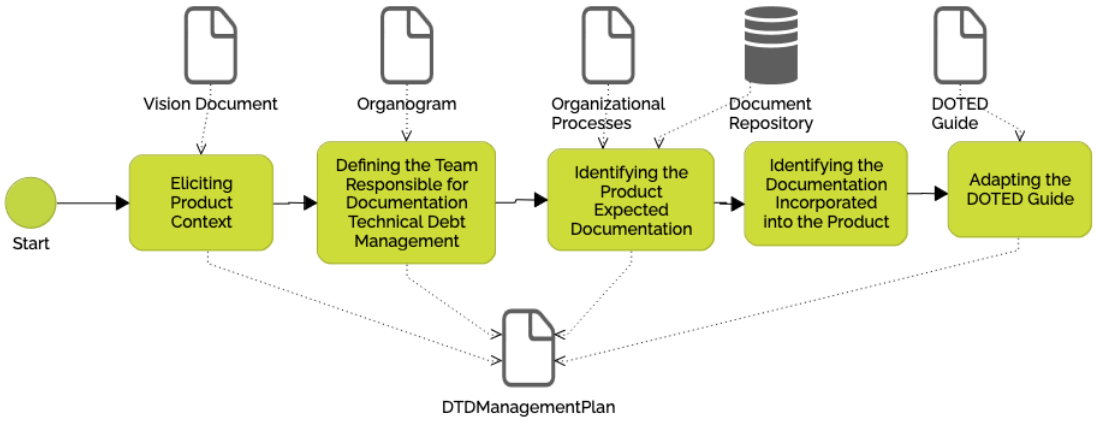

Figura 3. Phase Creating or Reviewing the Documentation Technical Debt Management Plan

Activity *Eliciting Product Context* aims to gather information about the product to be managed. The Mentor is responsible for collecting the necessary information. As needed, the Mentor is supported by the Team and other relevant product stakeholders. It is necessary to (i) identify the set of product characteristics relevant to the understanding of the organization's development process; (ii) identify the *Product Team* that participates in its construction and maintenance; and (iii) identify the involved software processes.

The activity *Defining the Team Responsible for Document Technical Debt Management* consists of defining which team members will be responsible for analyzing the documentation related to their technical knowledge. Then, in the activity *Identifying the Product Expected Documentation*, team members should review the organization's development processes to identify expected documentation. It is important to note that the absence of essential artifacts for the processes reveals the presence of documentation technical debt in the product.

Activity *Identifying the Documentation Incorporated into the Product* aims to identify the "unofficial" documentation, which is not expected by the company's established procedures and processes, but the team has been decided on its own to adopt them. For instance, the team may create user stories and prototypes instead of relying only on use case descriptions. Once this documentation is part of the team work's reality, it is necessary to avoid technical debts in these artifacts.

In the activity *Adapting the DOTED Guide*, the Mentor and the Team consult the *DOTED Guide* (see Section 4.1.1 and Appendix A) to create the version of it that will be used in the product context. They must identify which elements present in the Guide are inherent to the reality of the product, thus generating an adapted version of the product in question. A predefined list of DTD items is essential to identify, measure, and decide which debt items should be paid.

4.2.2. Identifying and Measuring Documentation Technical Debts

Fig. 4 presents DOTED second phase. In this phase, the technical documentation debts must be identified. Moreover, for each one, the Mentor must evaluate the relationship between the debt probability and impact and the cost-benefit of paying it. The information collected is recorded in the *DTDBacklog*.

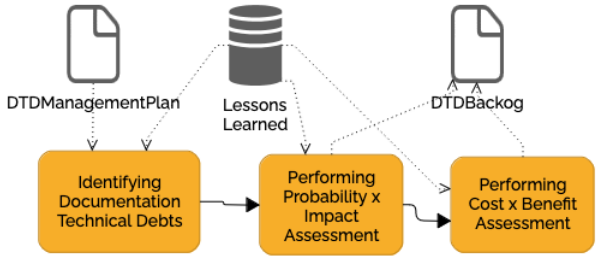

Figura 4. Phase Identifying and Measuring Documentation Technical Debts

The activity *Identifying Documentation Technical Debts* is performed by the Team members and the Mentor. They should assess whether (i) the expected documents are being generated, (ii) the content of the existing documentation is up to date, have the right amount of details, and is consistent, (iii) the documentation is adding value the way it is, and (iv) whether someone is using these documents. Possible causes for each identified DTD should also be identified. Data is logged into *DTDBacklog*.

The *DTDManagementPlan* should be consulted, as it lists the expected and used product documentation and possible causes and consequences of DTDs from *DOTED Guide*. Another best practice is to look for documentation of technical debts that occurred in other products, recorded in the organizational *Lessons Learned Base*. The sharing of opinions and personal experiences among the professionals involved in this activity is also crucial to help identify and measure the DTDs foreseen in the following activities.

Once the DTDs are identified, it becomes possible to measure them. *Performing Probability x Impact Assessment* activity aims to generate the *Criticality Matrix*.

The objective of the activity *Performing Cost x Benefit Assessment* is estimating the possible return on investment to resolve the identified DTDs. The perception of benefit is subjective, causing organizations to define their evaluation criteria. However, the method proposes a qualitative assessment based on the joint analysis of the degree of criticality resulting from the assessment of probability x debt impact (obtained from the *Criticality Matrix* and the cost of the debts, generating the *Benefit Removal Matrix*). It

should be noted that although the need to pay debts considered critical is evident since the cost of resolving this debt is high, the benefit obtained may be less advantageous. The objective of identifying and measuring technical debts is to facilitate decision-making on how to treat them [Seaman et al. 2012]. The choice of appropriate solutions and the decision regarding the technical documentation debts to be paid are carried out in the next phase.

4.2.3. Developing a Strategy for Resolving Documentation Technical Debts

Fig. 5 presents DOTED third phase. The purpose of identifying and measuring technical debts is to facilitate decision-making [Seaman et al. 2012]. Thus, once the debts are identified and measured, it becomes possible to prioritize and decide which ones should be paid. The main output of this phase is the creation of the *DTDResolutionPlan*. In addition, the *DTDBacklog* will be updated with the prioritization of the debts to be paid.

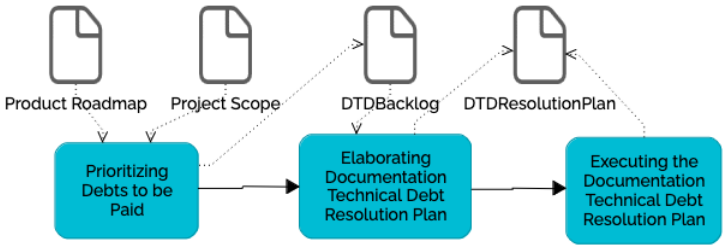

Figura 5. Developing a Strategy for Resolving Documentation Technical Debts

The objective of the activity *Prioritizing Debts to be Paid* is to define which DTDs should be paid as a priority and update the *DTDBacklog*. Prioritizing technical debt is challenging, as some debts may be essential to be paid for technical reasons, while others for commercial reasons [Yli-Huumo et al. 2016]. Thus, in addition to the *Mentor* and the *Team*, the participation of professionals from the organization's business area is also recommended. The *Product Roadmap* and *Project Scope* are inputs that provide crucial data to support the team during the prioritization process. The *Product Roadmap* presents what the product will look like at each period of its evolution and can be considered both a strategic document and a plan to execute the strategy. The *Project Scope* indicates the work required to be done on each product evolution project, including resource requirements and constraints and deadlines times.

To prioritize debts, the *Removal Benefit Matrix* (see Section 4.1.3) or another form of prioritization can be used, e.g., based on the organization's strategic objectives and the *Product Roadmap*. In this scenario, resolving debts that favor fulfilling the organizational strategic goals and objectives should be prioritized. Debts that can take a long time to pay off should also be considered.

Activity *Elaborating Documentation Technical Debt Resolution Plan* aims to create the *DTDResolutionPlan*, where the actions to resolve the DTDs are defined. Four tasks must be performed: (i) define the execution period for the actions; (ii) define the set of debts to be paid within this period (based on individual effort/time estimates for each

debt); (iii) identify the appropriate solution to resolve each debt within the established time-frame; and (iv) identify those responsible for resolving each debt. A new resolution cycle to execute the proposed solutions is given at this activity's end.

As a result of the activity *Executing a Documentation Technical Debt Resolution Plan*, the *DTDResolutionPlan* is updated. At the beginning of the resolution cycle, all debts are as *Pending*. When initiating the resolution of a debt, the debt manager should change the status to *In Progress*. At the end of the cycle, if the proposed solution has been successful, the debt should be set to *Resolved*. However, if the debt item cannot be paid within the predefined deadline, its status should be set to *Unresolved*. If a DTD has not been paid in one cycle, it is assigned to another cycle or considered *Accepted* by the team.

4.2.4. Monitoring Documentation Technical Debt

Fig. 6 presents DOTED fourth phase. This phase aims to share the results gathered by the execution of DOTED and update the documentation related to the product.

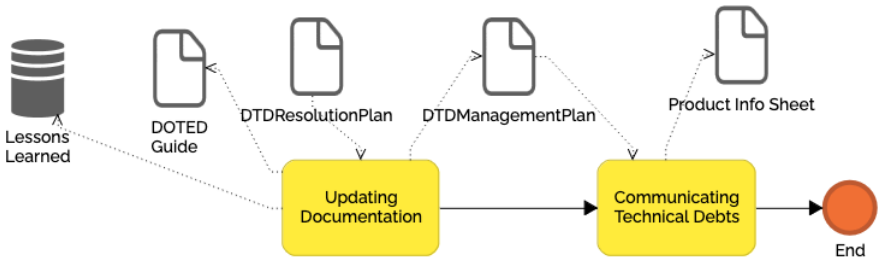

Figura 6. Monitoring Documentation Technical Debt

The activity *Updating Documentation* aims to update the documentation related to the product with the information obtained and generated during the execution of the method. The *DTDManagementPlan* must be updated with new information about causes, consequences, best practices, and documentation examples identified during the method execution and any changes in the product team. It is also considered extremely important to record lessons learned that may favor using DOTED in the future. The *DOTED Guide* must also be updated with information from the execution of the cycles.

The activity *Communicating Technical Debts* aims to reinforce the need to disseminate all the knowledge obtained during the DOTED execution. The importance of sharing the results obtained with the organization's other stakeholders in places such as forums, communication groups, and direct mail is highlighted. Likewise, updating the organizational information repositories, such as wikis and file repositories, is considered equally important.

4.3. DOTED Execution

DOTED execution must be iterative and cyclic. At first, we suggest executing phases *Creating or Reviewing the DTD Management Plan* and *Identifying and Measuring Do-*

cumentation Technical Debts. That way, the *DTDManagementPlan* and the *DTDBacklogLog* are created. The *DTDManagementPlan* should be revisited periodically to ensure it is adequate and consistent with the evolving strategy to develop and maintain the product. Whether the *DTDManagementPlan* should be incorporated into the project plan or is used independently depends on the team/organization. How the team is assigned tasks and new software versions are deployed might influence that decision.

The latter phase and the following two are iterative and executed in accordance with the organization's software development process. We suggest *Identifying and Measuring Documentation Technical Debts* phase and the first two activities of *Developing a Strategy for Resolving Documentation Technical Debt* to be executed aligned (or during) planning activities so the *DTDBacklogLog* are kept up-to-date and useful. For instance, it should be executed during the project planning activity in 'traditional' projects, during each sprint planning in agile projects, or periodically with the team using kanban. The activity *Executing the Documentation Technical Debt Resolution Plan* must be aligned with the development and maintenance activities so the strategy to pay DTD is put into motion.

The last phase *Monitoring Documentation Technical Debt* must be executed by the end of a DTD resolution cycle. It is supposed to increase the Teams perception of the importance of paying DTD debts and revisit the *DTDmanagementPlan* as needed. In Scrum projects, we suggest it aligns with Sprint Retrospectives.

5. DOTED Evaluation

We executed a participative case study [Baskerville 1997, Mills et al. 2010] to evaluate DOTED. As follows we present the study planning (Section 5.1), execution (Section 5.2), and results (Section 5.3).

5.1. Case Study Design

The *goal* of the participative case study [Baskerville 1997] was to evaluate DOTED use in a real context in the industry to support the adoption of documentation technical debt (DTD) management. The case study was executed in the software development sector of Organization A (name omitted due to confidentiality), a significant public health organization in Rio de Janeiro city. The study *context* was the team responsible for developing an educational product at an advanced stage of development. The product is intended to obtain and control Lattes curricula from the CNPq database, maintaining and controlling updates, searches, and access to curricula stored in the organization's database.

The study *participants* were a Project Manager, a Systems Analyst, and a Developer, members of the team involved in developing and maintaining a software product. All signed a "Term of Informed Consent Form."

The *procedure* planned for performing the study consisted of executing DOTED phases. The first author, acted as Mentor. He led the execution of the activities and the filling of templates part. The participants performed the activities and produced the envisaged information in all four phases of the method. The Mentor presented the method (including a description of activities and templates) to participants and basic concepts

about technical debt and technical debt documentation. A schedule for implementing the method was created with the team. It was suggested that the product team set aside at least one hour of its working day to carry out the study.

Data collection was performed in two moments: (i) doubts, reported problems, help requests, and general comments from the product team were registered during the method execution; and (ii) at the end of its execution, a predefined *questionnaire* was applied about the participants' perceptions regarding the method based on TAM (Technology Acceptance Model) [Davis 1989] dimensions: Perceived Usefulness, Perceived Ease of Use, and Self-Predicted Future Usage. Due to confidentiality issues, some data cannot be disclosed.

Data analysis was performed by the first author. *Data validation* was carried out by the second and third authors. We used all collected information to verify whether (i) DOTED produced what it was supposed to deliver and (ii) it could execute DOTED activities based on their descriptions.

5.2. Case Study Execution

The case study was carried out over four weeks. In the end, eleven follow-up meetings were held on different days. Organization A had no prior experience with DTD. At the case study beginning, the Mentor provided them with a short training on DTD concepts and the method. The Team was allocated part-time to maintain the product. We provided the participants with DOTED's description and templates in Word and Excel for documents *DOTED Guide*, *DTDManagementPlan*, *DTDBacklog*, and *DTDResolutionPlan*. The Team identified all DTDs independently and then discussed them with the Mentor. His interference was minimal, guiding them to follow the method and solve doubts. Although carrying out DOTED activities in groups is considered a best practice, the activities were conducted individually due to the participants' work commitments. The team performed only one execution cycle.

In the first phase *Creating or Reviewing the Documentation Technical Debt Management Plan*, the Mentor and the Project Manager in charge of the product started the activities *Eliciting Product Context* and *Defining the Team Responsible for Documentation Technical Debt Management*. The information necessary for the following activities, related to the product documentation, was provided by all study participants and updated in *DTDManagementPlan*. To obtain the information relevant to the activity *Identifying the Product Expected Documentation*, we used the organization's software development process as a reference. The use of only one documentation artifact not present in the organization's software process was identified: the "Microservices Request Model." During this activity, the team consulted the *DOTED Guide*, discussed its content, and also identified other causes, effects, and best practices that were integrated into the *DTDManagementPlan* and later to the *DOTED Guide*.

The *DTDBacklog* was created during the execution of the second phase *Identifying and Measuring Documentation Technical Debt*. Both Systems Analyst and Developer identified the DTD independently. During the DTD consolidation, priority was given to the Systems Analyst's assessments, as his technical knowledge was closer to the charac-

Tabela 1. *DTDBackLog* created during the case study

Technical Debt Item[1]	Type[2]	Source[3]	Benefit Matrix (P x I x C = B)[4]	Priority	Status
Lack of time to document (CA05)	TDC	A, D	M x H x M = H	1	Pending
Incomplete Vision Document (CA22)	TD	A	M x H x M = H	2	Assumed
Outdated Vision Document (CA04)	TD	A	M x M x M = M	3	Assumed
Lack of personnel to document (CA24)	TDC	A	H x H x H = H	4	Pending
Direct communication between developers and clients bypassing the procedures described in the software process (CA34)	TDC	A	H x H x L = H	5	Pending
No communication plan (ineffective communication) (CA03)	TD	A	H x H x M = H	6	Pending
Documentation processes are not followed completely (CA45)	TDC	D	H x M x H = M	7	Pending
Obsolete development process (CA44)	TDC	D	H x M x H = H	8	Pending
Inadequate communication due to dependency on outdated documentation from external sources (CA04)	TDC	A, D	H x H x H = H	9	Pending
Starting developing code without requirements and design artifacts (CA03)	TDC	D	H x H x H = H	10	Pending
Lack of test cases (CA03)	TD	A	H x H x M = H	11	Pending
Prioritizing delivering new features without documenting the changes (CA18, CA31)	TDC	D	H x M x H = H	12	Pending
Prioritization of deadlines over documentation (CA18)	TDC	A	H x H x H = H	13	Pending
Lack of standardization of code comments (CA43)	TD	D	L x L x M = H	14	Resolved

[1] The corresponding cause in DOTED Guide is shown in parenthesis. [2] TD = Technical Debt, TDC = Technical Debt Cause. [3] Who identified the item: A = Systems Analyst, D = Developer. [4] P = Probability, I = Impact, C = Cost, B = Benefit, L = Low, M = Medium = M, H = High.

teristics of these debts. To better identify and characterize the DTD affecting the product, participants consulted the DTD causes in the *DTDManagementPlan*. During the case study execution, we observed that technical debt items and their causes were not differentiated. Both causes and debts were added to the *DTDBacklog* in the same way, although specific solutions were defined for each case. Each identified debt was assigned a unique code to facilitate traceability and interoperability between artifacts. Participants also identified four new causes. Table 1 shows the five documentation technical debts (TD) and nine causes (TDC) identified.

To *Performing Probability x Impact Assessment* and *Performing Cost x Benefit Assessment* of the identified DTDs, in addition to personal criteria, such as technical knowledge and experiences in other projects, the participants used the measurement parameters proposed by the DOTED and the information collected in the *DTDManagementPlan*. Regarding evaluating costs, the person-hour ratio was adopted to measure the effort needed to be undertaken since the team was primarily composed of outsourced professionals and paid for service demand. For the evaluation of benefits, the *Criticality Matrix x Costs* was of great value to the team. Although the participants were based on personal criteria, the matrix corroborated the made assessment.

In the third phase *Elaborating Documentation Technical Debt Resolution Plan*, the team focused mainly on solving the identified and measured DTDs. The main priority criterion was based on the benefits that debt resolution could bring to the product. However, even though the participants acknowledged the criterion relevance, the priority

Tabela 2. *DTDResolutionPlan* Excerpt

Technical Debt (TD) or TD Cause (TDC)	Proposed Solution[1]	Responsibility	Effort[2]	Status
Lack of time to document (TDC)	Review and redefine project planning (BP09)	Analyst	(8; 0)	Pending
Lack of human resources to document (TDC)	Prioritize team tasks (BP11)	Analyst	(3; 0)	Pending
No communication plan (ineffective communication) (TD)	Create communication plan artifact (BP07, BP09)	Analyst	(3; 0)	Pending
Incomplete or out-of-date Vision Document (TD)	Revise artifact and include undocumented requirements (BP03)	Analyst	(26; 5)	Assumed
Lack of code comment standardization (TD)	Review the source code and standardize the use of comments (BP09, BP24, BP25)	Developer	(4; 4)	Fixed

[1] The corresponding best practice in *DOTED Guide* is shown in parenthesis. [2] Effort given in hours. Numbers in parenthesis represent estimated and actual values, respectively.

list was strongly influenced by the expected viability of resolving debts within the time available for conducting the case study. To help in the prioritization, participants analyzed the *DTD Benefit Removal Matrix*. Then, the team started the activity *Elaborating Documentation Technical Debt Resolution Plan*. In consensus, three DTDs were selected to be paid and two causes to be treated due to the limited time available for the execution of the case study. Table 2 shows a part of the created *DTDResolutionPlan*. The proposed solutions were suggested by the participants. They were influenced by the *DOTED Guide* best practices shown in parenthesis.

In the following activity *Executing the Documentation Technical Debt Resolution Plan* (Fig. 2), the proposed resolutions in *DTDResolutionPlan* were put into practice. Due to the limited time availability of study participants, only five DTD were selected to be paid.

In the fourth phase *Monitoring Documentation Technical Debt*, in the activity *Updating Documentation*, the DOTED artifacts were reviewed and updated, with the Mentor responsible for updating all documentation. The second activity of this phase (*Communicating Technical Debts*) was partially carried out. The Project Manager responsible for the product was periodically informed by the Mentor about the activities performed in each phase, and his approval was requested for all the information recorded during the study. Among other considerations, the Project Manager stated that the disclosure of DTDs could be done by email through a direct mail system created specifically for the product in question. The Systems Analyst reinforced this consideration when he emphasized that debts should be disclosed in communication channels of an exclusively institutional nature.

The correct execution of the *DTDResolutionPlan* should guarantee DTD is paid and the affected product documentation is updated. Nonetheless, we propose *Updating Documentation* activity as a safeguard to ensure the product documentation affected by DOTED execution is kept updated, consistent, and (at least) with no further unknown DTD. We also expect the next activity (*Communicating Technical Debts*) to enhance the perception of DTD management importance so the team gets more engaged in future DOTED cycles and regular development activities. We also believe that if DOTED use is

integrated into the team's everyday activities (e.g., during Sprint Plannings and Reviews) might create a positive feedback loop to reduce DTD.

5.3. Evaluation Results and Discussion

We base DOTED evaluation on two main sources: observation, feedback during and after the case study execution, and, as said in Section 3, based on the established requirements assessment.

5.3.1. Observation and Participants' Feedback

During the case study, we could gather important feedback and insights from the participants and how they executed the method. By the end of DOTED execution, we also ran a TAM-based questionnaire to collect their perceptions about it.

Although the entire content of DOTED was presented to the participants at the beginning of the case study, the demand for instructions for carrying out the activities and using the artifacts was great, especially during the method's initial phases. We noticed that many of the difficulties encountered were due to the participants not being so committed to reading the method description, seeking clarification directly with the Mentor. In addition, the team needed more solid knowledge of technical debt concepts. None of the problems, doubts, or comments collected were related to inappropriate activities for its phase or poorly structured. Instead, the participants gave positive reports regarding the structure of the method as a whole. Therefore, we concluded that DOTED activities were adequately defined and structured. Regardless, adjustments in the method's description and artifacts were made according to the feedback obtained.

For instance, a perceived limitation was that traceability and dependency were not explicitly considered. That information can help with TD evaluation. The *DTDBacklog* and *DTDResolutionPlan* were later evolved to register a unique identifier to the DTD and other related DTD. The *Benefit Removal Matrix* was evolved to show the values associated with dependent items as well. These improvements were implemented in the templates provided along with DOTED description and are replicated in the artifact's description in Section 4.1. We also updated *DOTED Guide* with causes, effects, and best practices identified during the case study execution.

To collect the participants' overall perception about DOTED, we aligned the use of open-ended questions to get more qualitative information on the method adoption and a TAM-based questionnaire [Davis 1989]. Table 3 presents the answers of participants Project Manager (P1), Systems Analyst (P2), and Developer (P3) to the TAM-based form questions [Davis 1989] concerning perceived usefulness (PU), perceived ease of use (PE), and self-predicted future use (IF). We used the Likert scale "Fully Agree" (FA), "Strongly Agree" (SA), "Partially Agree" (PA), "Partially Disagree" (PD), "Strongly Disagree" (SD), and "Fully Disagree" (FD). We align TAM with open-ended questions to better evaluate the method. So, the form also contained an open-ended question for each group of questions (i.e., PU, PE, and IF) in which participants were asked to explain the answers given. All answers were positive. They differed only in the level of agreement. No

answers of disagreement were given.

Tabela 3. Answers to the TAM-based Questionnaire

Type	Question	Answers
PU	Using DOTED makes the identification and measurement of documentation technical debt more efficient.	FA SA FA
PU	Using DOTED enables to resolve documentation technical debt more effectively.	FA SA FA
PU	Using DOTED artifacts effectively help the team during method execution.	SA SA FA
PU	Using DOTED increases the perception of documentation technical debt in the product.	FA FA SA
PU	Generally speaking, DOTED is useful for managing documentation technical debt.	FA FA SA
PE	DOTED activities and tasks can be carried out without difficulties.	SA SA FA
PE	I would find DOTED easy to use.	SA PA FA
IF	Assuming DOTED would be available on my job, I predict I will use it regularly in the future.	SA SA FA
IF	I would prefer using DOTED to the previous way I used to manage documentation technical debt.	SA SA FA

Regarding *Perceived Usefulness*, there was an agreement with the method's usefulness for managing the DTD. Participants also said, *"The experience with DOTED was useful for the process of identifying technical debts and measuring corrections"* (P1), *"Using DOTED made clear the DTD amount. If we had used it from the beginning, many of the debts would have been avoided or their impacts minimized"* (P3).

Concerning *Perceived Ease of Use*, the participants agreed that the method is easy to use, although some clarifications were needed during the execution of the method and that using some artifacts was not practical during the initial activities. However, participants commented that using the method may become more accessible as the team familiarity increases. When asked to comment on their answers, the participants said, *"The team found it easy to use DOTED, but it needed initial support"* (P1), *"Its use is not intuitive and requires guidance, whether verbal or written, so that the method can be efficient"* (P2) and *"Once the proposed mechanism is understood, the step by step ends up being organic and natural for the detection of technical debts"* (P3). It is noted, by the answers, the importance of the Mentor's presence to guide the execution of the method correctly.

According to *Intention of Future Use* answers, DOTED was well accepted by the participants and thus suggested its future use. Participants said, *"The way the method is presented helps a lot in identifying and resolving debts"* (P1), *"Using the method, when followed correctly, is the best way to achieve higher quality results"* (P2), and *"I believe that using the method from the beginning would bring more benefits to the project, as long as the team members embrace the idea and fully adopt the use of DOTED"* (P3).

5.3.2. Compliance with Established Requirements

As follows, we discuss DOTED support to DTD management and its use regarding the requirements (see Section 3) that led to its creation.

The structure of the phases and activities present in DOTED comprises identification, measurement, prioritization, monitoring, repayment, documentation, and communication (R1). Prevention support is accomplished by selecting a documentation technical

debt (DTD) cause instead of an existing technical debt to address in a DTD resolution cycle. Regarding the support DOTED provides to manage documentation technical debts (DTD) originated from any software lifecycle phase (R2) and during any software lifecycle phase (R3), the results are satisfactory. During the case study, we observed that the team could execute the method at an advanced stage of product development. All participants also reported that they could incur fewer DTDs if DOTED were executed at the beginning of the project.

We have evidence that DOTED artifacts (R4) supported the identification and registration of DTDs. Although the participants identified a need for improvement in some artifacts and the description of activities, the method proved flexible and fulfilled its objective. Moreover, the *DOTED Guide* content covers all lifecycle phases, directly supporting development teams from the requirements phase to software maintenance. It also helped the case study participants to identify and address DTDs (R5). Also, note that when commenting on the intention of future use, the participants indicated an intention to incorporate the method into the organization's software process.

5.3.3. Discussion

As explained in Section 3, our goal was to support organizations that develop software products to manage documentation technical debts. To that end, we developed DOTED. As mentioned before, the established requirements were considered one of the criteria for evaluating the proposed method. In addition, we assessed whether the method could effectively support the documentation of technical debt management. To this end, the viability and usefulness of the method were considered. DOTED should be considered feasible if it can achieve what it proposes and be used practically in the organization's reality. On the other hand, it would be considered helpful if it directly benefits the organization's DTD management. Based on the evaluations described before, DOTED fulfilled its purpose.

The questions related to Perceived Usefulness (PU) were adapted to capture whether DOTED supported DTD management properly (Table 3). The answers to all questions were positive. Moreover, we also considered observation and participants' feedback (Section 5.3.1) and the assessment of DOTED requirements based on the case study results (Section 5.3.2) to discuss our findings.

Organization A was chosen due to its interest in addressing its DTD. However, as the Project Manager could not oversee its execution, we chose to execute a participative case study due to the insights this method could provide. Baskerville [Baskerville 1997] states that participative case studies are a common and accepted scientific report proceeding from consulting projects, and there is strength in positing the findings of a participative case study for scientific readership. Also, according to Mills et al. [Mills et al. 2010], participatory case studies present a high-level commitment from the participants to participate and can provide researchers with greater insights into issues or problems by being presented with 'insiders' views and knowledge. Also, the authors argue that researchers can have greater confidence in interpreting the data since it is grounded in the partici-

pant's authentic experiences. In retrospect, after executing the case study, we found that to be true. The closer contact with the team using DOTED allowed us to react to the difficulties they faced timely. Moreover, we got their confidence along the way. Most process improvement actions (such as using a method for the first time) face resistance. To reduce such resistance, we put an extra effort into explaining how DOTED use could improve product development. Also, we provided an extensive description expecting to ease DOTED execution and provide a rationale for the needed decisions to deal with DTD management.

The team reported that as they used DOTED, their familiarity with it progressively increased, facilitating its use. The need for help to execute the method and use the artifacts occurred mainly during the initial activities. In addition, we realized the method execution was delayed because the team needed adequate knowledge about technical debt. Thus, we advise teams to be trained on executing the method and technical debt concepts to increase their capacity to identify technical debts and propose solutions to address them. In addition, organizations should seek to update the *DOTED Guide* with their examples over time to help identify and address the most common documentation technical debts in their products.

An important question arises regarding implementing DOTED in a complex software project. Some drawbacks can exist. For instance, identifying and estimating all DTD associated with the product can be very time-consuming, even if supported by the current version of *DOTED Guide* or one more robust. Therefore, we suggest it be used iteratively. We also found that DOTED may be more effective in fostering technical debt repayment if used early and continuously on the product's development lifecycle. Also, DTD identification might depend on the software engineering skills of the team and their knowledge of the product. Nonetheless, the DTD backlog and the effects of not paying it will only be addressed if the team makes the first step and acquire the culture of dealing with DTD accordingly. We expect DOTED can help with that. Moreover, to improve DOTED efficacy, the team should commit to executing it as described and, upon reflection, improve its integration into the development process to ease DTD management.

Section 2 presents related work and compares DOTED support to TD management activities with sources identified in cited mapping study [Li et al. 2015] and others. None of them focuses on DTD. Moreover, we could not find other methods focused on documentation TD. Nonetheless, the methods in [Li et al. 2015] inspired us to define DOTED phases, activities, and artifacts. We did not find much information on the challenges, sources, or effects of Documentation TD (available info on other DT types is easier to find) at the time. Therefore we executed the study in [Mendes et al. 2019] to create the *DOTED Guide* and get insights on how to deal with DTD. Later, it was used as a source for the Theoretical Framework of Documentation Debt in [Rios et al. 2020].

Nonetheless, this work has some limitations. The case study execution was shortened due to Organization A's internal issues. Therefore, not all identified DTD and DTD causes were selected to be paid. Yet, all DOTED activities and phases were executed accordingly and, most importantly, it allowed the team members to identify many documentation technical debt items and sources they were unaware of, thus promoting they

are addressed further. Considering that our objective with the case study was to evaluate DOTED usefulness and ease of use, the non-repayment of debts was not considered a compromising factor since evaluating the treatment of debts by the organization was not part of the scope of this work. We acknowledge that using psychometric measures such as TAM has limitations when answered by a few people. Nevertheless, it provided a valuable means to knowing how the method use and applicability are perceived in the organization when complemented by open-ended questions and observations. Although we created *DOTED Guide* to present causes, effects, and best practices covering all phases of a product lifecycle, it is still limited in content. We intend to expand its scope by executing a mapping study and including other DTD sources, such as the Theoretical Framework of Documentation Debt [Rios et al. 2020]. Also, note that the *DOTED Guide* items can be associated with phases other than what we point out. Organizations should review and adjust it according to their software process.

As follows, we discuss the threats to validity [Runeson et al. 2012]. The researcher was absent from any deliberation during the phases of the method, leaving the decision-making entirely at the discretion of the other participants. The description of the method, document templates, and case study report are available to enable replication of the study. To avoid difficulties in understanding DOTED and consequently impact the case study, the researcher trained the method with the participants, was available to answer questions from the participants whenever necessary, and reviewed the documentation produced for each performed activity. The participants evaluated the method description and its templates and evolved them from the case study. We combined open-ended questions to obtain more meaningful and broad feedback on using the method with a TAM-based questionnaire. The study was limited to only one organization and a single team, limiting its findings to this context. Also, DOTED execution took place during an actual project. The schedule pressured the team's ability to resolve all documentation technical debt identified.

6. Final Considerations

To answer the research question "How to support organizations that develop software products in the management of documentation technical debts?," we created DOTED, a method to support documentation technical debt management (DTD), comprising activities of identification, measurement, prioritization, and resolution. Additionally, the *DOTED Guide* displays possible causes, consequences, and best practices associated with documentation technical debt. We executed a case study to evaluate DOTED viability in supporting documentation technical debt management of a software product. The participants positively evaluated the method regarding their perception of the utility, ease of use, and intention of future use based on the Technology Acceptance Model (TAM). We improved the method description due to the participants' feedback. Moreover, the participants highlighted the benefits of DOTED use and its adequate support regarding its purpose.

Our research strategy used Design Science Research principles, a prominent approach in the Information Systems area. It allowed us to design an artifact (i.e., DOTED) iteratively, acquiring and creating knowledge along the way. Examples include

knowledge captured by empirical studies in a controlled and disciplined way, such as the ones that originated the *DOTED Guide* [Mendes et al. 2019] and the case study described here. Also, we could understand how technology (i.e., DOTED and its artifacts) influence practitioners and is perceived by them. One important finding from our experience is that technology must be adapted to the organizational culture, and it may be more effective if applied since the early phases of an information system lifecycle.

As future work, we foresee the implementation of the method in other organizations to collect more evidence of its applicability and identify new improvements. The *DOTED Guide* can also be extended to consider new causes, consequences, and best practices related to documentation technical debt. Moreover, it can be extended to support other types of technical debt. Besides, a tool that automates DOTED activities can facilitate its use and make it faster.

7. Acknowledgement

This work has been financially supported by UNIRIO (PPQ 03/2021) and FAPERJ (E-26/210.231/2021, E-26/211.437/2021).

Referências

[Alves et al. 2016] Alves, N. S., Mendes, T. S., de Mendonça, M. G., Spínola, R. O., Shull, F., and Seaman, C. (2016). Identification and management of technical debt: A systematic mapping study. *Information and Software Technology*, 70:100–121.

[Baskerville 1997] Baskerville, R. L. (1997). Distinguishing action research from participative case studies. *Journal of Systems and Information Technology*, 1(1):24–43.

[Besker et al. 2022] Besker, T., Martini, A., and Bosch, J. (2022). The use of incentives to promote technical debt management. *Information and Software Technology*, 142:106740.

[Bohnet and Döllner 2011] Bohnet, J. and Döllner, J. (2011). Monitoring code quality and development activity by software maps. In *Proceedings of the 2nd Workshop on Managing Technical Debt*, MTD '11, page 9–16, New York, NY, USA. Association for Computing Machinery.

[Bourque et al. 2014] Bourque, P., Fairley, R. E., and Society, I. C. (2014). *Guide to the Software Engineering Body of Knowledge (SWEBOK(R)): Version 3.0*. IEEE Computer Society Press, Washington, DC, USA, 3rd edition.

[Brown et al. 2010] Brown, N., Cai, Y., Guo, Y., Kazman, R., Kim, M., Kruchten, P., Lim, E., MacCormack, A., Nord, R., Ozkaya, I., Sangwan, R., Seaman, C., Sullivan, K., and Zazworka, N. (2010). Managing technical debt in software-reliant systems. In *Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research*, FoSER '10, page 47–52, New York, NY, USA. Association for Computing Machinery.

[Buschmann 2011] Buschmann, F. (2011). To pay or not to pay technical debt. *IEEE Software*, 28(6):29–31.

[Charalampidou et al. 2018] Charalampidou, S., Ampatzoglou, A., Chatzigeorgiou, A., and Tsiridis, N. (2018). Integrating traceability within the ide to prevent requirements

documentation debt. In *2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 421–428.

[Clements et al. 2003] Clements, P., Garlan, D., Little, R., Nord, R., and Stafford, J. (2003). Documenting software architectures: views and beyond. In *25th International Conference on Software Engineering, 2003. Proceedings.*, pages 740–741.

[Cruzes and Dyba 2011] Cruzes, D. S. and Dyba, T. (2011). Recommended Steps for Thematic Synthesis in Software Engineering. In *2011 International Symposium on Empirical Software Engineering and Measurement*, pages 275–284, Banff, AB, Canada. IEEE.

[Davis 1989] Davis, F. D. (1989). Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology. *MIS Quarterly*, 13(3):319.

[dos Santos et al. 2013] dos Santos, P. S. M., Varella, A., Dantas, C. R., and Borges, D. B. (2013). Visualizing and managing technical debt in agile development: An experience report. In Baumeister, H. and Weber, B., editors, *Agile Processes in Software Engineering and Extreme Programming*, pages 121–134, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Ernst et al. 2015] Ernst, N. A., Bellomo, S., Ozkaya, I., Nord, R. L., and Gorton, I. (2015). Measure it? manage it? ignore it? software practitioners and technical debt. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, ESEC/FSE 2015, page 50–60, New York, NY, USA. Association for Computing Machinery.

[Gat 2010] Gat, I. (2010). Revolution in software: Using technical debt techniques to govern the software development process. *Agile Product and Project Management, Cutter Consortium Executive Report*, 11(4).

[Green and Ledgard 2011] Green, R. and Ledgard, H. (2011). Coding guidelines: Finding the art in the science. *Commun. ACM*, 54(12):57–63.

[Guo and Seaman 2011] Guo, Y. and Seaman, C. (2011). A portfolio approach to technical debt management. In *Proceedings of the 2nd Workshop on Managing Technical Debt*, MTD '11, page 31–34, New York, NY, USA. Association for Computing Machinery.

[Hevner 2007] Hevner, A. R. (2007). The three cycle view of design science research. *Scandinavian Journal of Information Systems*, 19(2):87–92.

[Hevner et al. 2004] Hevner, A. R., March, S. T., Park, J., and Ram, S. (2004). Design science in information systems research. *MIS Quarterly*, 28(1):75–105.

[IEEE 1998] IEEE (1998). IEEE Standard for Software Maintenance. *IEEE Std 1219-1998*, pages 1–56.

[Letouzey 2012] Letouzey, J.-L. (2012). The sqale method for evaluating technical debt. In *2012 Third International Workshop on Managing Technical Debt (MTD)*, pages 31–36.

[Li et al. 2015] Li, Z., Avgeriou, P., and Liang, P. (2015). A systematic mapping study on technical debt and its management. *Journal of Systems and Software*, 101:193–220.

[Martini and Bosch 2015] Martini, A. and Bosch, J. (2015). The danger of architectural technical debt: Contagious debt and vicious circles. In *12th Working IEEE/IFIP Conference on Software Architecture*, pages 1–10.

[Melo et al. 2016] Melo, I., Santos, G., Serey, D. D., and Valente, M. T. (2016). Perceptions of 395 developers on software architecture's documentation and conformance. In *X Brazilian Symposium on Software Components, Architectures and Reuse (SBCARS)*, pages 81–90.

[Mendes et al. 2019] Mendes, L., Cerdeiral, C., and Santos, G. (2019). Documentation technical debt: A qualitative study in a software development organization. In *Proceedings of the 33rd Brazilian Symposium on Software Engineering*, SBES 2019, page 447–451, New York, NY, USA. Association for Computing Machinery.

[Michael Golden 2010] Michael Golden, J. (2010). Transformation patterns for curing the human causes of technical debt. *Cutter IT Journal*, 23(10):30.

[Mills et al. 2010] Mills, A., Durepos, G., and Wiebe, E. (2010). *Encyclopedia of Case Study Research*. Number v. 1 in Encyclopedia of Case Study Research. SAGE Publications.

[PMI 2017] PMI (2017). *A Guide To The Project Management Body Of Knowledge (PMBOK Guides)*. Project Management Institute, 6th edition.

[Power 2013] Power, K. (2013). Understanding the impact of technical debt on the capacity and velocity of teams and organizations: Viewing team and organization capacity as a portfolio of real options. In *2013 4th International Workshop on Managing Technical Debt (MTD)*, pages 28–31.

[Rios et al. 2020] Rios, N., Mendes, L., Cerdeiral, C., Magalhães, A. P. F., Perez, B., Correal, D., Astudillo, H., Seaman, C., Izurieta, C., Santos, G., and Oliveira Spínola, R. (2020). Hearing the voice of software practitioners on causes, effects, and practices to deal with documentation debt. In Madhavji, N., Pasquale, L., Ferrari, A., and Gnesi, S., editors, *Requirements Engineering: Foundation for Software Quality*, pages 55–70, Cham. Springer International Publishing.

[Rios et al. 2018] Rios, N., Spínola, R. O., Mendonça, M., and Seaman, C. (2018). The most common causes and effects of technical debt: First results from a global family of industrial surveys. In *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, ESEM '18, New York, NY, USA. Association for Computing Machinery.

[Runeson et al. 2012] Runeson, P., Höst, M., Rainer, A., and Regnell, B. (2012). *Case Study Research in Software Engineering: Guidelines and Examples*. John Wiley and Sons, New Jersey, USA, 1o edition.

[Sandberg et al. 2015] Sandberg, A., Staron, M., and Antinyan, V. (2015). Towards proactive management of technical debt by software metrics. In *CEUR Workshop Proceedings*, volume 1525.

[Seaman and Guo 2011] Seaman, C. and Guo, Y. (2011). Chapter 2 - Measuring and Monitoring Technical Debt. volume 82 of *Advances in Computers*, pages 25–46. Elsevier.

[Seaman et al. 2012] Seaman, C., Guo, Y., Zazworka, N., Shull, F., Izurieta, C., Cai, Y., and Vetrò, A. (2012). Using technical debt data in decision making: Potential decision approaches. In *2012 Third International Workshop on Managing Technical Debt (MTD)*, pages 45–48.

[Silva et al. 2019] Silva, V. M., Junior, H. J., and Travassos, G. H. (2019). A taste of the software industry perception of technical debt and its management in brazil. *Journal of Software Engineering Research and Development*, 7:1:1 – 1:16.

[Snipes et al. 2012] Snipes, W., Robinson, B., Guo, Y., and Seaman, C. (2012). Defining the decision factors for managing defects: A technical debt perspective. In *2012 Third International Workshop on Managing Technical Debt (MTD)*, pages 54–60.

[Soares et al. 2015] Soares, H. F., Alves, N. S., Mendes, T. S., Mendonça, M., and Spínola, R. O. (2015). Investigating the link between user stories and documentation debt on software projects. In *2015 12th International Conference on Information Technology - New Generations*, pages 385–390.

[Stochel et al. 2012] Stochel, M. G., Wawrowski, M. R., and Rabiej, M. (2012). Value-based technical debt model and its application. In *International Conference on Software Engineering Advances*, pages 205–2012.

[Wieringa 2014] Wieringa, R. (2014). *Design science methodology for information systems and software engineering*. Springer. 10.1007/978-3-662-43839-8.

[Yli-Huumo et al. 2016] Yli-Huumo, J., Maglyas, A., Smolander, K., Haller, J., and Törnroos, H. (2016). Developing processes to increase technical debt visibility and manageability – an action research study in industry. In Abrahamsson, P., Jedlitschka, A., Nguyen Duc, A., Felderer, M., Amasaki, S., and Mikkonen, T., editors, *Product-Focused Software Process Improvement*, pages 368–378, Cham. Springer International Publishing.

[Zazworka et al. 2011] Zazworka, N., Seaman, C., and Shull, F. (2011). Prioritizing design debt investment opportunities. In *Proceedings of the 2nd Workshop on Managing Technical Debt*, MTD '11, page 39–42, New York, NY, USA. Association for Computing Machinery.

[Zazworka et al. 2013] Zazworka, N., Spínola, R. O., Vetro', A., Shull, F., and Seaman, C. (2013). A case study on effectively identifying technical debt. In *Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering*, EASE '13, page 42–47, New York, NY, USA. Association for Computing Machinery.

[Zhi et al. 2015] Zhi, J., Garousi-Yusifoğlu, V., Sun, B., Garousi, G., Shahnewaz, S., and Ruhe, G. (2015). Cost, benefits and quality of software development documentation: A systematic mapping. *Journal of Systems and Software*, 99:175–198.

A. Appendix - DOTED Guide

DOTED Guide (as described in Section 4.1.1) presents possible DTD effects (Table 4) and causes (Table 5), and best practices to avoid them (Table 6). In all tables, we show the software lifecycle phase where each item is most likely to appear or cause an effect:

Requirements (R), Design (D), Construction (C), Testing (T), and Maintenance (M). We also indicate if the item is related to Project Management (G) activities.

Tabela 4. DOTED Guide - Effects

ID	Effects	Phases
EF01	Requirements do not match customer demands	- R D C T M
EF02	Tests are not performed efficiently or not effectively	- - - - T -
EF03	User dissatisfaction due to inconsistent documentation	- - - - T -
EF04	Users' real needs not met	- R D C T M
EF05	Difficulties in passing on knowledge to new team members	- - D C T M
EF06	Problems in software maintenance	- - - - - M
EF07	Rework in implementations and tests	- - - C T -
EF08	Difficulty of team members to have complete knowledge of the product being developed	- - D C T M
EF09	Inefficient testing due to lack of planning	- - - - T M
EF10	Ineffectiveness in the use of artifacts due to poorly organized information in their content	G R D C T M
EF11	Doubled effort in future to compensate for lack of prior documentation	G R D C T M
EF12	Unawareness of existing risks due to lack of documentation	G R D C T M
EF13	Activities execution inefficiency due to poor documentation	- - - C T M
EF14	Delays or impediments in creating documents due to insufficient resources	G R D C T M
EF15	Communication problems due to poor quality or lack of documentation	G R D C T M
EF16	Schedule delays due to inconsistent or non-existent documentation	G R D C T M
EF17	Cost increases due to inconsistent or non-existent documentation	G R D C T M
EF18	Project problems due to failure to validate documentation	G R D C T M
EF19	Inadequate understanding of the specified requirements	- R D C T M
FE20	Lack of perception of incompleteness or inconsistency of requirements	- R D C T M
FE21	Difficult to find where each feature has been implemented	- R - - - -
FE22	Increased effort for maintaining the product	- R - - - -
FE23	Omissions in requirements specification	- R - - - -
FE24	Inducing the error of incorrect coding	- - - C - -
FE25	Difficulty in consulting the user documentation	- R - - - M
FE26	Lack of credibility of users about the documentation presented	- - - - T -

Tabela 5. DOTED Guide - Causes

ID	Causes	Phases
CA01	Extensive documentation to be completed	G R D - T M
CA02	Lack of well-defined processes	G R D C T M
CA03	Missing documentation	G R D C T M
CA04	Outdated documentation	G R D C T M
CA05	Lack of time to document	G R D C T M
CA06	Inconsistent or inadequate documentation	G R D C T M
CA07	Unawareness of legal changes that affect the documentation	G R - - - M
CA08	Insufficient number of team members to document	G R D - T M
CA09	Selective maintenance of documentation	G R D C T M
CA10	Project with schedule delay	G R D C T -
CA11	High staff turnover	G R D C T M
CA12	Tacit knowledge not formalized on documents	G R D C T M
CA13	Redundant documentation	G R D - T M
CA14	Lack of adequate project planning	G - - - - -
CA15	Lack of technical capacity of the professional to document	G R D C T M
CA16	Negligence regarding documentation	G R D C T M
CA17	Documentation with poorly organized information	G R D C T M
CA18	Prioritization of deadlines over documentation	- R D C T M
CA19	Lack of understanding of the importance of documentation	G R D C T M
CA20	Change of management during the project	G R D C T M
CA21	Extraneous information in a document	G R D - T M
CA22	Incomplete documentation	G R D C T M
CA23	Lack of external resources to provide information needed for documentation	- R - - - -
CA24	Lack of resources to inspect documentation produced	G R D C T M
CA25	Political and hierarchical influences	G - - - - -
CA26	Lack of training on the process to be followed	G R D C T M
CA27	Not using the same documentation standard for new functionality development or existing functionality maintenance	- - - C T M
CA28	Poorly structured or non-existent comments in source code	- - - C - -
CA29	Poor time management	G R D C T M
CA30	Failure to adapt project documentation to information needs of specific roles	G R D C T M
CA31	Unrealistic commitments for releasing new features	G - - - - -
CA32	Conflicts between projects regarding staff allocation	G - - - - -
CA33	Lack of documentation of legacy systems	- R D C T M
CA34	The decision to follow the development process or not depending on individual initiative of the team members	- R D C T M
CA35	Replacement of information-rich document models (e.g. use cases and UML diagrams) by others focused on user interaction (e.g. prototypes)	- R D - - -
CA36	Use of artifacts originated from multiple sources or stakeholders	- R - - - -
CA37	Lack of standardization of artifacts	- R - - - -
CA38	The low priority given to design documentation activities	- - D - - -
CA39	Disbelief that design documentation can bring benefits	- - D - - -
CA40	Finding architecture documentation unnecessary in simple applications	- - D - - -
CA41	Redundant comments explaining obvious parts of the code	- - - C - M
CA42	Ambiguous code comments	- - - C - M
CA43	Lack of standardization of code comments	- - - C - -
CA44	Obsolete software development process	G R D C T M
CA45	Documentation processes are not followed	G R D C T M

Tabela 6. DOTED Guide - Best Practices

ID	Best Practices	Phases
BP01	Regularly update the artifacts used throughout the software life-cycle	- R D C T M
BP02	Comment source code	- - - C - -
BP03	Review/re-validate legacy or outdated documentation	- R D C T M
BP04	Produce documentation through the collaboration of different roles	- R D - T M
BP05	Use UML-based documentation for recording information and passing on knowledge	- - D C T -
BP06	Make team members aware of the problems caused by not adopting a proper documentation process	- R D C T M
BP07	Generate all the necessary documentation at the beginning of the project to avoid wasting future efforts	- R D C T M
BP08	Take personal initiative to generate your own documentation and share it with other team members	- R D C T M
BP09	Define documentation processes and templates	- R D C T M
BP10	Prioritizing debts to be paid according to the organization's current reality	- - D C - M
BP11	Define roles and responsibilities for managing documentation	- R D C T M
BP12	Peer-review of the documentation produced	- R D C T M
BP13	Create a documentation repository accessible to everyone in the organization	- R D C T M
BP14	Create tutorials on how to complete and use the documents	- R D C T M
BP15	Adopt stricter procedures for warning or punishment for non-compliance with the defined documentation process	- R D C T M
BP16	Use the organization's historical data to assist in the identification and measurement of technical debt	- R D C T M
BP17	Optimize information in a few documents	- R - - - -
BP18	Document only the necessary to meet the current needs of the organization	- R - - - -
BP19	Adopt tools to manage documentation	- R - - - -
BP20	Communicate identified debts, e.g., on wikis	- R D C T M
BP21	Write documentation from the reader's point of view	- - D - - -
BP22	Avoid unnecessary repetition of documentation	- R - - - -
BP23	Avoid ambiguities when writing notations	- R - - - -
BP24	Review the source code and its comments periodically	- - - C - -
BP25	Put a comment at the beginning of the code briefly stating its purpose	- - - C - -
BP26	Re-document procedures if necessary	G R D C T M