UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE CIÊNCIAS

DEPARTAMENTO DE COMPUTAÇÃO

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

MESTRADO ACADÊMICO EM CIÊNCIA DA COMPUTAÇÃO

BRUNO SABÓIA ARAGÃO

TESTDCAT: CATALOG OF TEST DEBT SUBTYPES AND MANAGEMENT ACTIVITIES

FORTALEZA

2019

BRUNO SABÓIA ARAGÃO

TESTDCAT: CATALOG OF TEST DEBT SUBTYPES AND MANAGEMENT ACTIVITIES

Dissertação apresentada ao Curso de Mestrado Acadêmico em Ciência da Computação do Programa de Pós-Graduação em Ciência da Computação do Centro de Ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em Ciência da Computação. Área de Concentração: Sistemas de Informação.

Orientadora: Prof. Dra. Rossana Maria de Castro Andrade.

FORTALEZA

2019

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

A671t Aragão, Bruno Sabóia.
TestDCat: Catalog of Test Debt Subtypes and Management Activities / Bruno Sabóia Aragão. – 2019.
110 f. : il. color.

Dissertação (mestrado) – Universidade Federal do Ceará, Centro de Ciências, Programa de Pós-Graduação em Ciência da Computação, Fortaleza, 2019.
Orientação: Profa. Dra. Rossana Maria de Castro Andrade.

1. Technical Debt. 2. Test Debt. 3. Testing Process. 4. Technical Debt management activity. I. Título.
CDD 005

BRUNO SABÓIA ARAGÃO

TESTDCAT: CATALOG OF TEST DEBT SUBTYPES AND MANAGEMENT ACTIVITIES

Dissertação apresentada ao Curso de Mestrado Acadêmico em Ciência da Computação do Programa de Pós-Graduação em Ciência da Computação do Centro de Ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em Ciência da Computação. Área de Concentração: Sistemas de Informação.

Aprovada em: 28/11/2019

BANCA EXAMINADORA

Prof. Dra. Rossana Maria de Castro Andrade (Orientadora)
Universidade Federal do Ceará (UFC)

Prof. Dr. Rodrigo Oliveira Spínola
Universidade Salvador (UNIFACS)

Prof. Dr. Joaquim Bento Cavalcante Neto
Universidade Federal do Ceará (UFC)

Prof. Dra. Valéria Lelli Leitão Dantas
Universidade Federal do Ceará (UFC)

À minha esposa, pelo apoio e incentivo incondicionais; ao meu filho, que trouxe uma nova inspiração; e à minha mãe, por ser meu alicerce e acreditar sempre na minha capacidade de superar desafios.

AGRADECIMENTOS

Gostaria de expressar minha profunda gratidão a todos que me apoiaram durante esta jornada do mestrado.

Primeiramente, à minha esposa, por seu apoio incondicional na decisão de iniciar o mestrado e por me acompanhar ao longo de todo o processo. Sua força, encorajamento e a revisão de meus textos e apresentações foram fundamentais para o meu sucesso. Sou imensamente grato por sua presença constante e por acreditar em mim.

Ao meu filho, que nasceu durante o período do mestrado. Sua presença na minha apresentação de defesa foi um momento único e inesquecível. Você trouxe uma nova perspectiva e inspiração à minha vida.

Ao meu pai, por todo o esforço para me proporcionar uma educação de qualidade, matriculando-me em bons colégios onde pude aprender com excelentes professores. Seu apoio foi essencial para minha formação e crescimento acadêmico.

Às minhas irmãs e ao meu cunhado, que também estão na área acadêmica e são verdadeiros exemplos para mim. Sua dedicação e conquistas me inspiram a buscar sempre mais e a acreditar no valor da educação.

À minha mãe, meu alicerce, que sempre me deu a base necessária para enfrentar os desafios e acreditar na minha capacidade de superá-los. Sua força e amor são inestimáveis.

À minha orientadora, Prof Dra. Rossana Maria de Castro Andrade que me acompanhou desde os primeiros semestres da universidade. Sua força, dedicação e comprometimento foram sempre um exemplo para mim. Sua competência me motivou a estudar e me dedicar para fazer um trabalho à altura de sua excelência. Sou grato por todo o seu apoio e orientação ao longo desta jornada.

Aos professores participantes da banca examinadora Prof. Dr. Rodrigo Oliveira Spínola, Prof. Dr. Joaquim Bento Cavalcante Neto, Prof. Dra. Valéria Lelli Leitão Dantas e Prof. Dr. Ismayle de Sousa Santos pelo tempo, pelas valiosas colaborações e sugestões.

E, claro, aos meus amigos, que me acompanharam nessa jornada. Sua companhia e apoio foram fundamentais para tornar este período mais leve e gratificante.

A todos vocês, meu sincero agradecimento.

"A pedra angular do estoicismo é aceitar o que não podemos mudar e focar no que podemos. Ao lidar com as adversidades, devemos nos concentrar nas ações que estão sob nosso controle e trabalhar para melhorar continuamente. Essa abordagem nos fortalece e nos permite enfrentar qualquer desafio com resiliência e determinação."

(Epicteto, 55 d.C., p. 12.)

RESUMO

Quando prazos e recursos de projetos de software se tornam escassos, testes geralmente são os mais impactados com suas atividades canceladas ou reduzidas. Se defeitos não puderem ser encontrados, a qualidade dos produtos pode ser afetada. Em um processo de desenvolvimento de software, atividades canceladas ou reduzidas que podem trazer benefícios a curto prazo, mas que podem ser prejudiciais ao projeto a longo prazo, são consideradas Dívidas Técnicas (DTs). Quando as DTs impactam as atividades de teste, elas são chamadas de Dívidas de Teste. Existem vários estudos que lidam com a Dívida de Teste, no entanto, essas soluções muitas vezes lidam com tipos específicos de testes (por exemplo, testes exploratórios e automatizados) e não abordam todo o processo de testes de software. Com o objetivo de preencher essas lacunas, este trabalho propõe um Catálogo de Dívidas de Teste com subtipos de Dívidas de Teste e atividades de gerenciamento de dívida técnica. Este catálogo foi construído com base nos resultados obtidos de um estudo empírico, uma revisão da literatura e entrevistas semiestruturadas conduzidas com profissionais que realizaram atividades de teste em cinco projetos da indústria. Para a avaliação do TestDCat, um estudo de caso foi realizado em projetos reais, a fim de identificar se o catálogo é de fácil utilização e se o seu uso ajuda o gerenciamento de dívidas de teste durante a execução das atividades de teste em um projeto de desenvolvimento de software. Os resultados obtidos na avaliação do estudo de caso apresentaram evidências de que o catálogo de dívidas técnicas de testes pode suportar o gerenciamento das mesmas e tem uma boa usabilidade.

Palavras-chave: dívida técnica; dívida de teste; processo de teste; atividade de gerenciamento de DT.

ABSTRACT

When deadlines and resources of software projects become scarce, testing is usually in the first row to have its activities aborted or reduced. If defects cannot be found, product quality can be affected. In a software development process, aborted or reduced activities that can bring short-term benefits, but can be harmful to the project in the long run, are considered Technical Debt (TD). When TDs impact testing activities, they are called Test Debt. There are several studies dealing with Test Debt, however, current solutions often deal with specific types of tests (e.g., exploratory and automated tests) and do not address the whole software testing process. Aiming to fill these gaps, this work proposes a Test Debt Catalog with subtypes of Test Debts and technical debt management activities. This catalog is built based on the results of an empirical study, a literature review and semi-structured interviews conducted with practitioners who perform testing activities on five projects from industry. For the TestDCat evaluation, a case study is conducted in real projects in order to identify if the catalog is user-friendly and if its use helps the test debt management during the execution of test activities in a software development project. The evaluation results obtained from the case study presented evidence that the information organized in the catalog can support the management of *Test Debts* and has good usability.

Keywords: technical debt; test debt; testing process; TD management activity.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

GREat	Group of Computer Networks, Software Engineering and Systems
GTF	GREat Test Factory
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
ISO	International Organization for Standardization
ISTQB	International Software Testing Qualifications Board
MR-MPS-SW	Brazilian Software Process Reference Model
TD	Technical Debt
TDM	Technical Debt Management
TP	Testing Process

SUMÁRIO

1 INTRODUCTION

This Chapter introduces the research subject studied in this work. Additionally, it summarizes the approach used to investigate and deal with the problem identified.

Section 1.1 contextualizes this master's thesis. Section 1.2 discusses the motivation behind the development of this work. Section 1.3 states the purpose and expected results of this work. Section 1.4 details the methodology and, finally, Section 1.5 presents the structure of the next chapters of this master's thesis.

1.1 Contextualization

Software Testing is one of the most commonly used approaches to evaluate software quality (ORSO; ROTHERMEL, 2014). In order to mitigate the risk of projects failing to perform tasks related to software testing, it is possible to make use of a well-defined testing process. By using a process, the software development team can monitor and control the activities as well as adjust them as required (ORSO; ROTHERMEL, 2014).

However, usually, tests are performed with limited resources, for example, time, people and financial resources (SLAUGHTER *et al.*, 1998). Moreover, when deadlines or resources become scarce, organizations tend to reduce even more tasks and practices related to software testing (WIKLUND *et al.*, 2017). Also, even using a testing process, members of a project may not perform (intentionally or unintentionally) some activities to achieve faster delivery and gain some competitive advantage (WIKLUND *et al.*, 2017). In the context of a Test Factory - independent organizations that can offer high-quality testing services at a lower cost (ANDRADE *et al.*, 2017b) (SANZ *et al.*, 2009) -, these decisions are even more critical as they can directly affect the quality of testing services offered to customers.

These kinds of technical commitments generated in software projects, that can bring short-term benefits, but which, in the long run, can be detrimental to project quality, are defined as *Technical Debts* (TDs) (LI *et al.*, 2015).

This concept was first used by Cunningham (CUNNINGHAM, 1992), who related the characterization of TD to problems in the code and the need for refactoring to pay the debts acquired. Other studies have addressed TDs in other activities of the software development process (*e.g.,* tests, requirements, usability) and provided solutions to manage them in software projects (LI *et al.*, 2015)(ALVES *et al.*, 2016). For example, *Technical Debt* that concerns the

software testing is known as *Test Debt* and arises when inadequate decisions regarding testing activities (*e.g.,* lack of tests, test estimation errors) are made (SAMARTHYAM *et al.*, 2017).

1.2 Motivation

Despite the growing number of work that deals with technical debts, Li et al. (LI *et al.*, 2015) and Alves et al. (ALVES *et al.*, 2016) highlighted the following points that still need the engagement of researchers and practitioners, such as:

- Most TD research is concentrated on a few types of technical debt (e.g., code and architecture), while other types of TD (e.g., tests, requirements, documentation) require further investigation.
- Research involving the identification of TDs focuses mainly on code-related debt. However, not identifying non-code-related TDs can be a risk, because this type of TD can also be detrimental to the project.
- Despite a growing number of proposals for approaches to managing TDs, few studies carry out case studies in the industry with these approaches, thus, for some types of TD, it is still not possible to understand the real impact and cost of using these management approaches.

Concerning test debt, although there is a significant number of work that deals with this type of debt (ALVES *et al.*, 2016), (LI *et al.*, 2015), most of them deal only with code-related debts by using, for example, tools for analysis of test code coverage and unit tests. Therefore, a more in-depth investigation of debts that cannot be managed by code analysis alone is necessary. In addition, most studies focus on only a few specific causes of test debts (for example, lack of automated tests (WIKLUND *et al.*, 2012) and exploratory tests (SHAH *et al.*, 2014)). Then, there is a need for studies covering a more significant number of causes of this type of debt.

Moreover, according to (SAMARTHYAM *et al.*, 2017), failure to identify and, consequently, manage test debts during software development can directly impact the quality of the software developed. In this context, it is necessary to investigate more deeply the causes of test debts and how they can be managed to control their costs and do not impact the maintenance/evolution of a software or make it unfeasible.

1.3 Objectives and Outcomes

The main goal of this research is to build a catalog with subtypes of test debts and management activities, which can be used to help in the management of test debts. This catalog should be appropriate to the reality of various organizations. Thus, proposed activities, techniques and artifacts should be able to adapt the activities, techniques and artifacts performed by each organization (including the existing testing processes in these organizations).

To achieve the objective of this research, first, an investigation of the causes of test debts and the management of technical debts is done. Next, approaches to dealing with test debt management are presented as well as suggestions to changes in a test process in order to systematize actions to manage test debts. This is important to identify and develop processes so that organizations can know which, how and when TDs should be paid (YLI-HUUMO *et al.*, 2016).

Thus, this work intends to fill the following gaps identified in the systematic maps (ALVES *et al.*, 2016) (LI *et al.*, 2015) and the literature review that is also done in this work:

- Lack of studies dealing with other types of technical debt (e.g., test debts)
- Identifying non-code-related TDs
- Few studies with more global approaches that cover more causes of test debts
- Need for validated solutions in projects in industry

As expected outcomes, the catalog is released for use by researchers and practitioners working with software testing. In addition, new causes (i.e. subtypes) of test debts related to the context of a test factory are identified and included in the catalog.

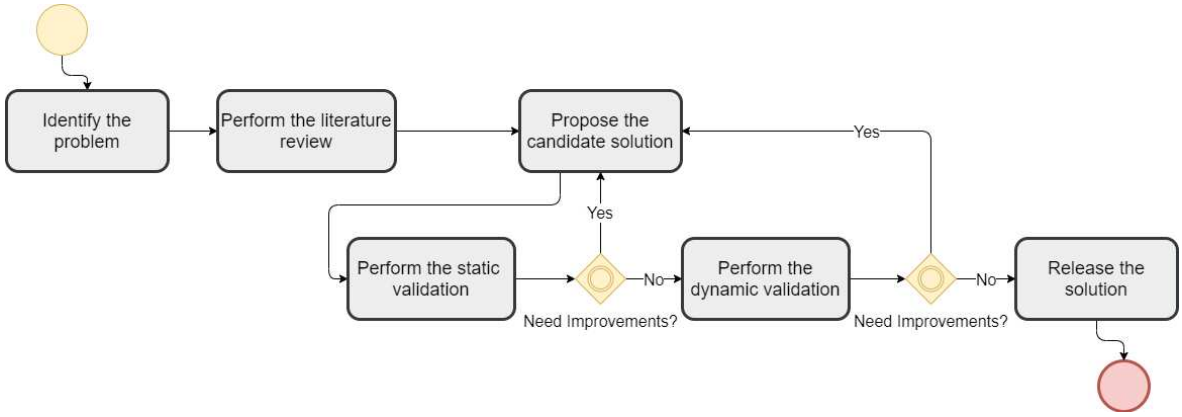
1.4 Research Methodology

The catalog is built following a methodology partially based on the technology transfer model presented by Gorschek (GORSCHEK *et al.*, 2006). This model favors cooperation between academia and industry and can be beneficial to both. It gives researchers the opportunity to study relevant industry issues and validate their results in a real environment. In return, professionals receive knowledge about new technologies that can, for example, optimize their processes.

This work methodology is presented in Figure 1 and detailed as follows. Regarding the first step of the methodology, the problem identification comes from the experience in

a successful long-term partnership with the industry (ANDRADE *et al.*, 2017a) in Research, Development and Innovation (R&D&I) projects that occur in the environment where this master's thesis is developed. In this kind of projects, the GREat[1] test factory team has followed a Testing Process (TP) as documented in Andrade et al. (ANDRADE *et al.*, 2017b). Also, in this step, an empirical study is conducted (Chapter 4) to identify the main issues faced on the GREat Test Factory (the results of this empirical study are also presented in the article (ARAGAO *et al.*, 2019)).

Figura 1 – Methodology



Fonte: Based on (GORSCHEK *et al.*, 2006)

As a result of the first step, the problem was identified and formulated as Test Debt. The second step of the methodology starts with a literature review (details in Chapter 3). This stage of the methodology seeks to deepen the knowledge on a test debt and identify how the related work deal with it. This review was performed based on the analysis of test debt studies identified in the systematic mappings carried out by Li et al. (LI *et al.*, 2015) and Alves et al. (ALVES *et al.*, 2016). From the analyzed work, based on the principle of Snowballing (WOHLIN, 2014), it was performed a search for studies that cited the set of previously selected papers.

The next step is to formulate and propose candidate solutions to tackle the identified problem. In this master's thesis, two candidate versions are presented. The first version is developed from three distinct and complementary sources: (i) Literature review; (ii) Empirical study; and (iii) Results of semi-structured interviews with professionals in the test area. The knowledge acquired from these three sources is organized in a catalog format. The second

[1] The Group of Computer Networks, Software Engineering and Systems (GREat) works on research and development software projects, developing web and mobile tools that are continually being tested by the GREat Test Factory team.

version of the candidate solution is obtained from the result of the evaluation performed in the first version of the catalog and the information collected in a new series of semi-structured interviews conducted with software developers who also performed testing activities.

The next step of the methodology refers to the static validation. For this validation, a survey is applied to evaluate the first version of the catalog, and a focus group (KRUEGER; CASEY, 2002) is conducted to evaluate the second version.

After performing the static validations, the next step of the methodology, dynamic validation, is started. For this validation, a case study (RUNESON; HÖST, 2009) is performed with industry projects at the GREat Test Factory.

As the final stage of the methodology, the final version of the catalog is presented. This version is suitable for use in the industry in order to help professionals to solve the problem previously identified.

1.5 Document Organization

This chapter presented the issues that motivated this work. Moreover, it was presented the objectives and expected results with this research as well as the research methodology followed to achieve the expected results.

The remainder of the document is organized as follows:

Chapter 2 presents the theoretical background related to this master's thesis. The concepts of test and test process are presented as well as the concept of technical debt, presenting the types and subtypes of technical debt, and activities and tools for managing these debts. The main focus is given to test debts, the subject of this work.

Chapter 3 describes a literature review comprising studies related to research with test debts and how to manage them. In relation to the core of this master's thesis, this chapter presents papers that analyze test debts, either in a specific subtype or in a general manner, and ways in which these debts can be managed. A comparison between the related work is also presented with the main aspects that will be tackled in this work.

Chapter 4 presents the empirical study conducted at the GREat Test Factory (GTF). As a result of this study, test debts, which occurred in the GTF, are identified. In addition, lessons learned and approaches are also used to create the catalog of test debts, focus of this master's thesis.

Chapter 5 describes the steps for building the catalog. It details the activities and

presents the results achieved during the construction of the catalog. The final built catalog is also presented, detailing its structure and describing some examples.

Chapter 6 presents a case study conducted in real projects in order to identify if the catalog is user-friendly and if its use helps the test debt management during the execution of test activities in a software development project.

Finally, **Chapter 7** describes the results achieved as well as the conclusions of this work and makes suggestions for future work.

2 BACKGROUND

In this chapter, the concepts that compose the theoretical basis of this work are presented. Section 2.1 presents the software test definitions with the main focus on the test process. Section 2.2 describes the concept of Technical Debt (TD) and their types. Section 2.3 presents the concept of test debt, which is the type of technical debt focus of this dissertation, and its causes. Section 2.4 describes ways of managing technical debt. Finally, Section 2.5 concludes this chapter.

2.1 Software Testing

Over the years, several researchers and organizations have presented software test definitions. According to Myers (MYERS *et al.*, 2011), software testing is a process or a set of processes that should verify that software does what it is designed to do. Hass (HASS, 2014) presents several definitions of software testing taken from standards, among them: (i) "The process consisting of all lifecycle activities, both static and dynamic, concerned with planning, preparation and evaluation of software products and related work products to determine that they satisfy specified requirements, to demonstrate that they are fit for purpose and to detect defects" (ISTQB, 2012)[1]; and (ii) "Set of activities performed to facilitate the discovery and/or evaluation of properties of one or more test items"(ISO/IEC29119-1, 2013)[2].

From these definitions, two important characteristics of software testing can be notice: (i) testing is directly related to quality; and (ii) testing is a process with a set of activities whose purpose is to test a software.

2.1.1 Test Process

Testing processes are commonly used with the aim of systematizing testing activities. These processes can vary according to the needs of each organization, but there are those that are generic and can be used in organizations with some adaptations, if necessary. In the context of this work, we will consider the testing process defined by the GREat Test Factory (GTF) and the process presented by Mette and Hass (METTE; HASS, 2008), that was used as a basis for the

[1] International Software Testing Qualifications Board (ISTQB), an international entity that offers a certification structure in software testing

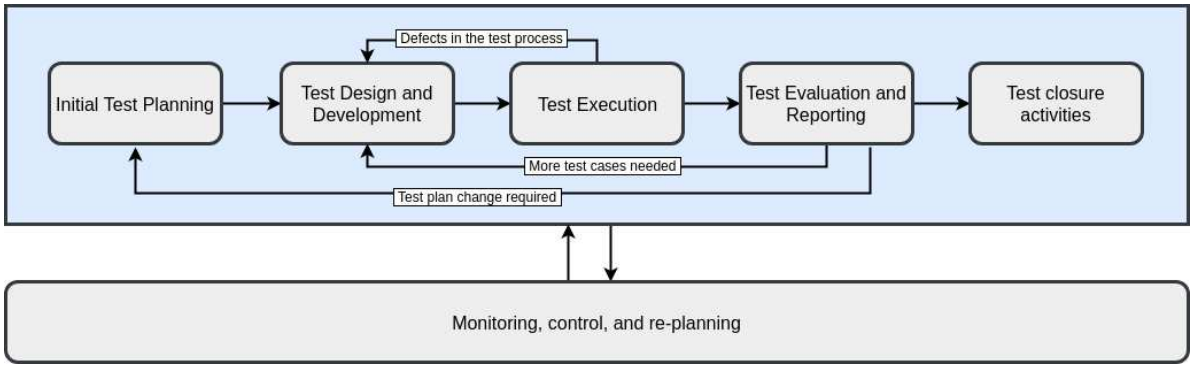
[2] International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE) 29119 Software Testing is a set of internationally recognized standards for software testing

process presented in ISO 29119 (ISO/IEC29119-1, 2013) and is adopted by ISTQB.

2.1.1.1 ISTQB Test Process

Figure 2 presents the steps of the test process proposed by Mette and Hass (METTE; HASS, 2008), adopted by the ISTQB. The process is iterative and does not necessarily need to follow a sequential order. Each activity can be divided into sub-processes with inputs, activities and outputs.

Figura 2 – Generic testing process adapted from Mette and Hass



Fonte: Based on (METTE; HASS, 2008).

The generic testing process proposed by Mette and Hass consists of the following six steps:

1. **Initial Test Planning:** It consists of verifying the purpose of the tests and taking the necessary actions to transform the test strategy into an operational plan.
2. **Test Design and Development:** Its purpose is to design the tests in a way that satisfies the requirements of the test plan. It also includes the definition of the test environment.
3. **Test Execution:** It consists of performing the test and recording the execution and results.
4. **Test Evaluation and Reporting:** It assures that the objective of the test has been achieved and communicates the results obtained in a way that is understandable and useful to the stakeholders.
5. **Test closure activities:** It aims to complete the activities, document lessons learned, ensure assets and information in the organization, and communicate with stakeholders about the completion of the activities.
6. **Monitoring, control, and re-planning:** It is intended to keep track of the acti-

vities performed and to make corrections when necessary.

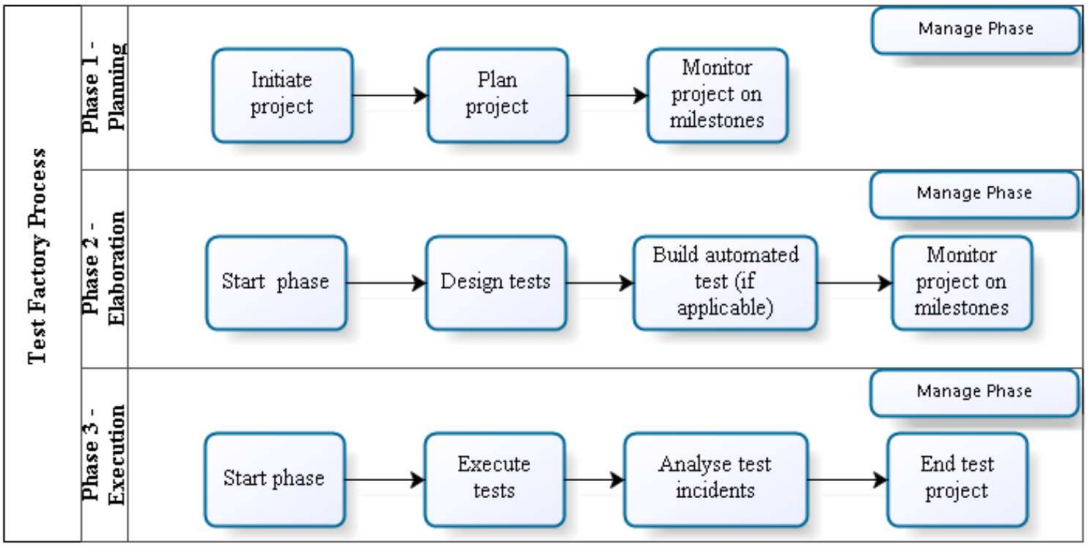
According to Hass (HASS, 2014), the main difference between the generic testing process adopted by ISTQB and the process model presented by ISO 29119 (ISO/IEC29119-2, 2013) is the organizational layer presented in ISO. This layer produces and maintains the organizational test specification. It includes the following activities: (i) Develop an organizational test specification; (ii) Monitor and control the use of the organizational test specification; and (iii) Update the organizational test specification. As a result, the test policy and the organizational test strategy are obtained.

The organizational testing process should be continuous and should ensure that the specification is aligned with the organization's business objectives.

2.1.1.2 GREat Test Factory Testing Process

Figure 3 presents the phases and activities of the GTF testing process (ANDRADE *et al.*, 2017b).

Figura 3 – GTF Test Process (ANDRADE *et al.*, 2017b)



Fonte: Andrade et al. (ANDRADE *et al.*, 2017b)

The GTF testing process consists, in summary, of three major phases:

1. **Planning:** Its purpose is to plan the testing of a particular demand. In the planning phase, the types of tests to be carried out, the schedule to be followed and the main risks for the demand in question are identified. The main activities of this phase are: (i) Start project; (ii) Plan project; and (iii) Monitor project at specified milestones.

2. **Elaboration:** In this phase, scenarios and test cases are prepared for the test demand. It is also planned to review the tests developed. This phase develops the inputs for the Execution. The main activities are: (i) Start phase; (ii) Design tests; (iii) Create automated test scripts (if applicable); and (iv) Monitor the project at the specified milestones.
3. **Execution:** It aims to perform the tests previously prepared and report the results obtained. In this phase, the test project is finalized with the documentation of the lessons learned. The main activities are: (i) Start phase; (ii) Execute tests; (iii) Analyze test incidents; and (iv) Finalize test project.

Unlike the Mette and Hass process, the GTF process is adherent to the Brazilian Software Process Reference Model (MR-MPS-SW) (ROCHA *et al.*, 2005) and is independent of the software development process. In addition, the GTF process has divided the testing activities into only three steps, while the generic process divides these activities into more steps.

2.2 Technical Debt

Technical Debt can be seen as a way to characterize the gap between the current state of a system and hypotheses of its "ideal" state (BROWN *et al.*, 2010). The term technical debt first appeared in Cunningham's work (CUNNINGHAM, 1992). In this paper, the author related TDs only to problems related to the code and he emphasized that TDs, as long as they were few and controlled, could accelerate the development. However, when accumulated, they could paralyze a project.

Originally TD was related to poor design or implementation, but this concept was expanded to characterize any immature artifacts developed during the software development life cycle that caused higher costs and lower quality (GUO *et al.*, 2016). Thus, TDs were categorized into various types according to the software development activity (e.g., requirements TD, Code TD and Test TD).

TDs can be classified into various types, depending on their cause within the software development lifecycle. In the works of Li et al. (LI *et al.*, 2015) and Alves et al. (ALVES *et al.*, 2016), some of the types of TDs identified in the literature were mapped and summarized in Table 1. The focus of this work is on TDs related to software testing.

Tabela 1 – TD Types

TD Type	Description
Requirements TD	Distance between optimal specification and actual application.
Architectural TD	Inadequate solutions regarding technologies and architecture standards.
Design TD	Shortcuts and deficiencies in the initial project that do not include quality aspects.
Code TD	Poorly written codes that do not follow development best practices.
Test TD	Shortcuts taken in tests (e.g., lack of integration and acceptance tests).
Build TD	Failure in the build process that can make it more complex.
Documentation TD	Insufficient, incomplete or outdated documentation.
Infrastructure TD	Configurations of technologies and support tools other than the appropriate ones.
Versioning TD	Problems with code version control (e.g., unnecessary branches).
Defect TD	Refers to defects, bugs and faults detected in the software product.
People TD	Problem with people who can impact the system (e.g., concentrated expertise).
Usability TD	Inappropriate usability decisions that will need to be adjusted at a later date.
Process TD	Definition of inefficient or inappropriate processes.
Service TD	Use of inappropriate web services that generate incompatibility of the service with the functional requirements of the applications.

Fonte: Adapted from (LI *et al.*, 2015) (ALVES *et al.*, 2016)

2.3 Test Debt

As stated in Section 2.2, TDs can be classified into several types. Technical debts related to software testing are known as *Test Debts*. They occur when inadequate decisions are made regarding testing activities (*e.g.,* non-implementation of unit tests, inadequate test coverage, incomplete test specification) (SAMARTHYAM *et al.*, 2017).

Technical test debts are usually caused by the following factors (SAMARTHYAM *et al.*, 2017) (SOUSA, 2016) (WIKLUND *et al.*, 2012): (i) pressure to deliver the product; (ii) inexperienced professionals; (iii) obsession with numbers (*e.g.,* team focuses on the test coverage indicators to be achieved and not on the quality of the tests developed); (iv) the use of inadequate testing tools; (v) poorly elaborated test plans, for example, with estimation errors in testing activities; (vii) lack of financial resources; and (viii) problems with test documentation.

The systematic mapping presented in the work of Li et al (LI *et al.*, 2015) categorize the test debts in the following subtypes causes:

a) **Low code coverage:** The system has a target range of coverage and that not reached during the release.

b) **Deferring tests:** Tests are postponed and leave the scope of the release that will be tested.

c) **Lack of tests:** Tests are not performed for a particular release.

d) **Lack of automated tests:** The automated tests of the functionalities that should

have these tests were not developed or executed for the release.

e) **Defects not found in tests:** Defects are not identified at the test time of the release. These are defects identified by the customer or end user.

f) **Expensive tests:** The tests identified are very complex or require a lot of effort for their documentation and performance.

g) **Test estimation errors:** Estimation errors are identified in the testing activities performed.

Due to the importance of software testing to evaluate and improve the quality of the software developed (ORSO; ROTHERMEL, 2014), failure to identify and manage test debts during the software development can directly impact on the quality of the software developed (SAMARTHYAM *et al.*, 2017).

Test Debts and all types of TDs must be managed in order to keep their cost under control and avoid possible damage to the developed software.

2.4 Technical debt management

TDs can be acquired intentionally when expected to increase the productivity of software development in the short term. However, they can be caused unintentionally, meaning that the team does not know about their existence, location and consequence. Both intentional and unintentional TDs must be managed in order to keep their cost under control (LIM *et al.*, 2012).

2.4.1 TD Management Activities

Li et al. (LI *et al.*, 2015) have identified in their systematic mapping a set of eight activities that are typically performed for Technical Debt Management (TDM), namely: (i) **Prevention** TDs from occurring; (ii) **Identification** of TDs acquired intentionally or not; (iii) **Measurement** of the cost/benefit of repayment a TD; (iv) **Prioritization** of TDs to assist in deciding which one should be paid first; (v) **Monitoring** changes in cost/benefit values; (vi) **Repayment**, which means resolving or mitigating TDs; (vii) **Documentation** to provide ways to document existing TDs; and (viii) **Communication** on passing on the TDs identified to those involved.

For each TD management activity, different approaches can be used. Li et al. (LI *et*

al., 2015) also summarizes the most commonly used approaches in the literature according to each management activity.

Regarding the **Prevention** management activity, with the approach of *Development process improvement*, the existing process is improved in order to avoid the occurrence of certain TDs. By the *Architecture decision making support* architectural design options with less potential TDs are chosen.Through the *Lifecycle cost planning*, it is developed a cost plan that monitors the entire software lifecycle cost to minimize the total TD of the system. And using the approach of *Human factors analysis* a culture that minimizes unintended TD caused by human factors (e.g., indifference and ignorance) is cultivated.

The **Identification** management activity brings the *Code analysis* approach, which consists of an examination of the code in order to check for coding rule violations and lack of testing. This approach also identifies design or architecture problems by calculating metrics from the source code. Also in Identification management activity, there is the approach of *Dependency analysis*, that analyzes dependencies between software elements (e.g., components, modules) to detect potential debts. By the *Check List* approach, a list is created in order to check predefined scenarios for the existence of TD. Lastly, through the approach of *Solution comparison*, it is compare the current solution with the optimal solution in several aspects (e.g., cost/benefit ratio). TD is acquired if the adopted solution is not ideal.

The management activity of **Measurement** presents the approaches of *Calculation model*, through which mathematical formulas or models are used in order to calculate the TD; *Code metrics*, that brings the use of source code metrics to calculate the TD; *Human estimation*, through which manual estimates based on the experience of team members are made; *Cost categorization*, an approach used to estimate the various types of cost of treatment for the incurred TD; *Operational metrics*, that bring the use of operational product quality metrics to estimate TD; and *Solution comparison*, used to calculate the difference from the current solution to the optimal solution.

Regarding the **Prioritization** management activity, with the approach of *Cost/benefit analysis*, the cost/benefit ratio of TD's repayment is analyzed. TDs for which the benefit of repaying is greater than their cost must be repayed first. On the other hand, with the *High remediation cost first* approach the prioritized TDs are the higher-cost ones. Through the *Portfolio approach* the TDs, the new functionalities and the bugs are considered as assets (with risks and investment opportunities). Thus, it is possible to select the set of assets that can

maximize the return or minimize the investment risk. Finally, with the approach of *High interest first*, the TDs prioritized are the ones with higher interest (possible penalty that must be paid in the future as a result of debt acquired in the present) rates.

The management activity of **Monitoring** presents five approaches. *Threshold-based approach* is used to set limits for quality metrics related to TDs and show warnings if limits are reached. Through *TD propagation tracking* it is used dependency analysis to check the influence of TD identified in other parts of the system that have TD. By *Planned check*, the identified TD is regularly checked to find possible changes. The approach of *TD monitoring with quality attribute focus* is useful for Monitoring the change in quality attributes that can deteriorate the status of TD. The fifth approach of this activity is *TD plot*, used to plot on the graph various aggregate measurements of the TD over time and observe the shape of the graph to identify trends.

The **Repayment** management activity brings the *Refactoring* approach, through which changes in code, design, or architecture of a software system are made to improve internal quality without changing current behaviors. The approach of *Rewriting* is used in order to rewrite the code that contains TD. *Automation* is an approach used to automate repeated manual labor (e.g., manual testing, manual deployment). Through *Reengineering*, the existing software is evolved to display new features and improve operational quality. *Repackaging* is an approach used to group cohesive modules together to simplify dependency between them (improving maintainability). *Bug fixing* is useful to correct identified bugs. And *Fault tolerance* is an approach that allows runtime exceptions to be strategically placed where the TD is.

The management activity of **Documentation** has only one approach, which is *Document format TD items*, useful to document each TD with a detailed description, including various fields (e.g., ID, location, responsible, type).

The last management activity is **Communication**. *TD Dashboard* is one of its approaches, and it is used to create dashboards presenting the identified TDs, their types and costs, in order to inform all stakeholders of their existence. *Backlog* is used to place identified TDs in the backlog of the project, along with all the tasks that must be developed. The TDs must have the same importance as the other tasks of the project. The approach of *Dependency visualization* allows the visualization of unwanted dependencies between software elements (e.g., components and packages). Through the *Code metrics visualization*, it is possible to view tool-generated code metrics and highlight elements whose measured quality is poor. The approach of *TD list* is used to maintain a list of identified TDs and make it visible to stakeholders.

And the *TD propagation visualization* is an approach the allows to present the links between the identified TDs and how each TD may affect the others.

2.4.2 TD Management Strategies

In addition to activities for managing TDs, strategies can also be used to support decision making on TD management.

In the systematic mapping presented by Alves et al. (ALVES *et al.*, 2016), strategies for managing TD were presented, according to Table 2.

Tabela 2 – Strategies for managing technical debt

Strategies	Description
Cost-Benefit Analysis	This strategy is to evaluate whether the interest for the repayment of the debt is high enough to justify it. To calculate the interest you must take into account the probability of the interest and its value.
Portfolio Approach	From the list of TDs identified by projects and a set of information regarding each TD (e.g., TD location, estimate of the principal, estimates of the expected interest amount and interest standart deviation (ISD), and estimates of the correlations of this DT with other DTs) an analysis is made of which TDs should be paid and which can be postponed.
Options	The investment in the repayment of the debt is similar to the purchase of the option that facilitates change to the software in the future.
Analytic Hierarchy Process (AHP)	AHP provides a method for structuring a problem, compare alternatives with respect to specified criteria, and determining a general classification. With regard to the management of TDs, the result of the strategy would be a prioritised ranking of TDs, indicating which one should be first repaid.
Calculation of TD Principal	The strategy follows a defined process to estimate the TD Principal (i.e. cost to pay the debt) and to associate the identified issues with different quality attributes.
Marking of Dependencies and Code Issues	Insert tags into the source code to mark dependencies that can cause TD, so that the development team can see where TD is inserted and thus decide when to pay based on, for example, the effort involved.

Fonte: Adapted from (ALVES *et al.*, 2016)

Some of the strategies mentioned in the mapping are also mentioned in the prioritization activity presented by Li et al. (i.e., cost-benefit analysis and the portfolio approach). The main difference between the approaches and the strategies is that the approaches take into account that the TD will already be paid, while in the strategies they provide support for decision making regarding the payment or not.

2.4.3 TD Management Tools

In addition to TD management activities and strategies, some studies also present tools that can assist in this management. So, Li et al. (LI *et al.*, 2015) also present tools identified in the literature that can be used to perform technical debt management activities. In summary, 29 tools were identified, most of which (19) deal with management of code TDs (e.g., Sonar TD plugin, CodeVizard and FindBugs). The other types of TDs that most have tools for TDM is the Design TD (e.g., Stan, iPlasma and Eclipse Metrics) and test TD (e.g., NCover and Coverage).

Another important point to be mentioned is about the number of tools per TDM activity. In this aspect, most of the tools work directly with TD identification activities (25). In this aspect, no tool works with TD prevention.

2.5 Conclusion

This chapter presented the basic concepts of software testing and testing process. Regarding the testing process, the generic testing process provided by Mette and Hass (METTE; HASS, 2008) and the testing process adopted by the GREat testing factory (ANDRADE *et al.*, 2017b) were presented.

Another subject directly related to the work was the concept of technical debt. The definition of the concept was introduced according to the literature and details on the management of technical debts were presented, as well as the demonstration of activities, strategies, and tools that can be used for this purpose.

It was also presented the type of technical debt related to software testing (i.e., Test Debt), the focus of this master's thesis. In addition to the concept related to this type of TD, the subtypes of test debt were also presented.

3 RELATED WORK

In this chapter, the related work to the theme of this dissertation is presented and discussed. Section 3.1 presents the research procedure used to identify the related works. Sections 3.2, 3.3, 3.4 and 3.5 present the related work found. Section 3.6 presents a comparison between these works. Finally, Section 3.7 concludes this chapter.

3.1 Research procedure

Following the proposed methodology (Chapter 1, Section 1.4), in order to improve knowledge about the concept of Technical Debt and how to manage it, a literature search was conducted through systematic reviews or systematic mapping of Technical Debt. Alves et al. (ALVES *et al.*, 2016) and Li et al. (LI *et al.*, 2015) were identified and highlighted for presenting a taxonomy regarding the types of technical debt, as well as activities and strategies for managing these debts, besides being works widely cited in the literature. Thus, in order to identify works related to the one proposed in this master's thesis, the works referenced by these mapping on the type of test debt were analyzed.

When analyzing the papers mapped in the works of Li et al. and Alves et al., those that specifically dealt with test debt and ways of managing it were selected. In addition, to expand the search, articles that cited the previously selected works were also included. This search was based on the research approach Snowballing (WOHLIN, 2014), which enables the identification of papers from the list of references (backward) and citations (forward) for a set of papers. In this work, it was performed only the search for citations (forward) at one level.

3.2 Understanding Test Debt

Samarthyam et al. (SAMARTHYAM *et al.*, 2017) present an overview of *Test Debts*, factors that contribute to this type of debts and strategies for repayment of acquired *Test Debts*. These authors also classify *Test Debts* into: (i) unit testing; (ii) exploratory testing; (iii) manual testing; and (iv) automated testing. For each type of test, they present possible factors that may generate debts.

Aiming to support the repayment of TDs, Samarthyam et al. (SAMARTHYAM *et al.*, 2017) propose a process with three macro activities: (i) Quantify the test debt, get the permission of the high administration, and execute the refund; (ii) Repay debts periodically; and (iii) Avoid

the *Test Debts* from accumulating. They also present strategies for the payment of *Test Debts* that involve the application of good practices of test codification and in the accomplishment of the activities of software testing. Besides that, these authors describe two case studies in industry, in which they report experiences with *Test Debts*.

Although Samarthyam et al. (SAMARTHYAM *et al.*, 2017) present a process with macro activities for the management of test debts and identify good practices to prevent and repay test debts, they do not propose an integration of these steps with an existing testing process in the organization. Another point to mention about this work is that although they present two case studies in industry, they are not detailed. Besides, they do not elaborate on the impacts of paying test debts in the case studies conducted.

3.3 Technical debt management for software testing process

The work of Sousa (SOUSA, 2016) aims to identify, within the software testing process, problems that occur during the execution of testing activities that could be considered as technical debt. Sousa presents a set of 22 TDs related to the software testing process collected from technical literature, their eventual causes and indicators. To evaluate these TDs, a survey was conducted with test professionals who answered if they agreed with each TD, its causes and indicators. Besides, a map was prepared to support professionals in managing TDs that may occur during the execution of the testing process. This map was evaluated by applying a questionnaire with software testing professionals.

The map proposed by Sousa presents the technical debts identified in the literature and focuses on the causes, indicators and possible solutions for these causes. Sousa, however, does not verify other important activities of technical debt management (e.g., measurement, prioritization, communication). In addition, the map was not applied to software organizations. Thus, its practical support in real projects was not confirmed.

3.4 Exploratory testing as technical debt

Shah et al. (SHAH *et al.*, 2014) performed a systematic review to answer the following questions: (i)“Does exploratory testing represent an archetypal example of technical debt inducing practice” and (ii)“Shall it be repaid later in the application life cycle?”. In this review, the authors present how the exploratory testing influences the test activities and the

related *Technical Debts*. Thus, they conclude that: (i) The lack of definition of test cases makes it difficult to perform regression tests and may cause residual defects; (ii) High human dependence, the missing of results evaluation, and lack of test planning may cause residual defects; (iii) Lack of documentation may lead to a poor understanding of the functionalities, generating rework and causing a wrong effort planning.

Therefore, the review by Shah et al. provides an overview of test debt concerning exploratory tests and presents how debts can be paid or prevented. However, they do not provide any more technical debt management activities (e.g., measurement, prioritization). Also, the work focuses only on test debts related to exploratory testing, not contemplating other types of software testing.

3.5 Technical Debt in Test Automation

Wiklund et al. (WIKLUND *et al.*, 2012) present studies indicating that there could be a general trend of high technical debt present in many test execution automation implementations, causing problems for the use, extension, and maintenance of these systems. In addition to the presentation of test process improvement models, and how they address the testing infrastructure, the results of a case study investigating potential sources of technical debt in test execution automation systems are reported.

The work focused on automated testing and brought contributions with observations specific to the area, such as the effects caused by the sharing of tools and the fact that there is instability in the execution of tests in different environments.

Therefore, there was no concern with the identification of test debts in other types of software tests. Also, the work does not detail ways to manage these debts.

3.6 Related Work Comparison

Table 3 presents a comparison of the identified related works. They were compared regarding the following characteristics:

1. Test debt management
2. Test debt management in a testing process
3. Conduct case study

The characteristic *Test debt management* indicates that the work presents some

activity, approach, or strategy to manage test debts. The characteristic *Test debt management in a testing process* denotes that the work presents suggestions for changes in a test process in order to manage the test debt. Finally, the characteristic Conducted Case Study indicates whether the work conducted any case study in which some test debt management activity, approach or strategy has been applied.

Tabela 3 – Comparison between related works

WORK	TEST DEBT MANAGEMENT	TEST DEBT MANAGEMENT IN A TESTING PROCESS	CONDUCT CASE STUDY
Samarthyam et al., 2017	Partially	No	Yes (two)
Sousa, C., 2016	Partially	Yes (based on ISO/IEC 29119)	No
Shah et al., 2014	Partially	No	No
Wiklund, K., 2012	Partially	No	No

Fonte: The Author

All works include characteristic *Test debt management*. However, they perform only a few management activities, mainly *identification* and *repayment*, and do not focus on the other management activities that may be useful for proper test debt management (e.g., measurement and prioritization).

Regarding the characteristic *Test debt management in a testing process*, only the work of Sousa et al. (SOUSA, 2016) proposes the management of test debts in an integrated manner with the testing process.

Regarding the conduction of case studies, only the work of Samarthyam et al. (SAMARTHYAM *et al.*, 2017) presents two case studies applying the approaches presented for test debt management.

3.7 Conclusion

In this chapter, the works related to this master's thesis were presented. In order to identify these studies, a search for studies dealing with test debt was performed based on the systematic mapping presented by Li et al. (LI *et al.*, 2015) and Alves et al. (ALVES *et al.*, 2016). Aiming to expand the results, a search was performed based on the Snowballing(WOHLIN, 2014) principle, seeking studies that cited those previously selected. With this search, four studies were found related to what is proposed in this dissertation. These studies were listed according to the following comparison characteristics: (i) Test debt management; (ii) Test debt management in a testing process; and (iii) Conduct case study.

After the process of comparison between the characteristics listed, it was possible to realize that none of the studies can contemplate all the characteristics presented or implement them only partially.

4 PROBLEM IDENTIFICATION

This chapter presents the empirical study conducted at the GREat Test Factory (GTF). This study occurred between 2017 and June 2019 and observed the test activities performed by the GTF team. The results of this empirical study were used as input for the construction of the catalog, focus of this master's thesis.

This chapter is divided as follows. First, Section 4.1 presents the scenario in which the problem was identified. Then, Section 4.2 provides an overview of the conducted study as well as the research questions, the profile of the members of the GREat Test Factory and the objects of the study (mobile applications and web systems). Section 4.3 details how the empirical study was conducted and Section 4.4 introduces the data collected in the study. Section 4.5 describes the conduct of the empirical study detailing the answers to previously planned research questions. Finally, Section refsec:form-problem presents how the problem in the empirical study can be formulated.

4.1 Problem identification scenario

The first step of the research methodology used in this work corresponds to the problem identification (Chapter 1, Section 1.4). This step aims to identify possible improvements that can be made in the subject of the research by analyzing the activities performed in the industry (GORSCHEK *et al.*, 2006).

In this research, the problem identification comes from the experience in a successful long-term partnership with the industry in Research, Development and Innovation (R&D&I) projects (ANDRADE *et al.*, 2017a). In these projects, the GREat Test Factory[1] (GTF) acts providing testing services.

The GTF team has followed a well-defined testing process (ANDRADE *et al.*, 2017b), presented in the Section 2.1.1. Following this process has proven to be a key factor to monitor testing activities, since there are events, such as daily and retrospective meetings, that allow the entire team to be aware of the issues that may impact the progress of the GTF activities (SUTHERLAND; SCHWABER, 2013).

In the monitoring of the activities performed by the GTF team, it was observed, for example, that when the team is pressured to deliver some artifact, it is common to have

[1] GREat Test Factory - http://fabricadetestes.great.ufc.br/

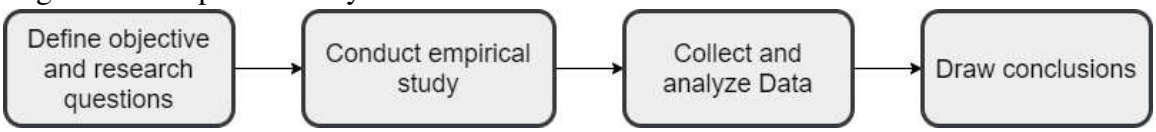
failure in the delivery or immature artifacts are generated. An immature artifact can be defined as any artifact that is not fully developed. In the case of test artifact, it can be an incomplete test specification, such as, for example, the lack of some test cases or their preconditions. The existence of these immature artifacts can cause, for example, the occurrence of false positives or test estimation errors (Chernak, 2001).

In order to identify and quantify immature artifacts generated by the GREat test factory team or the lack of test artifacts, an empirical study was conducted on five products tested by GTF team between 2017 and June 2019.

4.2 Empirical Study Overview

Figure 4 presents the steps of the empirical study presented in this chapter. This study is based on the work of Perry et al. (PERRY *et al.*, 2000) and is composed of the following activities: (i) Define objectives and research questions, step in which the explicit objective and research questions are presented; (ii) Observe the situation, step in which the study is conducted; (iii) Collect data, the data are collected according to the planning made; (iv) Analyze Data, in this step the analysis and discussion of collected data are performed; and (v) Conclusion, which present a discussion about the research questions.

Figura 4 – Empirical Study Overview



Fonte: Based on (PERRY *et al.*, 2000)

4.2.1 Definition of goals and questions

It was examined the testing activities performed in the GTF to test demands for five software products in the industry. This analysis aimed to identify and quantify the existence of immature artifacts or the absence of such artifacts. The questions that arise from the objective and that must be answered with this analysis are:

a) **RQ1.** *Are there any problems with the testing activities performed at the GREat Test Factory?*

b) **RQ2.** *How to identify the occurrence of problems in the testing activities performed by the GREat Test Factory team?*

c) **RQ3.** *How to prevent the occurrence of problems in the testing activities performed by the GREat Test Factory team?*

As a collection method for answering research questions, notes of members' comments were made during daily meetings and retrospective meetings. Also, some metrics were analyzed to complement the observations made.

4.2.2 Study objects

The objects of the study were five software products developed in partnership with the industry. The selected products were considered the most relevant for the client regarding number of active users (in the case of mobile applications) and importance for automation in the execution of the company's activities (in the case of web tools). In addition, the selected products were the ones that most demanded tests for GTF. Table 4 presents the products, the platforms for which they were developed and the number of releases launched between 2017 and June 2019. The names and versions were changed for confidentiality reasons.

The development processes used to generate the products were similar, but each product had its own particularity that the Test Factory needed to adapt to, such as different levels of specification and documentation of the products. In the cases of superficial information about the functionalities, the GTF team had to intensify communication with the development team to clarify possible doubts and mitigate the development of immature test artifacts.

A total of 159 releases were analyzed in the last three years. Some of them had the tests performed in the Test Factory and others not (see details in the section 4.4). The releases that were validated by GTF followed the testing process. However, the process could be adapted to better meet the needs of the test applicant (e.g., not performing some activity of the process that did not apply to demand or preparing test reports about battery and band consumption).

Two of the products analyzed were mobile applications (T1 and T2) based on Android technology. They were continuously updated with new features and, during this period, they have over 3 million active users together. Testing these applications is a challenge due to the need to test them on multiple Android versions, running on different target devices, to ensure that they work as planned. Non-functional tests were also performed, such as: (i) Battery tests, to identify the impact of the apps on the device's battery consumption; and (ii) Bandwidth consumption tests, to verify the consumption used while the apps were running.

Three of the products analyzed were web tools (T3, T4 and T5). The T3 tool is a

server that manages the sending of media content to a client application installed on Android phones. The messages are sent using a Google Cloud Message (GCM) service and then arrive in the Android application. Due to this intermediate service in the communication between server and client, it was necessary to have more detail and attention to the steps performed during the tests for the tracking of any error or inconsistency.

The T4 and T5 tools were of global use and their complexity required constant and well elaborated tests, because they presented critical information about the internal processes of the company: schedules, monitoring of activities and allocation of resources.

Tabela 4 – Tool and applications, and the number of releases in the years 2017, 2018 and 2019

Tool	Platform	#Releases (2017)	#Releases (2018)	#Releases (2019)
T1	Mobile	3	25	4
T2	Mobile	2	10	7
T3	Web	4	9	4
T4	Web	3	39	12
T5	Web	5	27	5
Total	2M e 3W	17	110	32

Fonte: The Author

4.2.3 Profile of the Test Factory Members

The Test Factory team who participated in the study had the following profiles: (i) One test manager, responsible for planning the tests and monitoring the activities performed; (ii) One test leader, more experienced member of the team, who assisted in the allocation of resources and technically monitored the team; (iii) Three test analysts, responsible for the preparation and execution of the tests, and for reporting the test incidents; and (iv) Three testers, less experienced members of the team, whose main function was to perform the tests previously specified.

4.3 Conducting the empirical study

As mentioned in Section 4.2, the GTF uses a testing process (ANDRADE *et al.*, 2017b) developed to, among other objectives, systematize and monitor the testing activities performed by the team. Through some events defined in this process (daily, kickoff and retrospective meetings), data reported in the tools for monitoring activities (*e.g.,* JIRA[2]) and tools for

[2] Tool for monitoring tasks and monitoring projects - https://br.atlassian.com/software/jira

documentation (*e.g..,* Confluence[3] and TestLink[4]), it was possible to identify problems in the testing activities that were performed in the GREat TF.

The daily and retrospective meetings used in GTF are similar to those proposed in the Scrum guide (SUTHERLAND; SCHWABER, 2013). The daily meeting is used to identify specific problems faced by the team and possible delays in the execution of the activities. In the retrospective meeting, the team discusses the positive points that occurred during the demands tested in a period, and that should continue happening in the next demands. The improvement points refer to the problems that occurred during the demands of the period analyzed and suggest actions for improvements to avoid that these problems occur again.

By observing the meetings held and analyzing the data reported in the monitoring tools used in the Great Test Factory, it was analyzed how the members of the GTF identified and quantified the existence of immature artifacts or the absence of tests. This analysis occurred in the period of 2017 to June 2019.

4.3.1 2017

In 2017, when observing the discourse of the GTF members in the daily meetings, it was noticed that, when running the tests, the team detected many errors in the cases of previously prepared tests, for example, the lack of certain prerequisites and essential steps for the correct execution of the test cases.

A widely used technique to identify problems in elaborated test cases is the test case review (SPILLNER *et al.*, 2014). Thus, after identifying the existence of these immature test cases, in a retrospective meeting, the team decided to systematize the test case review to identify the mistakes present in the test cases elaborated. This review started to be made systematically by the most experienced test analyst of the team, and the problems identified were passed on to the analyst who wrote the test case.

To standardize the reviews performed, a test case review template, based on the work of Petunova et al. (PETUNOVA; BĒRZIŠA, 2017), was defined. This template was also useful for the rest of the team because, with its dissemination, the team had access to how the test cases would be reviewed. Hence, it was possible to prevent similar errors from happening frequently.

In order to avoid that the improvements suggested in the revisions were not made

[3] Tool for general project documentation - https://br.atlassian.com/software/confluence
[4] Tool for test management - http://testlink.org/

and were lost, a task was created in JIRA with all the improvements of the test cases identified during the test demand that was being executed.

4.3.2 2018

In 2018, the analysis of the follow-up meetings held in the GTF (i.e., daily and retrospective meetings) and of the monitoring tools continued to be performed.

It was verified that, as a large number of reviews were under the responsibility of only one analyst, the review of the test cases was not being performed or was being carried out with delay, impacting on the previously defined deadlines.

In order to identify test case improvements and analyze the tests (code or documents), it is used peer test review. This technique is also used to pass the test knowledge from the most experienced to the least experienced professionals. (SMITH *et al.*, 2012). Thus, after discussion in a retrospective meeting about the reported problem, the team decided to use peer test review in order to decentralize the review of test cases and to encourage the discussion of improvements among the analysts who were conducting the review.

Also in 2018, when observing a kickoff[5] meeting of a test demand, it was identified that a previous release had not been tested and was already in the production environment.

The identification of this problem was only possible for the fact that the requesting team informed they had not sent the previous release for validation of the GREat Test Factory, as they had to deliver the release within the deadline agreed with the client, and there would be no time to perform the tests in the GTF.

In this context, it was observed that there could be releases not tested by the GTF launched in the production environment. Thus, it was necessary to identify the situations in which this occurred, in order to, for example, include, in the next demand for testing this software, the features not tested.

Besides asking the requesting team if there were previous releases without tests at the kickoff meeting, the GREat Test Factory team also started to analyze the release plan[6] of the projects that were being tested by the GTF. It is important to point out that this was possible because the GTF team had access to JIRA from the development team. By analyzing the release plan, it was possible to identify the existence of releases that were liberated for production

[5] Kickoff is a meeting defined in the GTF process, in which the team that requested the tests presents to the GTF team which features will be delivered in the release

[6] The release plan was registered in JIRA with the versioning of the releases that would be released.

without testing.

Another aspect observed in the year 2018 was related to *errors of test estimation*. It was decided to make this analysis due to the recurrence of deliveries after the deadline. Thus, it was started to register in JIRA the planned effort for the tests and the actual effort performed. Thus, for the identification of estimation errors, it was performed an analysis of the registered effort report for each test activity performed. If the actual effort was different from the planned effort, the estimation error was found.

As a way of mitigating perceived estimation errors, a study was also conducted by GTF team to semi-automatically calculate the estimated execution of test cases based on the complexity of the test cases developed. The result of this study was published (SOUSA *et al.*, 2018).

4.3.3 2019

In 2019, about improvements in test cases, no new strategies were added to identify these problems. However, concerning the existence of releases with no test, in addition to checking the release plan, a comparison between releases was carried out.

In this comparison, the existence of differences between the last tested release of the software and the release in production was checked. After the verification of the differences (*e.g.,* new implemented functionalities), it was also verified if in the existing documentation (*e.g.,* test plan, test cases, tasks in the project follow-up tool) there were indications that tests of these differences were executed. Otherwise, the absence of tests would be detected.

With regard to the analysis of the estimation error, in addition to the comparison between planned and actual, the effort of similar tests on a historical basis was also analyzed in order to compare with the current estimate and identify the possibility of estimation error.

4.4 Collecting and analyzing data

The data were collected and analyzed based on the notes made from the participation in the follow-up meetings that occurred in the GTF. Data were also collected from the monitoring tools used in the GTF (e.g., JIRA, Confluence, TestLink).

The presentation of these data in this master's thesis is divided by the problems identified in the three years of observations and notes performed in the GTF.

4.4.1 Improvements in test cases

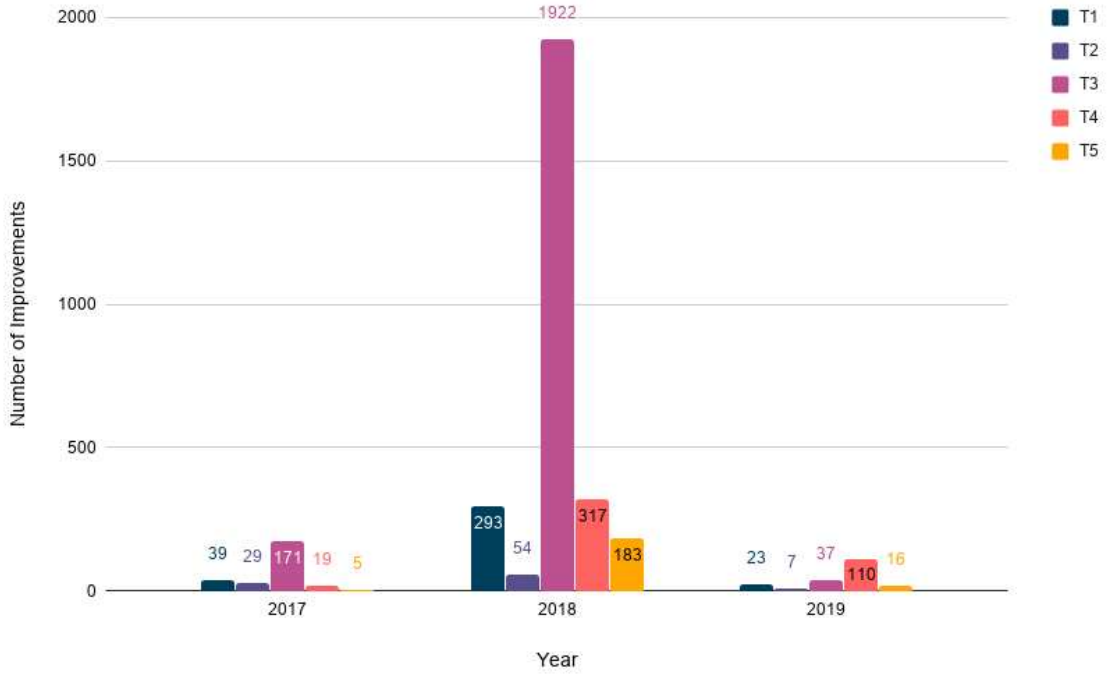
The execution of tests using immature test cases is a substantial threat to the quality of tests performed (Chernak, 2001). Thus, the identification of improvements in test cases is essential to reduce the risk of incorrectly performed tests.

As stated in Section 4.3, the identified improvements were documented in JIRA and Confluence. By analyzing the data collected from these tools, it was possible to quantify the improvements identified each year.

Figure 5 presents the improvements identified in the test cases of the five software products object of this study. Figure 6 presents the consolidated improvements identified in all products per year. Analyzing these data, it is possible to observe a greater number of improvements identified in 2018, when there was a greater amount of releases launched and tested by GTF (110 releases, while in 2017 there were 17 releases, and in 2019, 32 releases).

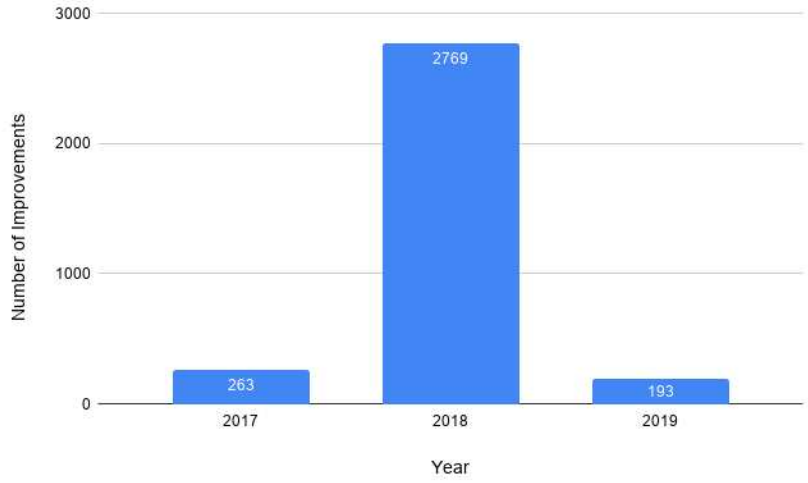
Another point identified is the far larger amount of improvements identified in the T3 product in 2018, when there was a refactoring in the system, which caused a change in almost all test cases elaborated for this product.

Figura 5 – Number of adjustments identified for each tool in 2017



Fonte: The author

Figura 6 – Number of total adjustments identified per year



Fonte: The Author

4.4.2 Lack of tests

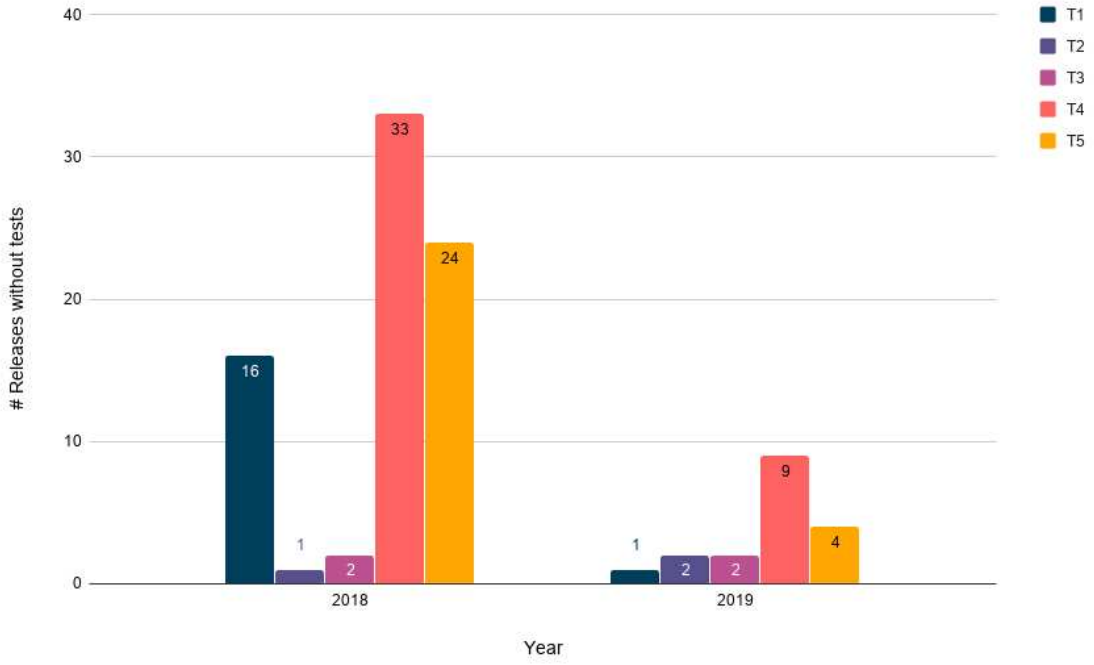
Failure to perform tests on releases already released in the production environment may contain critical defects that can impact the quality of the software and, in some cases, even make its use unfeasible. Thus, the identification of releases launched without testing is essential to ensure the quality of the software developed (HASS, 2014).

The releases for which no tests were identified were documented in JIRA and Confluence. Thus, it was possible to quantify the number of releases launched without tests in the years of the study.

Figure 7 presents the number of releases without tests identified in each object of study, divided between the years 2018 and 2019 (until June). It can be noticed a large number of releases delivered without validation, stands out T4 and T5 products in 2018, with more than 20 releases delivered without testing. It was observed that most of the releases launched without testing were *hotfix* releases (TECHTERMS, 2015). This type of release is necessary when defects are found in the production environment and require rapid intervention to correct the problem identified.

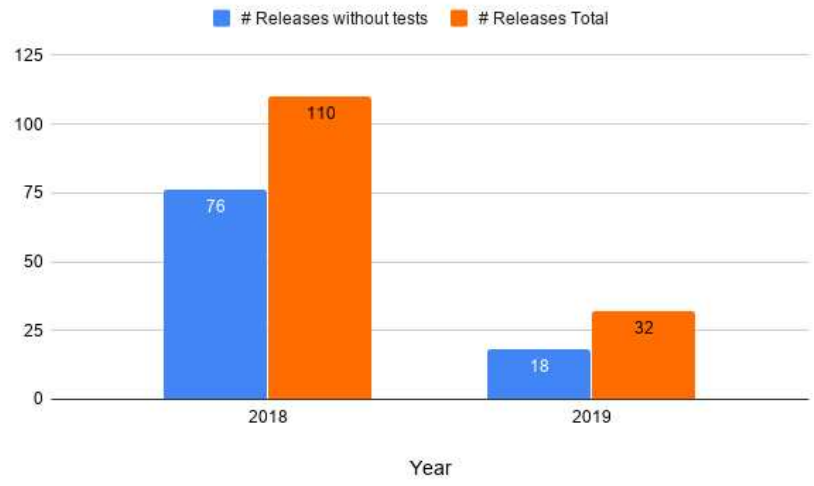
Figure 8 illustrates the number of untested releases compared to the total number of releases each year. Analyzing this data, it is clear that, proportionally, the number of releases without testing is very high.

Figura 7 – Lack of tests identified for each tool in 2018



Fonte: The Author

Figura 8 – Releases without tests versus releases total



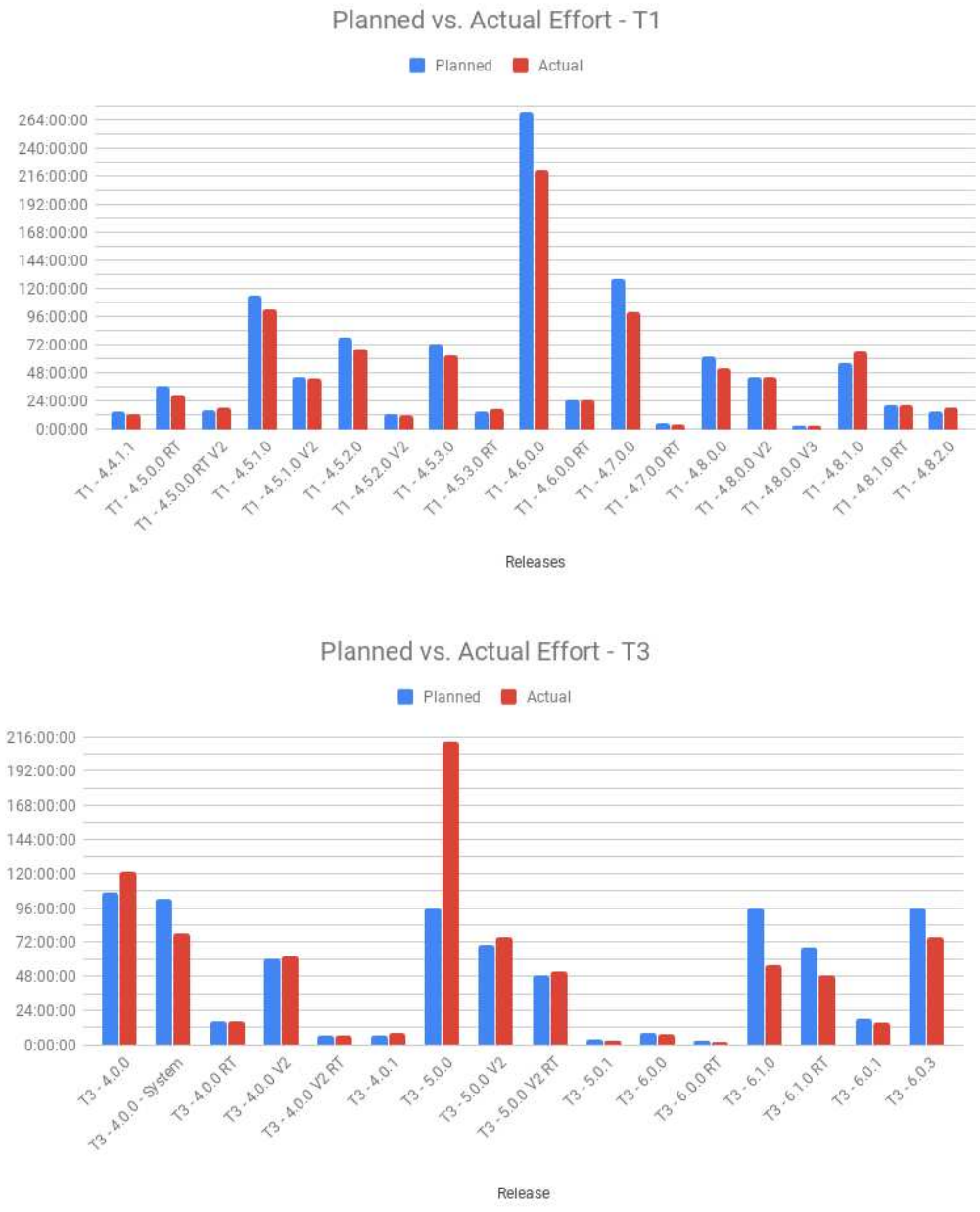
Fonte: The Author

4.4.3 Test estimation errors

In 2018 and 2019, it was also analyzed the data on the estimated effort and the actual effort expended in each release of the five software products, objects of this study. To collect this data, it was used Confluence (in which the estimated effort for each release of the software tested was documented) and JIRA (in which the GTF team reported the actual effort expended in each test task of the release being tested).

Regarding 2018, in the comparison between the planned effort (in hours) and the real effort for the five products, objects of this study, it can be observed that 4 of the 19 T1 releases; 10 of the 15 T2 releases; 4 of the 16 T3 releases; 6 of the 21 T4 releases; and 2 of the 5 T5 releases presented real effort above the estimated effort in the planning. In the analysis of these data, it was found that there were errors in estimating effort in 26 releases in 2018. Figure 9 presents the results for T1 and T3 products, which had a more significant disparity between planned and actual effort.

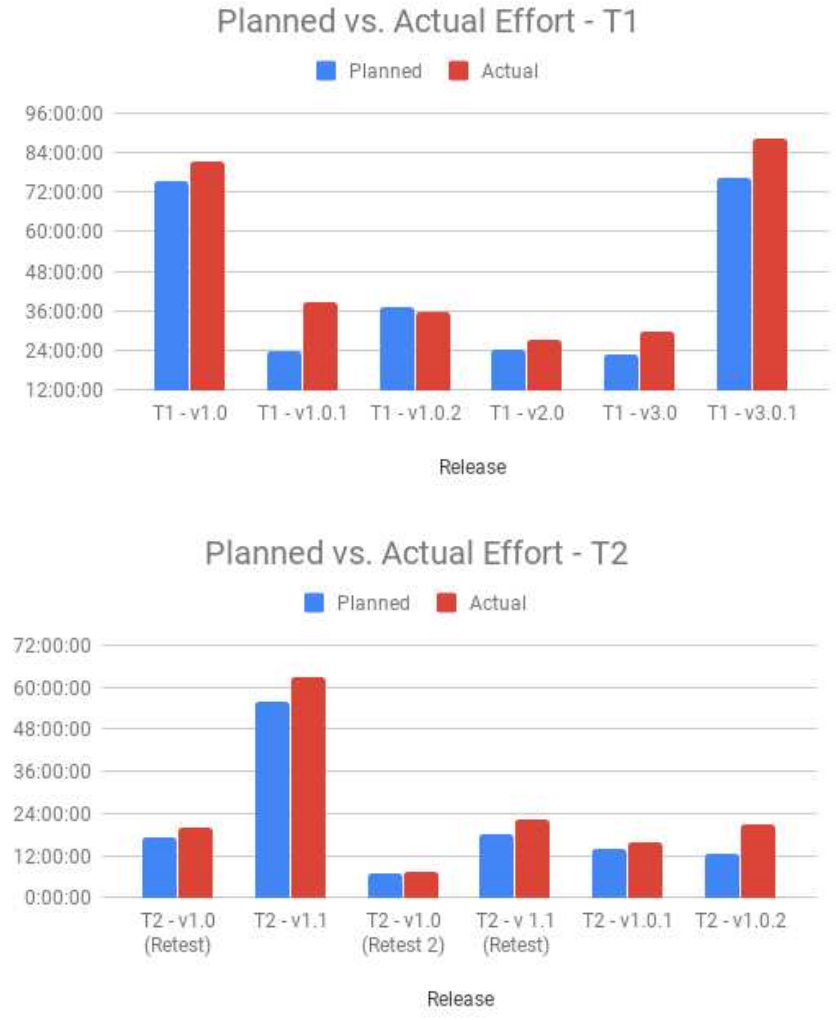
Figura 9 – Comparison of planned and actual effort for the analyzed products in 2018



Fonte: The Author

With respect to 2019, in the analysis performed, it was found that 5 of 6 T1 releases, 6 of 6 T2 releases, 1 of 4 T3 releases, 0 of 5 T4 releases, and 3 of 3 T5 releases had the real effort above the planned, totaling 15 releases with errors in effort estimates. Figure 10 presents the effort for specification and execution of the tests for T1 and T2, products that presented the worst effort estimates.

Figura 10 – Comparison of planned and actual effort for the analyzed products in 2019



Fonte: The Author

4.5 Drawing conclusions about the empirical study

In this section, based on the research questions, the results identified in the empirical study will be discussed.

4.5.1 Research questions

After analyzing the results collected during the empirical study, it is possible to answer the previously defined research questions as follows.

RQ1. *Are there any problems with the testing activities performed at the GREat Test Factory?*

In addition to observing the daily activities of the GTF and the follow-up meetings held, data from the monitoring tools (e.g., JIRA, Confluence, and Testlink) were analyzed to answer the first research question.

As presented in Section 4.4, it was observed that, during the years of the empirical study, there were immature test cases and lack of tests in the releases (including releases already launched in the production environment). In addition, test estimation errors that impacted the test delivery deadline were identified.

RQ2. *How to identify the occurrence of problems in the testing activities performed by the GREat Test Factory team?*

The second research question was answered by observing the actions taken by the GTF team to identify problems in the performed testing activities.

The experience in identifying the problems detected in the GTF was documented in the form of lessons learned so that they could be used in future projects or in other organizations with the goal of identifying similar problems.

The lessons learned in this empirical study are organized as follows: (i) **The event:** what happened that needs to be highlighted; (ii) **The cause:** why this event occurred; (iii) **The impact:** what was the consequence of this event; and (iv) **Actions:** what will be done in the next steps (SABINO, 2016). It is important to note that events, causes and impacts might be repeated in more than one lesson learned, but the actions are unique for each lesson.

L01*: Analyze customer test report, emails, or product reviews*

- **Event**: No identification of defects in production time
- **Cause**: Have not received or analyzed the defects found by customer or end user
- **Impact**: Customers or users unhappy with the use of the software
- **Actions**: To check the customer report, in cases where the release is also evaluated by the customer. The objective is to identify possible defects not found at test phase. With regard to e-mails and product comments, one should analyze them to identify defects that were not found at test time

***L02**: Use stand-up and graphical tools to identify test estimation errors*

- **Event**: Late identification of effort estimation errors
- **Cause**: The planned versus actual effort was only verified at the end of the test demand
- **Impact**: Delays in planned deliveries
- **Actions**: To use stand-up meetings to monitor the progress of the activities and whether the implementation effort remains close to what was estimated or not. In addition, to use graphical tools such as a burndown chart to assist in identifying anomalies between the estimated and the actual in the execution time of the demand (RUBIN, 2012)

***L03**: Use tools for documentation of identified problems*

- **Event**: The team did not register the identified problems (e.g, improvements in test cases, absence of testing, many defects found by the customer)
- **Cause**: Due to the fact that it was not in the process or due to pressure to deliver the tests
- **Impact**: Problem accumulation without documentation. Thus, these problems can be lost and not be solved
- **Actions**: To insert the activity of documentation of problems identified in the testing process. This documentation was done in project tracking tools (e.g., bug tracker and Wiki).

***L04**: Present the problems already identified to the team*

- **Event**: Some members of the team are not aware of previously identified debts
- **Cause**: Lack of proper communication
- **Impact**: Accumulation of new debts
- **Actions**: To present the already known problems to the team, so they understand how the problems were caused and, thus, avoid that similar problems are acquired again

***L05**: Use charts to assist in the visibility of the problems identified*

- **Event**: Even documenting the identified problem, the team did not perceive their existence
- **Cause**: Lack of visibility
- **Impact**: Unsolved problem accumulation
- **Actions**: To use charts presenting the amount of problems identified by version. These charts should be easily accessible to the team

The research question **RQ3.** *How to prevent the occurrence of problems in the testing activities performed by the GREat Test Factory team?*, was answered by observing the actions taken by the GTF team to prevent that the problems identified were repeated. Most of

the preventive actions came from the retrospective meetings held and applied to the next test demands.

These preventive actions have also been documented as lessons learned and are described below.

***L06**: Merge profiles of professionals in test demands*

- **Event**: In cases where only professionals with a junior profile worked, the amount of problems identified increased
- **Cause**: Professionals with little experience and limited knowledge of the defined process
- **Impact**: Immature test cases, resulting in, for example, defects not found at the time of testing
- **Actions**: To merge profiles of professionals allocated to test demands. More experienced analysts should be responsible for reviewing and passing on knowledge of the testing process to other less experienced analysts

***L07**: Perform test cases improvements at runtime*

- **Event**: Improvements in test cases were identified, but often not yet made at the same time of the release test
- **Cause**: Due to the delivery time, the team does not make the adjustments within the time defined for test demand
- **Impact**: The test cases remained immature or with errors. Thus, no defects could be identified with the use of these immature test cases
- **Actions**: To negotiate with the test requester a change in the delivery date to correct the identified improvements. If it was not possible to change the delivery date, the problems encountered should be reported at the stand-up meeting so that everyone is aware of them

***L08**: Estimate the test effort separately from the development effort*

- **Event**: Not much time to run the tests
- **Cause**: The development team decided the delivery date for the client without considering the effort required to perform the tests
- **Impact**: Tests not carried out or performed incorrectly due to urgent need for delivery
- **Actions**: To analyze the scope of the development and make appropriate estimates to ensure that there will be adequate time to conduct the system tests. Tests should be estimated separately from the development estimate. Thus, an eventual delay in development will not compromise the time already allocated for testing. In addition, the team can negotiate

with stakeholders for a new deadline for delivery in order to have more time to complete the release tests

***L09**: Monitor the use of existing test process*

- **Event**: The team was not using the testing process correctly
- **Cause**: Due to the lack of knowledge of the existing process or because of delivery pressure
- **Impact**: Occurrence of problems that could be prevented by the correct use of the defined testing process
- **Actions**: To monitor the existing process and check if the team is following the defined activities. If, during the use of the process, necessary improvements in the process are identified, they should be implemented with the aim of preventing the incidence of test debts. It is worth mentioning that after the change in the process, it is necessary to monitor whether these improvements have a positive impact on the quality of the tests performed or not. This analysis can be done by checking the results of previously defined quality indicators

***L10**: Use automated tests to improve the tests performed*

- **Event**: Errors in the execution of tests performed manually
- **Cause**: Human errors
- **Impact**: Immature tests cases, so that in their execution defects are not found in time for tests
- **Actions**: To use automated tests for the most critical functionalities of the system
- **Comments**: Experience in the use of automated testing in the GREat Test Factory is reported in (VIEIRA *et al.*, 2018)

4.6 Problem Formulation

After analyzing the literature on problems similar to those that occurred in the GREat Test Factory, it was identified that these problems could be formulated using the concept of Technical Debt. In continuation of the literature studies, it was identified that the subtypes of Test Debts presented by Li et al. (LI *et al.*, 2015) could be applied in the GTF and that they relate to the causes of test debts (Chapter 2).

Among the subtypes of test debts mapped by Li et al. (LI *et al.*, 2015), in the empirical study conducted at the GREat Test Factory, the problems identified can be linked to

the following subtypes of test debt: (i) Defects not found in tests; (ii) Lack of tests; and (iii) Test effort estimation errors.

The subtype *Defects not found in tests* refers to system failures found by customers and/or end users, which could have been identified during tests performed by GTF. This subtype can be linked to the problem identified in the GTF regarding the existence of immature or defective test cases.

The subtype *The lack of test* occurs when no tests are performed for any release of the product. This subtype can be linked to the problem identified in the GTF about the absence of tests in certain observed releases.

Finally, the subtype *Test effort estimation errors* refers to incorrect estimates that end up impacting the quality of the test performed or the delivery deadline of the test demand. This subtype was also identified and presented in the empirical study conducted in the GTF.

4.7 Conclusion

This chapter presents the empirical study conducted at the GREat Test Factory between 2017 and June 2019. The testing activities carried out for two software products that required testing at the GTF were observed. The following research questions were planned and answered through the empirical study: (i) *RQ1. Are there any problems with the testing activities performed at the GREat Test Factory?*; (ii) *RQ2. How to identify the occurrence of problems in the testing activities performed by the GREat Test Factory team?*; and (iii) *RQ3. How to prevent the occurrence of problems in the testing activities performed by the GREat Test Factory team?*

The answer to the first question presented which problems were identified in the execution of the activities in the GTF. The answers to the second and third questions presented the actions taken to identify the problems. These actions are displayed in the format of lessons learned in order to facilitate their use in later projects or in other organizations that work with software testing.

This empirical study identified the problem to be tackled in this master's thesis. After the identification of this problem, research was performed in the literature to find similar problems and realize how they could be managed. Through this research, it was observed that the problems identified in the GTF could be formulated as test debt. Thus, the approaches and techniques presented in the literature about test debts could be used to manage this type of debt.

The results obtained from the identification of the problems in the testing activities

of a test factory and the lessons learned to identify and prevent these problems formed one of the bases of the catalog developed in this master's thesis.

5 TESTDCAT

This chapter describes the steps to build the catalog, presents the intermediate versions generated during its construction and, finally, details its latest version. The catalog, called TestDCat, consists of technical debt management activities, test debt subtypes and a set of actions.

This chapter is divided as follows: Section 5.1 summarizes the steps used to build the catalog and the versions developed, describing the activities and presenting the results achieved during its construction. Section 5.2 presents the steps taken before the creation of the catalog, which provided the necessary inputs for its creation. Section 5.3 and Section 5.4 present, respectively, the steps for building and evaluating the first and second versions of the catalog. Finally, Section 5.6 details the latest version of the generated catalog, presenting its structure and elements.

5.1 Steps to build the catalog

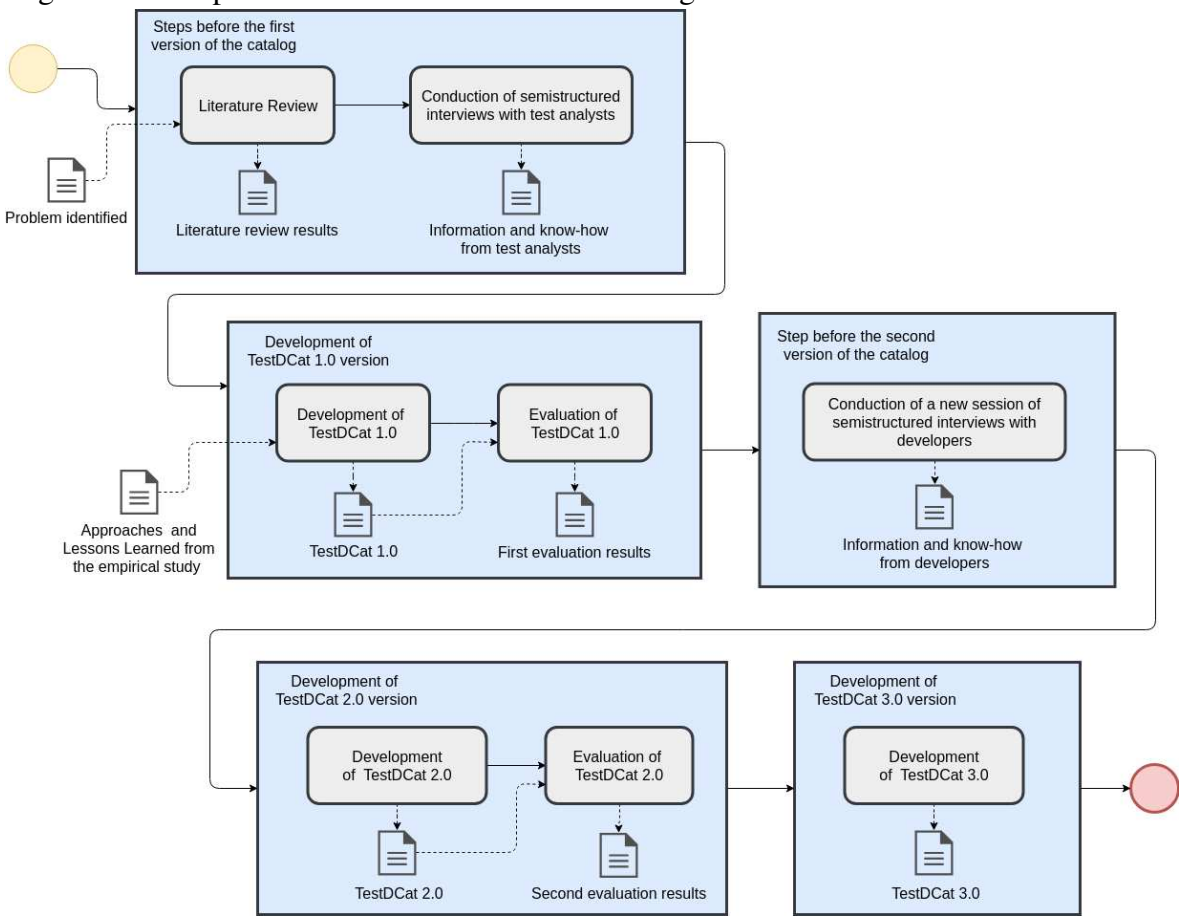
Catalogs are ways of organizing the information and know-how that originates from practitioners (CHUNG *et al.*, 2012). Catalogs can be used as a source of information on how to deal with technical debt (KRUCHTEN *et al.*, 2012) (OZKAYA *et al.*, 2011). Therefore, the knowledge acquired as a result of performing this master's thesis was organized in catalog format and made available for the use of practitioners who work with software testing and test debts.

The design to build the catalog follows the steps presented in Figure 11. These steps are based on the methodology presented in Section 1. This methodology favors cooperation between academia and industry, which allows researchers to know and offer solutions to the problems faced by industry.

The steps for the construction of the catalog, called TestDCat, are divided into sections:

1. **Steps before the version 1.0 of the catalog:** Consists of the steps taken to generate the necessary inputs to create the catalog. The steps of the literature review and conducting semi-structured interviews with test analysts were performed.
2. **Development of TestDCat version 1.0:** These are the steps taken to build the version 1.0 of the catalog and its first evaluation.
3. **Steps before the version 2.0 of the catalog:** Presents the execution of the

Figura 11 – Steps used to build the TestDCat Catalog



Fonte: The Author

activity of conducting new interviews in order to collect more data for the construction of the catalog. This new series of interviews focused on the vision of the software developer, who also performs testing activities.

4. **Development of TestDCat 2.0 version:** Refers to the construction and evaluation activities of the version 2.0 of the catalog. In this new version, the improvements identified in the previous steps were made.
5. **Development of TestDCat 3.0 version:** Construction of the latest version of the catalog based on the improvements identified in the evaluations previously performed.

5.2 Steps before the first version of the catalog

The following activities are related to the production of inputs for the creation of the first version of the catalog. A literature review was conducted in order to identify how the studies dealt with test debts. Also, the first series of semi-structured interviews were conducted

to collect from test practitioners what and how they dealt with the test debts they knew.

5.2.1 Literature review

Following the methodology used in this work (Chapter 1, Section 1.4), the next step after the identification of the problem is to conduct a study of the state of the art. This study aims to understand the identified problem better and find how work found in the literature deal with the problem.

To perform this study, a literature review was conducted in order to deepen the knowledge on test debts and identify which solutions are commonly used in the literature in the handling of these debts.

In deepening the study on test debts, it was identified that most of the papers dealing with this type of debt did not provide an overview of how to deal with various causes of test debts (details on the causes of test debt in Chapter 2, Section 2.3). Also, the majority of the work focused on test debts related to testing automation code. Thus, it was detected the need to identify and consolidate in one place ways to manage the several causes of test debts, involving those that are not only code-related.

With the results obtained from the literature review, techniques and approaches used in academic works were identified. However, to improve the results identified in the literature review, a series of semi-structured interviews were conducted in order to identify ways of managing test debts from the perspective of practitioners working in software testing activities.

5.2.2 Conduction of semi-structured interviews

Semi-structured interviews are a flexible and powerful approach to capture data about a given phenomenon (HOVE; ANDA, 2005). In this interview format, topics are defined and serve as a guide for conducting the interviews. However, the format of the questions can be adapted according to the progress of the interviews with each participant (RUNESON; HÖST, 2009). Thus, aiming to investigate practitioners' perspectives about test debts as well as to identify their strategies to deal with them, ten semi-structured interviews were conducted an divided in two different moments.

The first series of semi-structured interviews aimed to collect the point of view of professionals who worked exclusively with tests on how they dealt with the test debts they knew. Thus, this first series was focused only on test analysts.

The second series of the semi-structured interviews were conducted in order to collect the software developers' point of view on test debts and to make the catalog more complete for all professionals working with tests on their projects. This second series was carried out only with software developers who also performed testing activities on their projects. Conducting these interviews generated the necessary inputs for making improvements to the existing catalog and for generating its second version. Figure 12 presents the performance of one of these semi-structured interviews conducted.

Figura 12 – Conducting one of the interviews

Fonte: The Author

Two researchers (head and auxiliary/observer) conducted the interviews. As stated in the work of Hove et al. (HOVE; ANDA, 2005), with the participation of more interviewers, the participants usually talk more, and more questions can be asked. Table 5 shows the profile of the interviewers. Both researchers have already worked with software development and testing and are currently in the position of software development project managers. Following the guidelines of the work of Hove et al. (HOVE; ANDA, 2005), the interviewers have good interview skills, such as: encouraging the interviewees to speak freely and without fear of retaliation, ask questions relevant to the context of the interview, follow and explore the topics defined for interview.

Each interview session took about two hours on average and explored questions about practitioners' experience, subtypes of *Test Debts*, and TD management activities. All

Tabela 5 – Researchers profile

ID	SPECIALITY	EXPERIENCE IN THE FIELD	LEVEL OF UNDERSTANDING ABOUT TECHNICAL DEBT
I01	Software development project management	7 years	Much familiarity with the concept of technical debt
I02	Software development project management	12 years	

Fonte: The Author

interview sessions were recorded, transcribed, and analyzed later. As a result of these interviews, it was obtained information on which subtypes of test debt were acquired in their practical experience and actions taken to manage the debts identified.

The activities performed in this step were based on the work of Turner (TURNER, 2010), which provides a practical guide for the design of qualitative interviews. The activities are as follows: (i) Interviews planning; (ii) Interviews conducting; and (iii) Data analysis and consolidation.

5.2.2.1 Interview planning

Initially, some areas and topics were narrowed down to elicit conversation with the practitioners (RUNESON; HÖST, 2009). Then, the topics were organized in a script format with an opening statement, a set of general questions to be explored by each topic, and additional questions designed to probe for information, in the cases it did not come up (the interview script is available on Appendix A). The topics and questions of the script were formulated following the good practices presented in (HOVE; ANDA, 2005), using questions that the participant needs to describe how he/she works (e.g., How the existence of these test debts are documented?), and reflexive questions (e.g., Would you have any suggestions for improving the process or activities that could assist in this identification?).

Based on the Runeson et al. (RUNESON; HÖST, 2009) work, it was developed a protocol to guide the interviews. It was also piloted the interview guide to help improve its instrumentation (ROGERS *et al.*, 2011). Other than helping to pay close attention to the relationship between the questions asked and the content produced during the interviews, the protocol also included statements of confidentiality, consent, options to withdraw, and intended use of the results. The protocol established the following steps to conduct the interview sessions:

1. **Profile Identification:** which included questions about the participants demo-

graphics, background, and experience in the field;

2. **Questions about *Test Debts*:** to identify evidence of their presence in projects participants were currently working;
3. **Questions about the TDs management activities:** Questions to identify if respondents execute any form of management among those identified *Test Debts*; and
4. **Slowdown:** used for final considerations, with questions about the impact of lack of TD management activities and the possibility of using a catalog that could assist the process of managing *Test Debts*.

It is worth mentioning that the questions related to the test debt subtypes and the test debt management activities were based on the results of the systematic mapping presented in the work of Li et al. (LI *et al.*, 2015). Both the test debt subtypes and the management activities presented in the work of Li et al. are described in Chapter 2, Section 2.2. Li et al. (LI *et al.*, 2015) presents a systematic mapping to obtain a better understanding of the concept of technical debt and an overview of the studies carried out in the management of technical debts. It is an important work in the literature on the concept of technical debt and one of the most cited.

The selection of participants was made according to the availability of professionals in the GREat laboratory and the GREat Test Factory, so the technique classified as selective or purposeful sampling was used (COYNE, 1997). This technique is commonly used in cases where a careful selection is made according to established preconditions. All participants work in research and development projects conducted in partnership with the industry at the GREat Research Group. In 2019, such projects encompass the development of six web tools and two mobile applications, which are periodically tested by the GREat Test Factory. Table 6 summarizes the participants profile.

5.2.2.2 Interviews conduction

Interviews occurred in two different moments. First, five interview sessions focusing on test analysts' point of view were conducted. Hence, the participants included three test analysts, a test leader, and a software testing researcher who also works on projects with industry. On average, they had about three years of experience working in software tests field, including functional and non-functional testing. Later, five more interviews were conducted to try to address software developers perspective on testing. For these interviews, five more participants

Tabela 6 – Participants profile

ID	SPECIALTY	EXPERTISE	CURRENT POSITION	TDs REPORTED
P01	Software testing	Functional and non-functional tests	Junior Test Analyst	- Deferring testing - Lack of tests - Defects not found in tests - Expensive tests - Test estimation errors
P02			Mid-level Test Analyst / Test Leader	
P03			Junior Test Analyst	- Deferring testing - Lack of tests - Lack of tests automation - Defects not found in tests - Expensive tests - Test estimation errors
P04			Researcher	
P05			Junior Test Analyst	- Lack of tests - Expensive tests - Test estimation errors
P06	Software development and testing	Software development and functional tests	Mid-level System Analyst / Technical Leader	- Low code coverage - Deferring testing - Defects not found in tests - Expensive tests - Test estimation errors
P07			Senior Systems Analyst / Technical Leader	- Low code coverage - Deferring testing - Lack of tests - Expensive tests - Test estimation errors
P08			Mid-level System Analyst	- Deferring testing - Lack of tests - Lack of tests automation - Defects not found in tests - Expensive tests - Test estimation errors
P09			Mid-level System Analyst	- Low code coverage - Lack of tests - Defects not found in tests
P10			Researcher	- Low code coverage - Deferring testing - Lack of tests automation - Defects not found in tests - Test estimation errors

Fonte: The Author

were selected: four system analysts and a technical leader. Besides their vast experience in software development activities, all of them also worked with functional tests.

The interviewers followed the instructions from the interview protocol previously defined. The information collected from participants was properly kept anonymous and private and was used exclusively for the analysis of test debt characteristics and actions in the scope of this investigation. Based on the work of Turner (TURNER, 2010), before beginning the interview session, the researcher organized a room with no distractions, and set the evaluation environment, including voice recorder so that each person's interview session could be documented. Before a participant entered the room, the recorder and all data collection instruments were already

available and organized. When a participant entered the room, the researcher explained what the investigation was about and requested participants' permission to record the interviews. Then, the participant would sign a *terms of free and informed consent* [1], which assured data confidentiality and anonymity.

5.2.2.3 Data analysis and categorization

Data can be analyzed quantitatively or qualitatively (RUNESON; HÖST, 2009). Quantitative analysis can be performed using statistics, while qualitative data can be analyzed by categorization and sorting.

With the purpose of quantifying the existence of test debts identified by the practitioners, it was performed a brief quantitative analysis on closed-ended questions including, for example, whether practitioners felt pressured to perform test activities, and the occurrence of different subtypes of *Test Debts*, also presented in Table 6. To further analyze the comments that often followed the objective answers, a qualitative analysis of these observations were also carried out.

Then, a qualitative analysis of the recordings of the answers to each interview was performed. Due to resource constraints (e.g., the effort required for analysis and experts to assist with interpretation) on more sophisticated methods, such as Content Analysis and Grounded Theory (ROGERS *et al.*, 2011), critical comments on the transcripts performed were identified and categorized by means of an open coding approach (BURNARD, 1991).

Overall, the analysis of the transcripts took about 70 hours. It was used the classification tree and *Technical Debts* management activities proposed by Li et al. (LI *et al.*, 2015) as a basis for the data categorization. Hence, the transcripts were organized into eight well-defined categories (as mapped by Li et al. to test debt subtypes) that guided data consolidation.

5.3 Development of TestDCat 1.0

For the development of TestDCat 1.0, the results of the literature review (Subsection 5.2.1), and the information obtained in the first interview session (Subsection 5.2.2) were considered. An evaluation was also performed in order to identify errors or possible improvements.

[1] This term can be accessed on the link: https://github.com/great-ufc/TestDCat/blob/master/Documents/Terms_of_free_and_informed_consent.pdf

5.3.1 TestDCat 1.0

The information obtained was organized in a matrix that represents the TestDCat 1.0 catalog. The matrix was formed by test debt subtypes, and, for each of them, a set of TD management activities was associated. Within each of these activities, there were information about “Approaches”, “Points of attention in the test process” and “Good Practices”. *Approaches* are ways of performing technical debt management activities and were chosen because it is commonly used and documented in the literature (LI *et al.*, 2015) (ALVES *et al.*, 2016). In order to contemplate cases in which there is already a well-defined testing process, it was chosen to describe the *points of attention in the test process* aims to make evident the activities that must be carried out to perform the TD management activity. *Good practices* was selected because it refers to relevant information that was identified during the interviews, which, if used, can assist in performing the TD management activity. The Figure 13 presents an example with the test debt subtype *Low code coverage* and the TD management activity *Identification*.

Figura 13 – *TestDCat 1.0* example

Test Debt subtype	**TD Management Activities**
	Identification
Low code coverage	**Approaches:** **Code analysis:** Investigate the code to analyze its coverage. This analysis is usually done with a specifc tool (e.g., Eclema).
	Points of attention in the test process: **Test Planning:** Definition of the coverage targets. **Test Design:** Registration of test coverage **Test Execution:** feedback about the coverage to test monitoring **Test Monitoring:** Collection and analysis of measurements
	Good Practices: Use of continuous integration to automate the code coverage verification process

Fonte: The Author

5.3.2 Evaluation of TestDCat 1.0

The goal of the first evaluation was to gather the participants’ perception of the catalog clarity, ease of use, and completeness. So, the evaluation intended to get answers to the following: *Does the TestDCat catalog have clear and enough information to support the management of Test Debts?*

For this evaluation, the entire catalog was presented, printed on paper to improve visualization and make it easier for participants to take direct notes on points that should be changed. This evaluation was performed with the five participants of the first series of interviews.

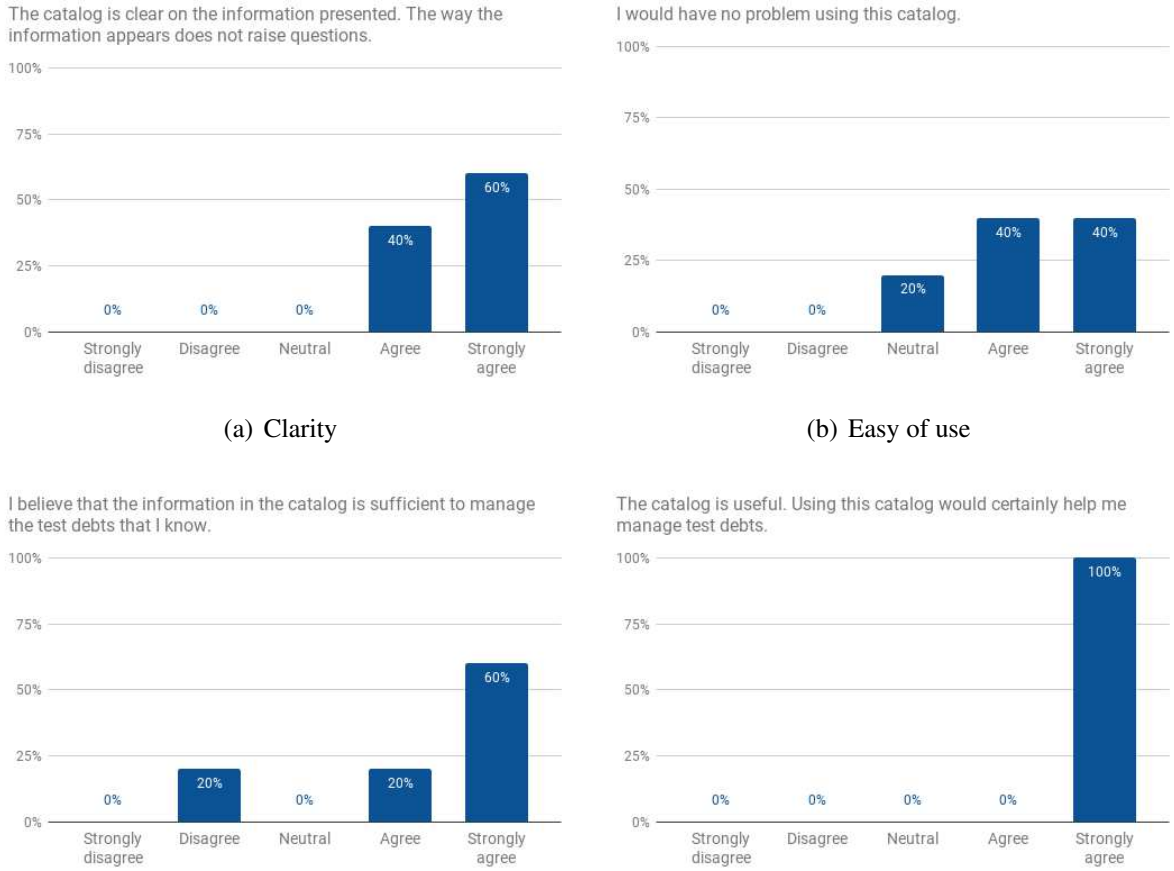
After reading and analyzing the catalog, they answered the questions of the survey[2].

Figure 14(a) presents the data with respect to clarity, all participants agree, 60% strongly agree and 40% agree, that the catalog presents the information in a clear and objective way. Furthermore, Figure 14(b) presents that most of the participants agrees, 40% strongly agree and 40% agree, that the catalog is easy of use. Just one of them chose a neutral response.

Regarding the completeness, Figure 14(c), most of the participants agrees, 20% strongly agree and 60% agree, that the catalog has enough information to support the management of *Test Debts*. On the other hand, one of them disagreed about the completeness of the catalog.

Figure 14(d) presents the data with respect to the usefulness of the catalog, all participants agree, 100% strongly agree, that it would certainly help users to manage their *Test Debts*.

Figura 14 – Survey Results



(a) Clarity

(b) Easy of use

(c) Usefulness

(d) Completeness

Fonte: The Author

In addition, the participants suggested improvements to the catalog in the open

[2] Available in: https://forms.gle/4LMKKww1Xp8U8ew86

questions. For instance, they asked for more details on each approach presented, as well as more practical information that would make the use of the catalog more precise.

5.4 Development of TestDCat 2.0

Based on the results of the first evaluation and new information from the second interview session, a new version of the catalog was created. In this new version, compared to version 1.0, the order of the elements *TD management activities* and *subtypes of* Test Debts has changed. In TestDCat 2.0, for each TD management activities, a list of test debt subtypes are associated. This decision was due to feedback received in which the evaluators informed that they preferred to access the actions according to the specific management activity. Thus, if the users wanted to focus only on actions related to the prevention of test debts, they can access the aspired category and find all actions related to this activity.

In the 1.0 version, *Approaches*, *Points of attention in the test process* and *Good Practices* were presented for the management of test debt subtypes. However, in the analysis of the evaluation feedback and the new interviews, participants felt the need for more assertive guidelines for the handling of test debts. Thus, the catalog was changed to present, for each subtype of test debt, a set of actions to manage the test debt. The actions are presented using the 5W1H model (Who, What, When, When, Where, Why, Why, and How). This model is often used for the development of action plans. It provides a detailed direction of how to perform the planned action (FERNANDES *et al.*, 2013). All the elements listed in the first version of the catalog were adapted to actions using the model 5W1H.

It is worth mentioning that, since this version takes into account the point of view of software developers who also work in software testing, many improvements were made in the catalog. These improvements were mainly in the subtypes of test debt that are code-related (e.g., Low code coverage and lack of test automation).

Figure 15 presents an example of the new catalog using the model 5W1H for the TD management activity “Identification” and the test debt subtype “Low code coverage”.

5.5 Evaluation of TestDCat 2.0

In order to verify if the content of the TestDCat 2.0 could be used with higher assertiveness, an in-depth evaluation was performed to assess the correctness, quality, and

Figura 15 – *TestDCat 2.0* example

TD management activity	Test debt subtype	What?	Why?	Where?	How?	Who?	When?
Identification	Low Code Coverage	Evaluate test coverage (i.e. existence of tests for each system functionality) by analyzing the code	By checking the percentage of code coverage, you can identify whether the intended code coverage has been achieved	Code repository	Investigate the code to analyze its coverage. This analysis is usually done using specific tools (e.g., Eclema). It is worth mentioning that it is important to define a coverage objective to perform the comparison. A good practice is to use continuous integration to automate the process of verification of code coverage	Development Team	Can be performed with a certain frequency (e.g., whenever a new code is uploaded into the repository)
		Change the test process used by team (if the team has one)	In order to systematize the activities for analyzing the test coverage	Changes should be made to the activities of the existing process	**Test planning:** Define the objective of code coverage **Test specification:** Register the identified coverage **Performing the tests:** Provide feedback about the identified tests coverage at runtime **Test Evaluation and reports:** Include achieved coverage in the report **Monitoring, control and replanning:** Collect and analyze code coverage for eventual intervention	Project manager or someone responsible for the test process management activity	To be defined by the team. (It is suggested that it be done periodically when the need for changes in the process is detected.)

Fonte: The Author

coverage of the catalog content - including all statements, descriptions, and subcategories - under the perspective of practitioners.

Focus group is an inexpensive and powerful approach from the social sciences largely used in human-computer interaction experimental and empirical researches to help generating a deeper and more nuanced understanding of issue (LAZAR *et al.*, 2017), (SHNEIDERMAN *et al.*, 2016). Hence, to capture impressions of potential catalog users and elicit their suggestions, reactions, frustrations, and fears, three sessions of focus groups were convened.

Based on the guide proposed by Tynan (TYNAN; DRAYTON, 1988), the focus group was planned, piloted, and conducted by an experienced moderator, together with two assistant observers who took structured notes. TestDCat Catalog was used as a discussion guide, as all catalog sections and actions were systematically discussed in three focus group sessions, which lasted from 2-3 hours each. Each participant received a hard copy of TestDCat Catalog, some Post-its, and pens, and was encouraged to make annotations and suggestions as the discussions progressed. Due to the long sessions, the typical question and answer style script was combined with prioritization and summary exercises. Additionally, refreshments were made available, and regular comfort breaks were offered to the participants. Figure 16 shows the environment in which the focus groups were held.

As summarized in Table 7, five participants were carefully selected according to their expertise, background, and availability. The focus group sessions gathered practitioners with different types of expertise within the software development process as well as various levels of

Figura 16 – Focus group session

Fonte: The Author

experience. They also had different levels of knowledge about the concept of technical debt. The goal was to obtain insights and feedback about the utility, clarity, and applicability of TestDCat Catalog to practitioners in different stages of their careers. Although opinions differ on optimal sizes for focus groups (LAZAR *et al.*, 2017), smaller groups are more appropriate to produce deeper and more fruitful discussions (KRUEGER; CASEY, 2002). The variety of participants provided a broad range of viewpoints and insights among peers with whom participants shared a common background: the knowledge and practical experience about software testing, and the need to manage *Technical Debts* in some level.

Tabela 7 – Focus group session

ID	SPECIALTY	EXPERIENCE IN THE FIELD	CURRENT POSITION	LEVEL OF UNDERSTANDING ABOUT TECHNICAL DEBT
P01	Software testing	3 years	Junior Test Analyst	I'm familiar with the concept
P02	SW development and testing	13 years	Senior Systems Analyst / Technical Leader	
P03	SW development and testing	1,5 year	Junior Systems Analyst	I've applied strategies to manage TDs
P04	Software testing	2 years	Software testing researcher	I'm familiar with the concept
P05	Software development	4 years	Mid-level Systems Analyst	I'm not familiar with the concept

During the evaluation session, for each Test Debt subtype, the moderator promoted discussions on every component (5W1H) of the listed actions. Then, in a round table, the focus group participants commented on each other's point of view, often challenging each other's motives and actions, and relating their own experiences to the catalog descriptions. This procedure allowed participants to review each action and its steps thoroughly. Also, they wrote down questions and suggestions for improvements.

A flip chart was available so that the moderator could summarize the identified drawbacks and advantages, suggestions for improvements, and doubts that were common to all participants. The moderator would sum up essential points at convenient times, making sure participants had understood them. After discussing an action, the mediator asked whether the participants indicated that the action should be kept in the catalog, undergo changes, or whether they felt it was not appropriate and should leave.

When all actions of a subtype category were discussed, participants individually used a 5-point Likert rating scale to assess the following criteria: applicability to the full range of intended uses, concreteness, clarity, ease of understanding, ease of use, impartiality, and relevance to the context. Such criteria were adapted from those proposed to assess the quality of evaluation checklists in a particular area (STUFFLEBEAM, 2000).

Figure 17(a) shows the results obtained about the actions present in the **identification activity**. All participants agreed or totally agreed with the criteria Applicability to intended uses. In Concreteness, there was one Neutral answer and the other four participants agreed or strongly agreed with the criteria. Regarding Clarity, the answers varied between Neutral (2) and Agree (3). In Ease of understanding, two participants agreed and three remained neutral. In Ease of use, one participant disagreed, but the rest agreed or strongly agreed with the criteria. All the participants agreed or strongly agreed with the criteria Impartiality and Relevance to the context.

Figure 17(b), presents the results about the actions present in the **measurement activity** are shown. About the criteria Applicability to intended uses, Disagree, Agree and Strongly Agree were chosen once each, while two participants remained neutral. About Concreteness, there were two Neutral answers and the other three participants agreed with the criteria. Regarding Clarity, two participants answered that they disagreed, while two answered Neutral and one chose Agree. About Ease of understanding and Ease of use, the answers varied between Neutral (3 and 1, respectively), and Agree (2 and 4, respectively). All the participants agreed or totally agreed with the criteria Impartiality. Finally, regarding the criteria Relevance to the

context, one participant remained neutral, while the others answered that they agree or totally agree.

Through Figure 17(c), we can see the results about the actions present in the **prioritization activity**. All participants agreed or totally agreed with the criteria Applicability to intended uses. In Concreteness, there was one Neutral answer and the other four participants agreed or strongly agreed with the criteria. In Ease of understanding, one participant disagreed and four agreed with the criteria. All the participants agreed or strongly agreed with the criteria Ease of use. About the criteria Impartiality, Agree and Strongly Agree were chosen by four participants, while the other participant remained neutral. All the participants agreed or strongly agreed with the criteria Relevance to the context.

Figure 17(d) shows the results obtained about the actions present in the **communication activity**. In the criteria Applicability to intended uses, Disagree, Neutral and Strongly Agree were chosen once each, while two participants have chosen Agree. In Concreteness, there were two Neutral answers and the other three participants agreed with the criteria. About Clarity and Ease of understanding, the answers were the same: one Disagree, one Neutral an three Agree. All the participants agreed with the criteria Ease of use. In Impartiality ans Relevance to the context, the answers varied between Agree (3 and 2, respectively), and Strongly Agree (2 and 3, respectively).

In Figure 17(e) the results about the actions present in the **monitoring activity** are shown. Regarding Applicability to intended uses, Disagree, Neutral and Agree were chosen once each, while two participants have chosen Strongly Agree. All the participants agreed or totally agreed with the criteria Concreteness. About Clarity, two participants agreed, while Disagree, Neutral and Strongly Agree were chosen once each. Regarding Ease of understanding two participants remained neutral, while Disagree, Agree and Strongly Agree were chosen once each. In Ease of use, one participant remained neutral, while the others answered that they agree or totally agree. All the participants agreed or totally agreed with the criteria Impartiality and Relevance to the context.

Figure 17(f) shows the results obtained about the actions present in the **documentation activity**. In the criteria Applicability to intended uses, Neutral was chosen once, while the other four participants have chosen Strongly Agree. About Concreteness, there were two Neutral answers and the other three participants agreed or strongly agreed with the criteria. About Clarity, two participants remained neutral, while Disagree, Agree and Strongly Agree

were chosen once each. Regarding the criteria Ease of understanding, one participant answered that he/she disagreed, while two chose Neutral and the other two answered that they totally agree. Regarding Ease of use, one participant remained neutral, and the other four agreed or totally agreed with the criteria. All the participants agreed or totally agreed with the criteria Impartiality and Relevance to the context

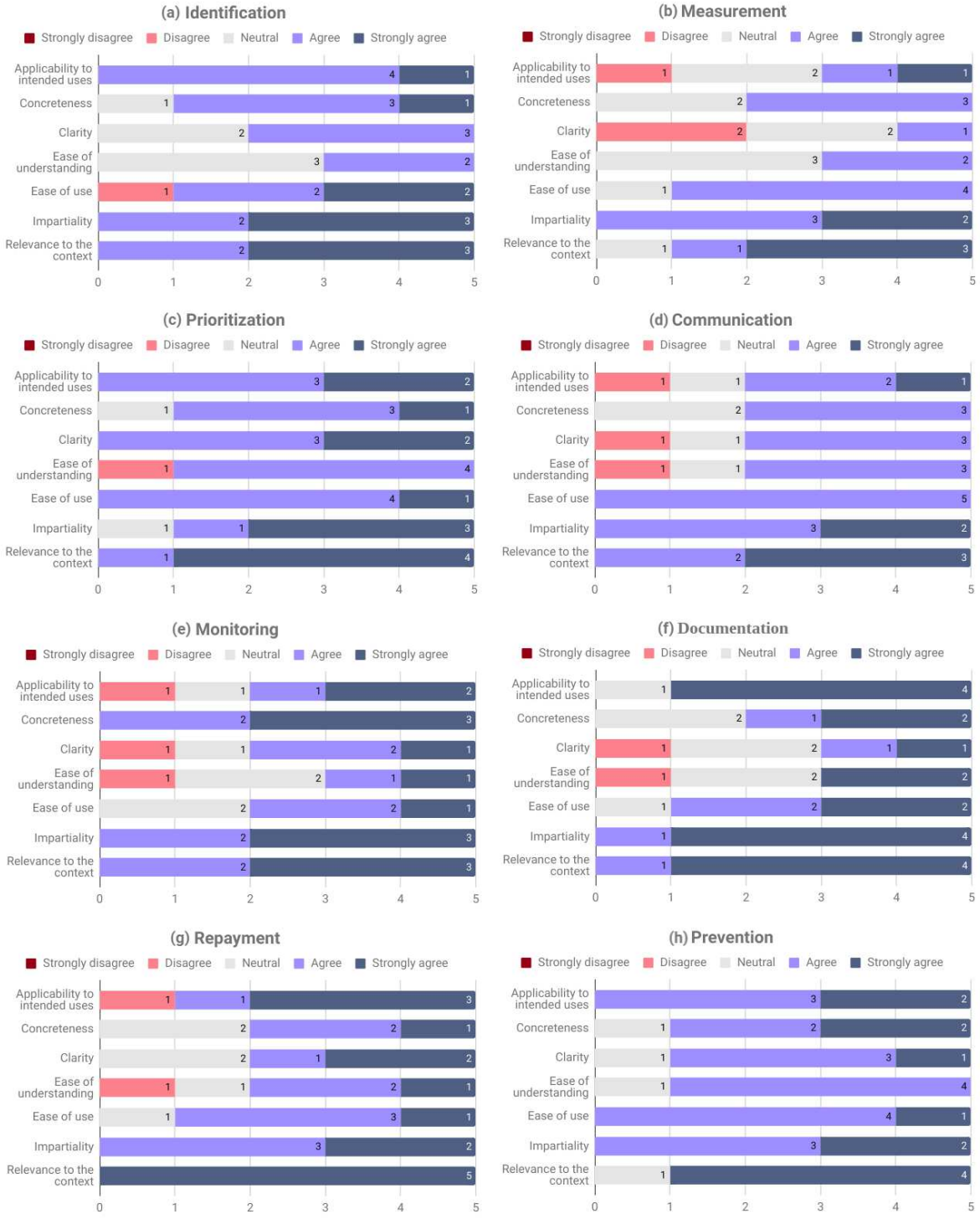
Through Figure 17(g), the results about the actions present in the **repayment activity** are shown. About the criteria Applicability to intended uses, Disagree and Agree were chosen once each, while three participants chose Strongly Agree. About Concreteness, there were two Neutral answers and the other three participants agreed or strongly agreed with the criteria. Regarding Clarity, two participants answered Neutral, one chose Agree and two Strongly Agree. Regarding the criteria Ease of understanding, one participant answered that he/she disagreed, one chose Neutral and the other three answered that they agree or totally agree. Regarding Ease of use, one participant remained neutral, and the other four agreed or totally agreed with the criteria. All the participants agreed or totally agreed with the criteria Impartiality. Finally, regarding the criteria Relevance to the context, all the participants answered that they totally agree.

Figure 17(h) presents the results about the actions present in the **prevention activity** are shown. All participants agreed or totally agreed with the criteria Applicability to intended uses. In Concreteness, Clarity and Ease of Understanding, there was one Neutral answer and the other four participants agreed or strongly agreed with each criteria. All the participants agreed or totally agreed with the criteria Ease of use and Impartiality. Finally, regarding Relevance to the context, one participant remained neutral, and the other four totally agreed with the criteria.

In a nutshell, all criteria had more than 50% agreement. The criteria “Applicability to all intended uses” obtained 77.5% agreement, and “Relevance to the context” obtained more than 95%, which reflects that the participants believe that the catalog can indeed be used to assist in the management of *Test Debts*.

The criteria “Clarity” and “Ease of understanding” were the ones that had the highest rate of disagreement with 12.5%. In the individual analysis of the answers, we noticed that the majority of the participants who had these opinions had less work experience. However, as the use of the catalog must be made by all levels of expertise, these are points of improvement that must be addressed. It is worth mentioning that during the evaluation, the participants made many observations in order to improve these criteria, so we believe that when considering these new improvements, these rates tend to decrease.

Figura 17 – Results of the evaluation

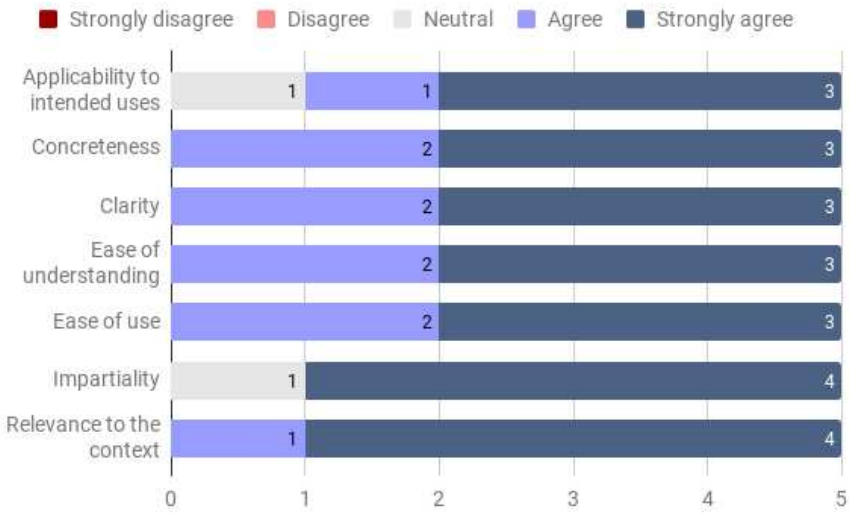


Fonte: The author

After the evaluation of the catalog actions, participants were invited to evaluate the website that presents the catalog and allows interactive use. Figure 18 presents the results of the questionnaire that contained the same criteria as used in the previous questionnaires. Regarding the criteria Applicability to intended uses, the answers varied between Neutral (1), Agree (2), and strongly agree (3). All participants agreed or totally agreed with the criterias Concreteness,

Clarity, Ease of understanding, Ease of use and Relevance to the context. Regarding the criteria Impartiality, the answers varied between Neutral (1) and strongly agree (4)

Figura 18 – Website Evaluation



Fonte: The Author

By the end of the sessions, the moderator and observers conducted a debriefing and led a summary exercise to gather key themes and check for further understanding of participants, moderation, and observers' notes. Besides, they identified and categorized note themes, hunches, interpretations, and ideas. Then, they labeled, compared, and contrasted information from field notes and other materials.

A total of 63 actions were evaluated and analyzed. Overall, participants suggested to remove an action related to the *Communication* activity and add a new action to the *Identification* activity. Relevant questions emerged regarding *Test coverage and code coverage* as follows: "Is it separate or is it the same thing? And how to identify the ideal version?". The main improvements suggested were "Categorize separating tests of what is manual and automated" and "Standardize the terms (especially for the columns *Who* and *Where*)". Finally, some of the changes requested included: "Put a glossary with test area terms and technical debt (TD) for people who are not very experienced in testing and TD", "As it is dependent on some preconditions, it would be better to clearly separate what is precondition and what is the action in fact", "Use the term iteration instead of Sprint", and "Keep the term follow-up meetings in all actions that mention holding meetings".

The evaluation proved to be very useful, despite requiring much effort. Several improvements were suggested and only one of the actions was considered inadequate for the

proper purpose. In a nutshell, the proposed catalog got good results.

5.6 TestDCat Catalog

As stated in Section 5.5, in the second evaluation, participants reviewed each action and made notes with observations and suggestions for changes. Also, two assistant observers took notes on the observations made.

Based on the analysis of the annotations carried out, improvements have been made to the existing catalog and generated a new version called TestDCat 3.0. The structure of the catalog remains the same, using the 5W1H model, and the data is categorized according to TD management activities and subtypes of test debt, both mapped by Li et al. (LI *et al.*, 2015). It is worth remembering that the test debt subtypes are the possible causes of this kind of debt (Chapter 2). In addition to the subtypes identified by Li et al., two other subtypes have been identified: *Inadequate equipment* and *Inadequate allocation*. Regarding the subtype of *Inadequate Equipment* test debt, the incorrect use of equipment can impact the quality of the tests performed, generating, for example, false positives. *Inadequate allocations* can impact the results of tests performed, when, for example, professionals who do not have the necessary expertise to perform tests on a given demand are allocated, and critical defects are not detected. The other test debt subtypes are described in Chapter 2.

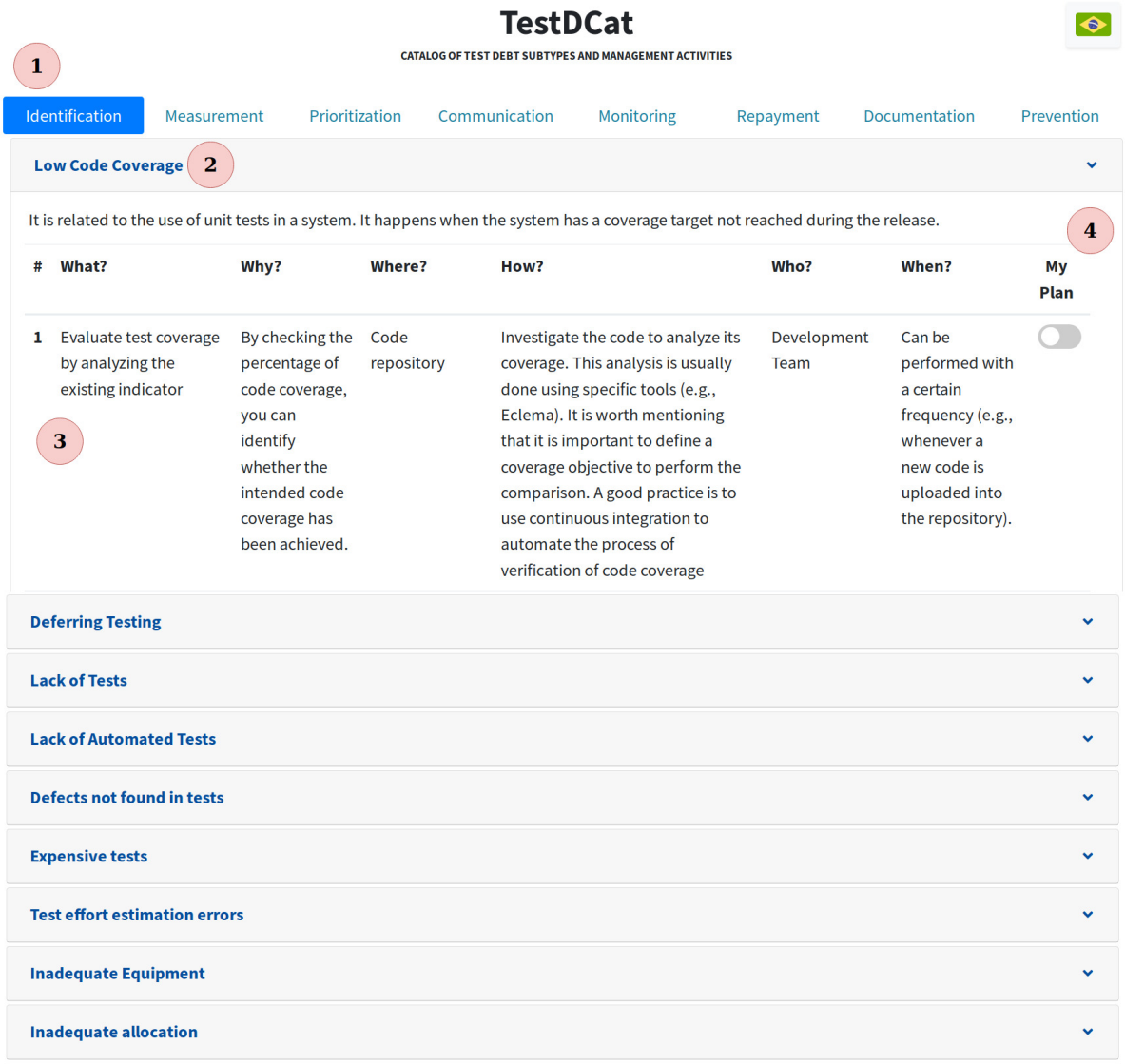
Figure 19 presents the catalog, emphasizing with listed circles parts of the catalog. Circle number 1 emphasizes technical debt management activities: Identification, Measurement, Prioritization, Communication, Monitoring, Repayment, Documentation, and Prevention. By clicking on any of these activities, subtypes of *Test Debts* and its related actions are presented. In this example, the "Identification" activity was selected.

The circle number two presents the subtypes of *Test Debts*: Low code coverage, Deferring testing, Lack of tests, Lack of tests automation, Defects not found in tests, Expensive tests, Test effort estimation errors, Inadequate equipment, and Inadequate allocation. In this catalog, we present a set of actions for each subtype. For example, after clicking on the "Identification" activity, the user can choose which subtype he wants to handle. In this example, we choose "Low Code Coverage", which has two related actions.

Circle number three brings forward the actions identified through semi-structured interviews conducted. They are following the 5W1H model. In this case, these are the suggested actions to help catalog users to identify *Test Debts* caused by "Low Code Coverage".

Actions can be used together or individually. Thus, in circle number four, we present the functionality in which the user can select which actions are most related to their context. After selecting a custom action plan will be created. Finally, it is possible to print or download this plan.

Figura 19 – *TestDCat latest version* structure



Fonte: The Author

A total of 63 actions were catalogued and all the catalog's information can be viewed on the TestDCat Catalog's website [6].

[6] TestDCat Catalog website: https://great-ufc.github.io/TestDCat/index.html

5.7 Examples of catalog actions

Only the identification and repayment activities have specific actions for each subtype of test debt. For the other activities, the described actions could be applied for all subtypes.

5.7.1 Identification

The management activity *Identification* refers to the tasks performed to identify technical debts during the development of software. At the end of this activity, the list of TDs identified in the project is expected.

Table 8 presents two identification actions. In the first one, it is expected to identify subtypes of test debts *Lack of tests* by analyzing the list of versions of the software in order to find indications of non-performance of tests in any of these versions analyzed. In the second action, it is expected that, by analyzing feedback from users, possible defects that were not detected at the time of testing will be identified.

Tabela 8 – Example of Identification Actions

TD management activity	Test debt subtype	What?	Why?	Where?	How?	Who?	When?
Identification	Lack of Tests	Evaluate launch plan or list of software versions	When evaluating the launch plan or the list of software versions, you can check and identify the existence of versions that have been launched without testing	Launch plan or list of software versions	In the analysis, you must check whether there are indications that tests have been carried out for the version being analyzed. If there are versions that could not identify the existence of some tests, it's likely that there was a lack of tests in that version.	Development and test team	Periodically or whenever new versions have been released for testing.
	Defects not found in tests	Analyze feedback from end users	Users may submit defects found by them via e-mail or via product reviews in any app/systems store and social networks	E-mail, product reviews and social networks	Analyze e-mails and user reviews of released products to identify possible defects not found at development/test team	Customer Support, development/test team	Periodic reports or from the analysis of some specific metrics (e.g., bugs found by the end user)

Fonte: The Author

5.7.2 Measurement

Measurement management activity aims to quantify the benefit and cost of known technical debts. This estimate can be performed for an individual TD or for the entire system in order to identify the level of TDs in the system.

Table 9 presents an example of action to be taken in order to quantify the cost/benefit ratio of identified debts. Some examples of the cost calculation described in this action are: (i) Effort required to document and perform manual tests; (ii) Infrastructure required to perform the tests; and (iii) Impact of changes in other system test cases.

Tabela 9 – Example of Measurement Action

TD management activity	Test debt subtype	What?	Why?	Where?	How?	Who?	When?
Measurement	All Test Debt Subtypes	Quantify the cost/benefit of acquiring and paying a known test debt	It is necessary to measure the cost/benefit of acquiring and paying a debt in order to assist in making a decision on the payment of the debt.	Depending on the technique used to quantify the TD (e.g., test case, bugs list)	The benefit of acquiring the TD and the cost (i.e. the sum of the actual effort to pay the TD (principal) and the most needed effort due to postponement of payment (interest)) must be calculated. The benefit calculation normally takes into account the time/effort gained from acquiring DT. This analysis can be done through mathematical calculation based on the test cases that need to be performed or through human estimation. The cost can be calculated taking into account the following points: - Necessary effort for the correction of bugs derived from the non-performance of tests - Necessary effort for documentation and manual test run - Effort required to implement automated testing - Necessary effort for maintenance of automated test scripts developed - analyze the feasibility of automating a particular test case - Infrastructure required to perform the tests - Impact of changing test cases on other system test cases Effort quantification can be performed through human estimation or mathematical model (e.g., Test Case Point analysis). The cost can also be analyzed in a more subjective way, for example: - Loss of user confidence - Loss of customer loyalty - Increased number of complaints about the product with service requests - Loss of brand credibility	Technical leader, development and test team	To be defined by the team. It is important to keep visible the need to measure the identified TDs (e.g., activity of measuring in a list of activities).

Fonte: The Author

5.7.3 Prioritization

The *Prioritization* management activity is intended to rank identified technical debts to assist in defining which should be repaid first and which may be postponed to another time.

Table 10 presents two actions for *Prioritization* activity. It is worth mentioning that these actions may be applied to all subtypes of test debts. The first action is related to the analysis of the cost/benefit ratio performed in the *Measurement* activity. Based on this analysis, it is possible to realize the prioritization. The second action is related to the relevance of the functionalities that have known TDs. The debts that are related to the most relevant functionalities should be prioritized.

5.7.4 Communication

The communication management activity aims to inform all stakeholders of the debts identified. By conducting this activity, it is expected that project members will be aware of the existence of the identified TD and will be able to take the necessary actions, if pertinent.

Table 11 presents two examples of actions. The first action concerns the use of follow-

Tabela 10 – Example of Prioritization Actions

TD management activity	Test debt subtype	What?	Why?	Where?	How?	Who?	When?
Prioritization	All Test Debt Subtypes	Cost/benefit analysis of the acquisition and payment of a debt	By analyzing the cost/benefit it is possible to identify which debit should be paid first	Identified debts list	From the list of identified debts, the cost/benefit ratio of the debts defined in the measurement activity should be analyzed. Debts with the highest repayment rates (i.e. that the acquisition cost is high and the acquisition benefit is low) should be prioritized to be paid first	Project manager	Depends on the team's strategy, but can be done in planning the activities to be developed in the release
		Analysis of the relevance of software functionalities that have identified debts	When prioritizing TDs related to the most important functionalities, the effort is focused primarily on what is most relevant for the customer/user	Debts list and functionalities list and traceability between functionalities and Debts identified	First, it is necessary to have the list of system functionalities with the level of importance of each functionality. Some ways to measure the importance of the functionalities are: - those that have the greatest Impact for the client or users - those that are most commonly used by users (by analyzing logs to identify how the user uses the system) - the ones with the most related bugs - those with a high risk of finding defects - those with already identified defects that are more critical - those with many links to other system features. With this prioritized list of functionalities, the list of identified TDs should prioritize the TDs related to the most important functionalities. It is worth mentioning that it is necessary to have a traceability between the identified TDs and the system functionalities.	Project manager	Depends on the team's strategy, but can be done in planning the activities to be developed in the release

Fonte: The Author

up meetings to report identified debts. The second action is about the use of communication tools (e.g., Skype, e-mail) to report the existence of debts.

Tabela 11 – Example of Communication Actions

TD management activity	Test debt subtype	What?	Why?	Where?	How?	Who?	When?
Communication	All Test Debt Subtypes	Conduct follow-up meetings	Follow-up meetings happens when all or part of the team is together and can be a good opportunity to communicate the debts identified	Follow-up meetings	Take advantage of follow-up meetings (e.g., daily and retrospective meetings) to pass on the identified debts to those involved. At this point, the team can even discuss these debts a little more and make decisions about them. The daily meeting is one of the most suitable for this quick communication, because in addition to taking place every day, a large part of the team participates	All the team	Depends on the frequency of the meetings.
		Use communication tools	Tools for instant communication can be very effective for communicating the existence of debts identified	Communication tools	Communicate debts using communication tools (e.g., Skype, email). This communication should contain the information from the debts identified. It is important to ensure that the information from the debts is sent to the right stakeholders.	All the team	Whenever a debt is identified

Fonte: The Author

5.7.5 Monitoring

The *Monitoring* management activity aims to monitor the cost/benefit ratio of each identified debt in order to notice changes in this ratio. This monitoring is important because test

debts that at the time of identification are not relevant to the repayment may become relevant due, for example, to a change in the importance of the functionality in which the test debt was identified.

Table 12 presents some of the examples of the actiond defined in the catalog for the "Monitoring"activity. The first action is related to the monitoring of the cost/benefit ratio of a known debt. This analysis can use the same techniques used to measure the TD and should be carried out periodically in order to identify this change. The second action is related to the definition and monitoring of triggers. When a particular trigger (e.g., change in the complexity of a feature that contains a debt and discontinuity of a product) is identified, a new analysis of the cost/benefit ratio of the debt should be performed to, if necessary, carry out actions in this regard.

Tabela 12 – Example of Monitoring Actions

TD management activity	Test debt subtype	What?	Why?	Where?	How?	Who?	When?
Monitoring	All Test Debt Subtypes	Monitor changes in the cost/benefit ratio of the identified debt	When the cost/benefit ratio of identified debts is monitored, it is possible to verify a change in this ratio in comparison with the measure carried out previously	List of Identified debts	On the basis of the list of identified debts and the measurements previously taken, the debts must be checked periodically to observe possible changes in the cost/benefit ratio of that debt. The way to check the change in this relationship can be the same as that used in the measurement activity of that debt; for this purpose, the measurement method must be documented in the debt. A good practice is to place the measurements every time a new release is planned, since it is possible to add the payment for the debt if the new measurement influences this decision	Project manager and test leader	Periodically or whenever new versions have been released for testing
		Monitor triggers	When using triggers it is possible to identify when there are changes in the cost/benefit ratio of the debt	List of Identified debts	Use triggers to identify possible changes in the cost/benefit ratio of an identified TD. Some triggers that can be used are: - Change in the complexity of a particular functionality that the debt has been identified for - Increase in the number of identified defects in a functionality that had identified debts - Discontinuity of the product or of a specific functionality that had debt identified - Increased use or importance of a functionality that had an identified TD - Development of new functionalities that depend on functionalities that already have known debts	Project manager and test leader	The monitoring of these triggers must be constant

Fonte: The Author

5.7.6 Repayment

The repayment activity aims to resolve or mitigate a known debt. It is performed when the team believes that failure to pay the debt can cause a major impact on the maintenance of the software under development.

For the repayment activity, actions were generated for all test debt subtypes. Table

13 presents examples of repayment actions for the subtypes *Lack of tests* and *Defects not found in tests*. The first action is related to the elaboration and performance of test cases of releases that have not been tested. For this subtype of test debt it is very important that the payment of the debt is made with high priority, because the release that has not been tested may have many defects that have not yet been identified. The second action is, by analyzing the identified defects, to insert new test cases to cover the defect found or to make changes in existing test cases in order to expand or improve them to cover the defect under analysis.

Tabela 13 – Example of Repayment Actions

TD management activity	Test debt subtype	What?	Why?	Where?	How?	Who?	When?
Repayment	Lack of Tests	Elaborate and perform tests for releases that were not tested	In order to pay the debts related to the lack of tests	System test cases	Elaborate and execute absent tests from previous releases that were prioritized to be done in the current cycle. A good practice is to pay for this type of debt as soon as possible, as relevant defects may no longer be identified due to the lack of tests. One way to do this is to always allocate extra time within the current testing cycle to pay for debts for the absence of previously identified tests	Development/test team	Based on debts prioritization decision
	Defects not found in tests	Change test cases by analyzing defects	In order to identify the need for new tests to be implemented or executed	System defect list	Based on the list of identified defects, you should analyze whether the defect occurred due to a lack of specific tests to cover the defect in question or whether an update of existing test cases is necessary. In the case of lack of tests, new cases of tests that cover the identified defect must be elaborated. In the cases of immature tests cases, the existing cases must be amplified or improved in order to cover the defect in question or similar	Development/test team	Based on debts prioritization decision

Fonte: The Author

5.7.7 Documentation

The documentation management activity is intended to document identified TDs for the knowledge of all stakeholders and for future consultation. Tabela 14 presents an example of an action that aims to set a standard for documentation of identified debts. Following a standard for documenting debts is essential to ensure that important information is not neglected and is included in the documentation of a debt. This documentation can be accessed when necessary and its content updated if relevant.

Tabela 14 – Example of Documentation Actions

TD management activity	Test debt subtype	What?	Why?	Where?	How?	Who?	When?
Documentation	All Test Debt Subtypes	Standardize debit documentation	Using a standard facilitates documentation and understanding of the identified debt	Project tracking tool, WIKI and e-mail	Standardize debt documentation using information that will be useful for conducting debt management activities. Some of the information that can be used are: - Identifier - Responsible - Status (e.g., to do, doing, ready) - Location - Description - Type (e.g., low indicator coverage, no test) - Cost/benefit ratio - Prioritization - Relationship with other debts A good practice is to enter as much information as possible, because it facilitates the analysis of the identified debt.	All the team	Where a debt is identified or changed

Fonte: The Author

5.7.8 Prevention

This management activity aims to prevent the occurrence of TDs. Table 15 presents two actions related to this activity. The first one is related to the presentation of the debts already identified to stakeholders in order to prevent new debts of the same type from being acquired again. The second one is to add review to the test cases elaborated with the purpose of identifying improvements and, thus, avoiding the occurrence of new debts.

Tabela 15 – Example of Prevention Actions

TD management activity	Test debt subtype	What?	Why?	Where?	How?	Who?	When?
Prevention	All Test Debt Subtypes	Present already identified debts	In order for the team to be aware of existing debts and avoid incurring new debt of the same type	Debt presentation meetings	Present to the team the debts already known so that they understand how they were caused and how they can prevent similar debts from being acquired	Project manager, test leader	To be defined by the team
		Review elaborated test cases	In order to identify possible improvements or errors	List of system test cases	In addition to the person who will elaborate the test, a second person should review the test in order to identify if it is adequate or if it contemplates alternative paths to the main flow of functionality. Also, a review about the established test estimatation can be performed. A second person can identify errors in this estimatation and propose changes.	Development/test team	Whenever there is a test creation or change

Fonte: The Author

5.7.9 Actions to improve the test process

In addition to the specific actions for each activity, and subtype of test debt, it was implemented in TestDCat, actions with suggestions for improvements in the test process. These improvements are carried out in the existing activities of the test process and are designed to

systematize the test debt management actions. Table 16 presents some examples of this type of action in different management activities and test debt subtypes.

Tabela 16 – Example of Test process adjustments actions

TD management activity	Test debt subtype	What?	Why?	Where?	How?	Who?	When?
Identification	Inadequate allocation	Changes in the test process (if the team has an existing test process)	In order to systematize the activities to identify situations of inadequate allocation	The changes must be made in the activities of the existing process	**Test Planning:** Delegate tasks observing the team's expertise to perform the tasks. **Test Design:** Report critical delays for the delivery of the system under test. **Test execution:** - Report test failures. - Report critical delays for the delivery of the system under test. **Test monitoring:** - Collect and analyze reported failures. - Collect and analyze reported delays. This analysis should verify if the reasons for the bugs or delays were due to an inadequate allocation of the team that would test the software.	Project manager	Carried out periodically as defined by stakeholders in the project
Repayment	Lack of tests		In order to systematize the activities for payment of debts related to the lack of tests		**Test planning:** Define which tests from previous releases that were not performed will be prepared and executed in the current tests. This decision must be made according to the prioritization previously carried out. **Test design:** Elaborate tests defined in planning **Test execution:** - Carry out the tests defined in planning - Report the actual cost of carrying out the activity **Test closure:** Update the cost/benefit analysis of the debt to serve as the historical basis for other similar debts.		

Fonte: The Author

5.8 How to use the TestDCat

The catalog proposed in this work has three main elements: (i) Technical debt management activities; (ii) Subtypes of test debts; and (iii) Actions for the management of test debts. Any of these elements can be used as a guide to using the catalog. The user can take the management activities as a guide and perform all the actions of the activity that he/she chooses (for example, he/she may want to perform the identification of all the test debt subtypes without performing the measurement activity). The user can also choose the test debt subtypes as a guide. Depending on the subtype chosen, he defines which management activities he/she wants to use (for example, he/she can choose the subtype "Lack of tests"and make the Identification actions for this subtype).

5.8.1 From TD management activities

There are eight technical debt management activities in the catalog: Identification, Measurement, Prioritization, Communication, Monitoring, Repayment, Documentation, and

Prevention. Some of these activities are dependent on each other, for example, except for the prevention activity, all of the others need the identification activity to be performed before them.

Catalog users can choose specific activities to analyze the test debt management actions related to that activity (respecting the dependencies) or perform all management activities together. Once the management activities have been chosen, users can decide which of the actions defined in the catalog best suits their needs.

5.8.2 From subtypes of test debts

Another way to use the catalog is from the test debt subtypes. You can choose which subtype is most important to him/her. From this, choose which TD management activities and, consequently, the actions he/she would like to perform for these chosen subtypes.

Users can choose one or more subtypes to manage. The subtypes are independent and can be used separately. It is worth mentioning that, as stated in the section 5.7, in some management activities, the actions can be applied to all subtypes.

5.9 Conclusion

This chapter detailed the steps for building the catalog proposed in this master's thesis. First, the literature review was presented. This review occurred by analyzing articles in the literature that presented similar problems to those presented in this work.

The conduction of semi-structured interviews was also presented in this chapter. The interviews were divided into two parts and provided a portion of the necessary inputs for catalog building. First, the interviews were planned with the definition of a protocol with the steps for their realization. The profile of the interview participants was also detailed. Also, it was presented how the data from the interviews were analyzed.

The structure of the first version of the catalogue and the evaluation carried out for this version were also presented. This first evaluation intended to get answers to the following: *Does the TestDCat catalog have clear and enough information to support the management of Test Debts?*

This chapter also described the new structure of the catalog. This new version was defined by analyzing the results of the second series of interviews, and the evaluation carried out with the first version. In this new version, the 5W1H model was already used to describe

the actions to manage test debts. Also, we present the second evaluation performed. In this new evaluation, a focus group was made, whose objective was to analyze each action contained in the catalog and identify possible improvements in these actions.

The latest version of TestDcat was also presented. This version is structured into three elements: TD management activities, test debt subtypes, and actions. These actions are organized using the 5W1H model (a model commonly used to conduct action plans) and aim to serve as a guideline for performing the TD management activity to which it is related. Examples of actions were also presented for each of the defined TD management activities. Besides, an example of action related to changes in the existing testing process was presented. This type of action aims to improve the testing processes used in teams to manage test debts. Finally, how TestDcat can be used were presented, using TD management activities or test debt subtypes as a starting point for using the catalog.

The next chapter presents the case study performed with the last version of the catalog. This case study was conducted on industry projects and aims to answer the following questions: (i) The use of the catalog assists in the management of test debts?; and (ii) How easy is TestDCat to use?.

6 CASE STUDY

Following the research methodology used in this master's thesis (Chapter 1, Section 1.4), a dynamic validation was planned and conducted. This validation was performed with a case study at GREat Test Factory (GTF) and the goal was the use of the catalog in real projects with the industry. This chapter presents the case study and is divided as follows: Section 6.1 introduces the description and planning of the case study, Section 6.3 presents the execution of the case study, Section 6.4 details the collected results, and Section 6.5 provides a discussion about the results.

6.1 Case study design

Case studies are commonly used to study phenomena that occur in their natural context (RUNESON; HÖST, 2009). So, this master's thesis conducted a case study in a test factory to evaluate the use of the catalog for managing subtypes of test debts in real industry projects. The case study was based on the guide proposed by Runeson et al. (RUNESON; HÖST, 2009), which is a relevant guide for conducting case studies in software engineering.

6.1.1 Definition of goals and questions

The case study was conducted with the intention of measuring the impact of the use of a catalog of test debts in a Test Factory that operates in testing activities of real projects with the industry. To do this, it was necessary to understand the knowledge of the GREat Test Factory (GTF) participants on test debts, to measure the current status of the occurrence of test debts in GTF and to investigate the impact on using a catalog of test debts. The questions that came from the objective and which should be answered with the conducting of the case study are:

a) **RQ1.** *The use of the catalog assists in the management of test debts?* This question aimed to identify whether the use of the catalog really helped in the management of the test debt subtypes. As a collection method, in addition to the application of a survey, tools were also investigated to monitor the test activities carried out by the FTG, as well as the verification of indicators collected after the use of the catalog. Table 17 summarizes these metrics and their respective calculation formulas.

b) **RQ2.** *How easy is TestDCat to use?* This question was designed to verify the

perception of GTF participants regarding the use of TestDCat. As a collection method, the SUS questionnaire (BROOKE *et al.*, 1996) was used to identify the level of usability of the catalog.

Tabela 17 – Metrics used in the case study

GROUP	METRIC	CALCULATION	INTERPRETATION
Absolute Numbers	Total number of test cases	Total Number	N/A
	Number of test cases passed		
	Number of test cases failed		
	Number of test cases blocked		
	Number of defects found (bugs)		
	Number of planned test hours		
	Number of actual test hours		
	Number of bugs found after shipping		
Test Effort	Number of tests run per time period	Number of tests run/total time	The higher the result, the more test cases were performed in total time
	Test design efficiency	Number of tests designed/ total time	The higher the result, the more test cases were elaborated in total time
Test Effectiveness	Test Efficiency	[DT / (DT + DU)] * 100 (DT = Defect by Testing team & Development team and DU = Defect by the customer)	The higher the value of the result, the better the set of tests that were performed on demand. This is due to the greater amount of defects found by the test team compared to the defects found by the customer or end-user.
Test Economics Metrics	Schedule Variance	[(Actual effort – estimated effort) / estimated effort] * 100	From the result of the formula it can be concluded: (i) If the value is positive, it means that the actual effort was higher than the estimated effort; (ii) If the value is negative, the actual effort was less than estimated; (iii) If the result is zero, it means that the estimate was made correctly.
Test Debt Metrics	Number of test debts (divided by subtype)	Total Number	N/A

Fonte: The Author

6.1.2 Study Objects

The objects were three software products developed in partnership with the industry. The selected products were the ones that demanded tests for the GREat Test Factory during the period of the case study.

Table 18 presents the products, the platforms for which they were developed, the number of releases launched during the period of the case study, the number of code lines (Loc) of the software product as well as the number of test cases already elaborated in the GTF for each product.The names and versions were changed for reasons of confidentiality. A total of five applications of the catalog were analyzed.

The P1 product is a mobile application based on Android technology. It currently has 366.203 active users. The tests of this application are related to validation of the state machine that has many different states that need to be validated. In addition, it is necessary to perform the tests on several different versions of android to ensure the correct operation on these various

versions that the application should work.

The P2 product is a WEB tool used to support the internal processes of the company. This tool is used as a means to perform the activities of the company. Its malfunction impacts on all the work done in the company. Thus, the tests must be careful to ensure that this tool works properly.

Tabela 18 – Objects of the case study

Tool	Platform	#Releases	#Loc	#Test Cases
P1	Mobile	3	77.494	1.336
P2	Web	2	230.874	1.491

Fonte: The Author

6.2 Case study participants

The GTF team who participated in the case study is composed of five professionals, divided as follows: (i) One test leader, responsible for the creation and execution of scenarios and test cases, also monitors the execution of the activities performed and technically assists the team; (ii) Two test analysts, responsible for the creation and execution of scenarios and test cases; and (iii) Two testers, responsible for the execution of previously prepared test cases.

6.3 Execution of the case study

In order to conduct the case study, the catalog was applied in five test demands of the GTF, being three applications in the demands of project P1, and two applications in the demands of project P2. Before the beginning of the applications, a presentation of TestDCat was made in which the objective of the catalog was presented and how it could be used.

Catalog applications on P1 and P2 software products occurred as testing demands were requested to the GTF. The application of the catalog followed this step by step, defined by the researcher together with the GTF leader:

a) The GTF team meets to analyze the catalog and identify which subtypes of test debt would be managed in that demand. In addition, at this meeting, it is also defined which actions would be taken from each management activity of the previously chosen test debt subtype.

b) Analyze the chosen actions and adapt the activities of the test process established

in the GTF (Chapter 2, Section 2.1.1) in order to manage the chosen test debt subtypes.

c) Perform the actions together with the activities of the GTF testing process.

d) Collect indicators to identify test debts and indicators already collected in the GTF (e.g., Number of bugs identified by demand, Number of test cases by demand, Planned effort versus actual effort). The collected indicators will be detailed in Section 6.4.

The main necessary change in the GTF test process for the use of the catalog was to insert the TestDCat analysis and define the actions for the management of the test debt subtypes in the meeting to define the current demand test scope. The adaptations of the activities performed in the GTF were carried out according to the choice of the test debt subtype that would be managed in the current test demand. The adaptations made to each demand will be presented as follows.

6.3.1 First application

After the training on how to use TestDCat, the first application of the catalog was started in the test demand requested to GTF to perform the tests of a specific release of software product P1.

Starting the steps established for the application of the catalog in the realization of the test demands, the GTF team met to analyze the catalog and discuss which subtypes of the test debt would be managed in the release in question. Figure 20 shows a meeting held for discussion and analysis of the catalog.

For this first application, the team chose to deal, using the catalog, with the subtype of test debt *Defects not found in test*. This choice was due to the prioritization of the team in dealing with debts that could generate defects found by the client, because the problems found by the client would bring greater inconvenience to the GTF. In addition, as seen in the empirical study (Chapter 4), *Test Estimation Errors* was the first subtype of test debt identified in the GTF.

With the test debt subtype defined, the team analyzed and chose the actions they would take to manage the chosen test debt subtypes. According to the selected actions, it was necessary to add or change some activities of the testing process used in the GTF to perform the management of the test debt subtypes.

Table 19 presents a summary of the actions chosen in this first application of the

Figura 20 – TestDCat discussion and analysis meeting

Fonte: The Author

catalog. It can be seen that the team prioritized actions that were already fully or partially performed in GTF and that would have less impact on the activities already performed by the team.

Tabela 19 – First application of TestDCat

SELECTED TEST DEBT SUBTYPES							
1. Defects Not found in tests							
SELECTED ACTIONS							
IDENTIFICATION	MEASUREMENT	PRIORITIZATION	COMMUNICATION	MONITORING	REPAYMENT	DOCUMENTATION	PREVENTION
Analyze feedback from end-users	Quantify the cost/benefit of paying a known test debt	Cost/benefit analysis of the acquisition and payment of a debt	• Using Product Backlog • Conduct follow-up meetings	Monitor changes in the cost/benefit ratio of the identified debt	Change test cases by analyzing defects	Standardize debt documentation	Present already identified debts
REASONS FOR THE CHOICE							
• Affects the end-user directly • Adequate to the reality of the application that will be tested	In order to have an idea of the cost/benefit to correct the debts identified	Understand the importance of payment based on the cost/benefit analysis of the TD	It was chosen because it is similar to the activities that are already carried out in the GTF	Periodic verification has been chosen as it is the most appropriate in the context of the GTF	It is similar to the way the GTF team is already used to dealing with problems of this type	Facilitate understanding of TDs	Mitigate the events of identical or very similar TDs
CHANGES IN ACTIVITIES							
Add the activity of tracking app users' reports by analyzing sent emails or comments in the android app store in order to identify possible defects not found by the testing team	For benefit analysis, the impact of the defect found for the end-user was analyzed For cost analysis, it was calculated the necessary effort for the documentation and manual test run	Based on the list of identified debts, the cost/benefit ratio defined in the measurement activity was analyzed. Debts with the highest repayment rates were prioritized	• Insert the identified TDs into the product backlog • Take advantage of follow-up meetings (e.g., daily and retrospective meetings) to pass on the identified debts to those involved	Check the cost-benefit ratio of the test debts identified in order to note changes in that ratio	Create new test cases or improve existing cases in order to cover the defect in question or similar	The identified debts were registered in the documentation tool used in the GTF, following the standard: ID, description, location, cost/benefit ratio, prioritization and subtype of test debt.	Present to the team the debts already known so that they understand how they were caused and how they can prevent similar debts from being acquired

Fonte: The Author

6.3.2 Second application

The second application of the catalog still occurred in the project P1. At the catalog analysis meeting, the GTF team decided that they would manage the following subtypes of test debt in this session: (i) *Defects not found in test*; (i) *Lack of tests*; and (i) *Test effort estimation errors*. All these subtypes were identified in the empirical study carried out in the GTF, and it was known to the team that they frequently occurred in the GTF. Thus, the team decided to manage them in this second application of the catalog.

After defining the test debt subtypes, the team defined which actions would be performed to manage the chosen subtypes. Table 20 shows the actions selected for this new application. It is worth mentioning that the management activities that the actions were not changed in relation to the previous application were omitted from the table, but all the management activities were performed.

Tabela 20 – Second application of TestDCat

SELECTED TEST DEBT SUBTYPES				
1. Defects not found in tests 2. Lack of tests 3. Test effort estimation errors				
SELECTED ACTIONS				
IDENTIFICATION	PRIORITIZATION	COMMUNICATION	REPAYMENT	PREVENTION
To identify *Defects not found in tests*: • Analyze feedback from end-users To identify *Lack of tests*: • Compare different versions of the same software • Evaluate the launch plan or list of software versions To identify *Test effort estimation errors*: • Analyze reported effort	Analysis of the relevance of software functionalities that have identified debts	• Using Product Backlog • Conduct follow-up meetings • Project tracking tools	To pay *Defects not found in tests*: • Change test cases by analyzing defects To pay *Lack of tests*: • Elaborate and perform tests for releases that were not tested To pay *Test effort estimation errors*: • Analyze and correct estimation errors	• Present already identified debts • Review elaborated test cases
REASONS FOR THE CHOICE				
• Identify defects that affect the end-user • Identify lack of testing in important features • Identify test estimation errors	Prioritize TDs related to the most important functionalities, the effort is focused primarily on what is most relevant for the customer/user	Report identified test debts using, backlog, meetings, and GTF test monitoring tools (e.g., JIRA)	The actions for payment of these test debts were already usually used in GTF	Added the review of tests and test estimates carried out in order to avoid the commitment of test debts
CHANGES IN ACTIVITIES				
The following activities were added to the testing process: (i) Analyze e-mails or comments posted at the android app store; (ii) Compare different versions of the product; (iii) Analyze product release plan; and (iv) Analyze reported test effort. All activities were added in the planning phase	With the list of features of the system prioritized by importance, the test debts related to the features that had the greatest impact for the end-user were prioritized	In addition to the use of the backlog and the follow-up meetings, the test debts were documented in JIRA and Confluence for communication with the other members of the team.	• Create new test cases or improve existing cases in order to cover the defect in question or similar • Elaborate and perform tests for releases that were not tested • Change effort reported according to the analysis performed	In addition to presenting the identified test debts to the team at the end of the test demand, the team began to review the test estimates made, as well as the test cases and scenarios created

Fonte: The Author

6.3.3 Third application

The third application of the catalog has occurred again in the product P1. In this application, three subtypes of test debt were chosen to be managed: (i) *Test estimation errors*; (ii) *Inadequate allocation*; (iii) *Inadequate equipment*. For this application, two new subtypes were chosen, inadequate allocation and inadequate equipment. The intention was to identify and manage different test debts that the GTF did not yet know. Also, in this new application, a new member joined the GTF team, so it would be possible to manage potential test debts caused by inadequate allocation.

Table 21 presents details of the actions chosen to manage the subtypes of test debts chosen in this third application. It is important to highlight that all management activities were performed, but this table only details the activities whose actions were not performed in the two previous applications.

Tabela 21 – Third application of TestDCat

SELECTED TEST DEBT SUBTYPES			
1. Test effort estimation errors 2. Inadequate equipment 3. Inadequate allocations			
SELECTED ACTIONS			
IDENTIFICATION	PRIORITIZATION	COMMUNICATION	REPAYMENT
To identify *Test effort estimation errors*: • Analyze reported effort To identify *Inadequate equipment*: • Analyzing test results To identify *Inadequate allocations*: • Analyze delays • Analyzing bugs identified by the customer	Cost/benefit analysis of the acquisition and payment of a debt	• Using Product Backlog • Conduct follow-up meetings • Project tracking tools • Use communication tools	To pay *Test effort estimation errors*: • Analyze and correct estimation errors To pay *Inadequate equipment*: • Make changes to the test environment To pay Inadequate allocations: • Change team allocation
REASONS FOR THE CHOICE			
• Identify test estimation errors • Identify the use of inadequate equipment that impacted the results of the tests performed • Identify inadequate allocations that could impact the quality of the tests performed	Understand the importance of payment based on the cost/benefit analysis of the TD	In addition to the actions already carried out in previous applications, the GTF team decided to include one more means of communication using, in this case, chats in Skype	The actions for payment of these test debts were already usually used in GTF
CHANGES IN ACTIVITIES			
The following activities were added to the testing process: (i) Analyze reported test effort; (ii) Analyzing results of non-deterministic tests (for example, very different values collected during the energy consumption test); (iii) Analyze bugs to verify that they were not found at test time due to some lack of knowledge of the tester allocated to perform the product tests; (iv) Analyze delays in test releases to identify inadequate allocations (e.g., the tester did not have the necessary knowledge to perform those specific tests)	Based on the list of identified debts, the cost/benefit ratio defined in the measurement activity was analyzed. Debts with the highest repayment rates were prioritized	Inclusion of the reporting of test debts identified using Skype. This notification was made whenever a new test debt was identified and documented	• Change effort reported according to the analysis performed • changes in the environment, such as: alteration of a specific equipment (e.g., cell phone model, battery test equipment), alteration of the software version used to perform tests (e.g., monkey test, performance test) • Make changes in team allocation according to identified needs

Fonte: The Author

6.3.4 Fourth application

The fourth application of the catalog was carried out in software the product P2. For this application, the GTF team decided to manage the following subtypes of test debt: (i) *Deferring tests*; (ii) *Lack of tests*; (iii) *Test estimation errors*; and (iv) *Inadequate allocation*. Of these subtypes, only the *Deferring tests* had not been managed in previous applications of the catalog. The subtypes *Lack of testing* and *Test estimation errors* were chosen because they were the subtypes of test debt that had already been identified in this product in previous releases (without the use of the TestDCat). The subtype Inadequate allocation was chosen because, as occurred in the third application of the product P3, a new GTF member was working in the test of this product.

Table 22 shows the actions chosen to manage the previously selected test debt subtypes. As occurred in the explanation of previous applications, the management activities that the actions were not changed in relation to the previous applications were omitted from the table.

Tabela 22 – Fourth application of TestDCat

SELECTED TEST DEBT SUBTYPES		
1. Deferring tests 2. Lack of tests 3. Test estimation errors 4. Inadequate allocation		
SELECTED ACTIONS		
IDENTIFICATION	REPAYMENT	PREVENTION
To identify *Deferring tests*: • Analyze test scope To identify *Lack of tests*: • Compare different versions of the same software To identify *Test effort estimation errors*: • Analyze reported effort To identify *Inadequate allocations*: • Analyze delays	To pay *Deferring tests*: • Prepare and execute tests previously postponed To pay *Lack of tests*: • Elaborate and perform tests for features that were not tested To pay *Test effort estimation errors*: • Analyze and correct estimation errors To pay Inadequate allocations: • Change team allocation	• Present already identified debts • Review elaborated test cases • Consider adequate time to perform the tests
REASONS FOR THE CHOICE		
• Identify the cases in which the test has been postponed • Identify lack of tests in some releases • Identify test estimation errors • Identify inadequate allocations that could impact the quality of the tests performed	The actions for payment of these test debts were already usually used in GTF	In addition to the presentation of test debts and the review of test cases and estimates, the GTF team identified the need to negotiate more flexible deadlines with the client requesting the tests
CHANGES IN ACTIVITIES		
The following activities were added to the testing process: (i) Analyze the scope of the test to identify postponed test cases; (ii) Compare different versions of the product to identify the existence of new features and verify if these features have evidence that they were tested; (iii) Analyze reported test effort; (iv) Analyze delays in test releases to identify inadequate allocations (e.g., the tester did not have the necessary knowledge to perform those specific tests)	• Based on the prioritization of postponed test debts, the team should define which tests should be developed and executed during the test demand • Elaborate and perform tests for features that were not tested • Change effort reported according to the analysis performed • Make changes in team allocation according to identified needs	• Test Debt presentation meeting • Review of test cases and scenarios as well as test estimates • Negotiation with the client for more flexible deadlines that take into account adequate test schedules.

Fonte: The Author

6.3.5 Fifth application

The fifth and last application of the catalog in the scope of this case study was performed on product the P2. For this application, the following test debt subtypes were chosen: (i) *Lack of tests*; and (ii) *Test estimation errors*. The choice of these subtypes was due to the frequent occurrence of these test debts. Only these two subtypes were chosen because this release was short, and the team was worried about having an impact on the demanding deadline if more subtypes of test debt were treated.

For this application, no new actions were performed other than those already carried out in the previous applications. All management activities were performed, and the actions for the subtypes were the same as those carried out in the previous application performed on the product P2.

6.4 Results

During the application of the catalog, some metrics were collected, and, after the completion of the applications, it was applied a survey with the participants of the case study in order to collect feedback on, for example, the catalog usability and effectiveness.

6.4.1 Collected metrics

The metrics collected were part of the GTF process and are based on articles available in white literature (SEELA; YACKEL, 2017) and academic (LAZIC; MASTORAKIS, 2008). In addition to the debts already collected in the GTF, the metric *Number of test debts* was also collected to measure the amount of test debts identified during the test demands.

Table 23 presents the results of the metrics collected after the application of the catalog in each of the test demands for projects P1 and P2.

With regard to test debts, Figure 21 presents the number of test debts identified, repaid, and not repaid during the releases of P1 and P2 products.

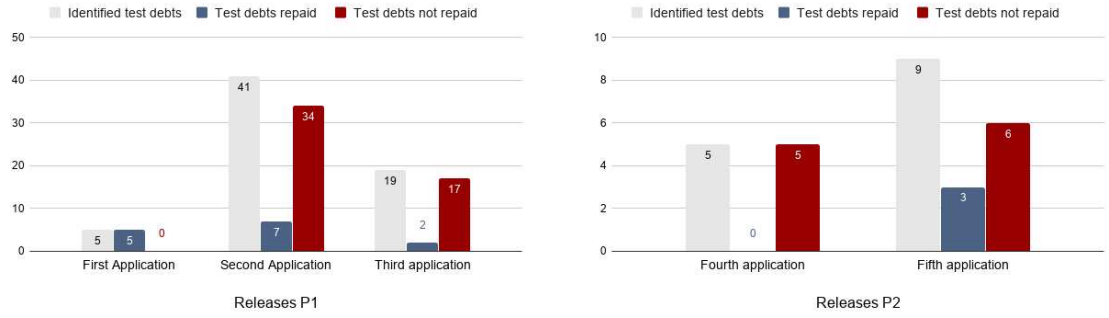
Regarding the time of analysis of the catalog, the first application was the one that required the most time for analysis of the catalog, taking in total two hours of analysis. The other applications took an average of 30 minutes.

Tabela 23 – Metrics collected after the case study

GROUP	METRIC	FIRST APPLICATION	SECOND APPLICATION	THIRD APPLICATION	FOURTH APPLICATION	FIFTH APPLICATION
Absolute Numbers	Total number of test cases	280	309	900	103	131
	Number of test cases passed	252	305	860	61	40
	Number of test cases failed	28	4	40	40	42
	Number of test cases blocked	0	0	0	2	21
	Number of defects found (bugs)	5	2	28	27	18
	Number of planned test hours	28h	51h	87h	20h	59h
	Number of actual test hours	40h	50h	103h	29h	63h
	Number of bugs found after shipping	0	0	0	0	0
Test Effort	Number of tests run per time period	7.2	6	8.4	3.6	1.8
	Test design efficiency	0	0	4.8	4.2	2.4
Test Effectiveness	Test Efficiency	100	100	100	100	100
Test Economics Metrics	Schedule Variance	42.8	-1.9	18.3	45	6.7
Test Debt Metrics	Number of test debts (divided by subtype)	5 Defects Not Found In Tests	41 Lack of Tests	16 Defects Not Found In Tests; 1 Test Effort Estimation Errors; 2 Lack of Tests	5 Defects Not Found In Tests	5 Defects Not Found In Tests; 4 Lack of Tests

Fonte: The Author

Figura 21 – Test debt status in the releases of products P1 and P2



Fonte: The Author

6.4.2 Survey Results

A survey[1] was applied with the participants of the case study to collect subjective aspects using the catalog in the GTF. This survey was divided into three parts: (i) collection of the participants' profiles; (ii) application of the SUS questionnaire to evaluate the usability of the catalog; and (iii) subjective questions about the impact of the use of the catalog on the GTF test demands. A pilot test was conducted to verify if the questionnaire was adequate to be applied with the other GTF members. After this test, the survey was conducted with the GTF members who participated in this case study.

[1] Survey applied after conducting the case study - https://forms.gle/4Gn1jqpV9cHmgvcj8

6.4.2.1 Participants profile

As stated in Section 6.2, five GTF members participated in the case study. Thus, five responses to the applied survey were obtained. As everyone works with software testing, the profile of the participants was defined as *Testing practitioners*. Table 24 presents the profile of the participants who responded to the survey.

Tabela 24 – Survey participants profile

ID	SPECIALITY	EXPERIENCE IN THE FIELD	KNOWLEDGE ABOUT TEST DEBT BEFORE USING THE CATALOG	EXPERIENCE USING CATALOGS
P01	Test analyst	More than three years of experience	-	None
P02				
P03	Test Leader			
P04	Test Intern	One year of experience		
P05		Two years of experience	Basic	

Fonte: The Author

6.4.2.2 SUS Application

The SUS questionnaire (BROOKE *et al.*, 1996) was used to evaluate the usability of the TestDCat. This questionnaire aims to measure the usability of a system according to the perspectives of users. It is based on ISO 9241-11 (STANDARD, 1998) and is commonly used to evaluate the usability of systems, products, and services.

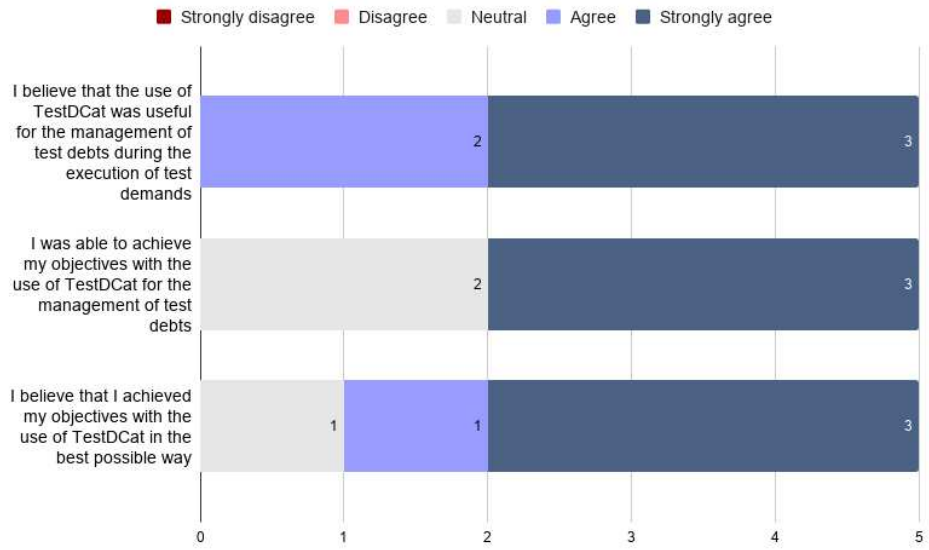
The questionnaire was applied with the participants of the case study, and five answers were obtained. After calculating the results of the questionnaire, a score of 82.5 was obtained.

6.4.2.3 General questions about the catalog

In order to identify from the perspective of GTF members about the usefulness of the TestDCat, some questions were inserted into the Survey: (i) I believe that the use of TestDCat was useful for the management of test debts during the execution of test demands; (ii) I was able to achieve my objectives with the use of TestDCat for the management of test debts; and (iii) I believe that I achieved my objectives with the use of TestDCat in the best possible way. These issues used the 5-point Likert rating scale.

Figure 22 presents the answers collected from the third part of the applied survey.

Figura 22 – Survey answers on the catalog utility



Fonte: The Author

6.5 Discussion

This session will discuss the results achieved with the collection of the metrics and the application of the survey. After that, the research questions will be discussed and the threats to the validity of the case study conducted will be presented.

6.5.1 Collected metrics

Regarding the data collected, it was possible to observe that in all applications, test debts were identified.

In the first application, five test debts of the subtype *Defects not found in tests* were identified, and all of them were paid still in execution time. It was observed that the metric variance of the calendar showed that the effort made was higher than the estimated effort, which may indicate that the use of the catalog may have impacted the effort undertaken in the release.

In the second application, the GTF team chose to manage the test debt subtypes *Defects not found in tests*, *Lack of tests*, and *Test estimation errors*. However, only the subtype *Lack of tests* was identified. Forty-one test debts related to *Lack of tests* were identified, and only seven were repaid during the release. Of the remaining debts, ten were not paid because they

were related to a system functionality that was not prioritized; the remaining were documented to be paid in later releases. In this release, it was possible to observe that the use of the catalog did not impact the schedule variance, even with a large number of test debts identified.

In the third application, the following test debt subtypes were chosen to be managed: (i) *Test estimation errors*; (ii) *Inadequate allocation*; (iii) *Inadequate equipment*. From the subtypes chosen, only one debt related to *Test estimation errors* was identified and no test debts related to *Inadequate allocation* and *Inadequate equipment* were identified. However, other subtypes that were not the focus of this release were identified, such as Sixteen *Defects not found in tests* and Two *Lack of tests*. Due to a lack of prioritization, only two of the identified debts were paid during the release. Nevertheless, the remaining debts were already paid during the execution of the subsequent release. The fact that test debts were identified that were not the focus of the release may indicate that the team has already internalized the management of debts that had already been the focus of previous releases.

Regarding the effort made in the release, despite a large number of test cases performed, no significant difference was observed between the actual effort and that planned, which may indicate that the catalog did not have a significant impact on the increase of hours performed by the team.

The fourth application occurred in product P2 and the following subtypes of test debt were chosen: (i) *Deferring tests*; (ii) *Lack of tests*; (iii) *Test estimation errors*; and (iv) *Inadequate allocation*. However, no test debts of the subtypes that were the focus of this release were identified, but five test debts related to *Defects not found in tests* were identified. These debts were not paid at release time due to low prioritization but were documented for payment in later releases, if appropriate.

This release again showed an increase in the schedule variance metric. However, it is believed that this was due to a large number of bugs identified in the execution of the tests, which required more time for documentation and reporting of these bugs.

In the fifth application, the following subtypes were chosen: (i) *Lack of tests*; and (ii) *Test estimation errors*. However, test debts related to *Defects not found in tests* and *Lack of tests* were identified, five and four, respectively. From the debts identified, six were paid in the release, and six were documented and paid in the subsequent release. The schedule variance metrics showed no significant change.

With the completion of the five applications of the catalog, it can be observed that

the team always identifies test debts in the releases, but not always the debts identified are those that were chosen to be managed in the meeting of the scope of the test demand. This is positive since the team identifies test debts independent of the type that was chosen to be managed in the release.

It can also be observed that the main subtypes that occur in products P1 and P2 were *Defects not found in tests* and *Lack of tests*. Thus, for these products, the GTF team should pay more attention to preventing the acquisition of these debts.

6.5.2 Survey

Regarding the usability of the system, to situate and qualify the usability of a system after application of the SUS questionnaire, Bangor et al. (BANGOR *et al.*, 2008) present a scale listing usability adjectives ranging from the *worst imaginable* to the *best imaginable*, and an *Acceptable* and *Unacceptable* range of scores according to the score obtained after application of the SUS questionnaire.

After applying the SUS questionnaire, the score obtained was 82.5, which indicates, according to Bangor et al. (BANGOR *et al.*, 2008), that the usability of TestDCat is in the *Acceptable* margin of usability. Regarding the adjective presented in the scale, the TestDCat is in the margin between *Good* and *Excellent*.

Regarding the perspective of GTF members about the usefulness of the TestDCat, it was observed that all participants believe that the catalog was useful for the management of the test debts identified in the case study. Most also agree that, with the use of the catalog, they felt able to manage the test debts they chose and that the actions used for this management were adequate and sufficient.

6.5.3 Research questions

After analyzing the results collected during the case study, it is possible to answer the research questions defined in the case study planning.

6.5.3.1 ***RQ1.*** *The use of the catalog assists in the management of test debts?*

In order to answer this question, the survey results were analyzed concerning the user's perception of the catalog usefulness. Also, some metrics collected during the case study

were analyzed.

Regarding the feedback of the participants, all answers were positive about the usefulness of the catalog. Also, by analyzing some defined metrics, it was observed that using the catalog helped the GTF team manage their test debts. There were a total of 79 test debts identified in the five catalog applications, and only five were not paid within the catalog application period. Therefore, it is believed that the catalog assisted in the management of test debts.

*6.5.3.2 **RQ2.** How easy is TestDCat to use?*

To answer this question, the SUS questionnaire was applied to assess the usability level of the TestDCat. Also, an open question was inserted in the survey to receive feedback from participants about the usability of the system.

When analyzing the score obtained by SUS, it was observed that the usability of TestDCat could be considered between good and excellent. However, when analyzing the answers to the open question of the survey, some participants commented that the first use was confusing. However, with the continuous use of the catalog, it was possible to use it without significant problems.

Therefore, it is believed that the catalog has acceptable usability, but it would be interesting to improve it in order to shorten the learning curve required for its use.

6.5.4 Threats to validity

Regarding the threats to validity, we discuss threats related to *Internal Validity* and *External Validity*. According to Wohlin et al. (WOHLIN *et al.*, 2012) and Runeson et al.(RUNESON; HÖST, 2009), threats to *Internal Validity* are influences that can affect the independent variable concerning causality, and threats to *External Validity*, in turn, are conditions that limit the ability to generalize the results to industrial practice.

In our case, the main threat regarding the *Internal Validity* is the selection of the subjects that were made based on convenience sampling (WOHLIN *et al.*, 2012). In the case of *External Validity*, a small number of participants is our main threat. Despite these limitations, it is worth noting that the participants were professionals with significant experience in software testing activities in the industry. Furthermore, these professionals had experience in testing mobile and web software.

6.6 Conclusion

This chapter detailed the case study conducted in the GREat Test Factory. This study aimed to evaluate the application of TestDCat in real products. Five applications were performed in two software products, and the collected results were analyzed.

The case study planning was described with the presentation of the objective of the case study and the research questions derived from this objective. In addition, the users' profiles and the software products that were tested by GTF during the case study period were presented.

The steps for performing the case study were described, and the five TestDCat applications were detailed, presenting the test debt subtypes chosen to be managed and the actions chosen for this management.

The results obtained from the case study were presented. The collected metrics and the survey results with the participants were also described.

Finally, the results were discussed, research questions were answered, and threats to the validity of the case study were presented.

7 CONCLUSION

This master's thesis presented TestDCat, the catalog of Test Debt Subtypes and Management Activities. This catalog was produced from three primary sources: (i) Results of an empirical study in a Test Factory; (ii) Review of the literature on test debt management; and (iii) Results of semi-structured interviews with industry professionals. After intermediate evaluations, a final version of the catalog was generated. With the last version, the results obtained from a case study carried out at the GREat Test Factory were also presented.

Then, this chapter concludes this master's thesis and it is organized as follows. Section 7.1 describes an overview of the work carried out in this master's thesis. Section 7.2 summarizes the main results achieved. Section 7.3 introduces the limitations of this work, and Section 7.4 presents some perspectives for future work.

7.1 Overview

Test Debts have a high impact on software quality. So, they require the use of management activities to keep them visible and under control. However, most studies in the literature dealing with this type of technical debt concentrate in specific subtypes of test debt (e.g., automated tests, exploratory tests). Few studies present an overview of the subtypes and ways of managing them. In addition, there are not many articles focused on non-code related test debts. Finally, despite the growing number of approaches that deal with test debts, there are few case studies in the industry that use these approaches, which makes it difficult to understand the real impact and cost of using these management approaches.

Aiming to address these gaps and support the practitioners in the management of *Test Debts*, the catalog, called TestDCat, was created. TestDCat is based on the results of an empirical study, on a review of the literature, and on the information gathered from semi-structured interviews conducted with professionals from industry.

TestDCat presents an overview of management activities, subtypes of *Test Debts*, and actions to assist in management activities. As an initial assessment, the catalog was presented to the participants of the first series of semi-structured interviews (Chapter 5, Section 5.2.2). They answered a survey regarding clarity, ease of use, and completeness. In the second evaluation, a focus group was organized to analyze in details the actions of the catalog and to suggest changes and improvements. The results of these two evaluations were used to improve the evaluated

version of the catalog and generate its latest version.

From the latest version of TestDCat, a case study was conducted in the GREat Test Factory. It aimed to evaluate the application of TestDCat in real products. Five applications of the catalog were performed in two software products (Chapter 6 and Section 6.1.2), and the collected results were analyzed. Two research questions were defined for the case study: (i) RQ1. *The use of the catalog assists in the management of test debts?*; and (ii) RQ2. *How easy is TestDCat to use?*. The results obtained from the case study presented evidence that the information organized in the catalog can support the management of *Test Debts* and has good usability. Thus, it may help the development and testing team to monitor the current debts and to take the necessary actions.

7.2 Main Results

The main results of this master's thesis are summarized as follows:

- **TestDCat**. A total of 63 actions were identified for the management of the test debt subtypes in this catalog. These actions can be used by professionals who work with software testing to assist them in the management of test debts.
- **Lessons Learned**. Ten lessons learned from the empirical study conducted to identify problems in the GREat test factory in 2017 to mid-2019 are presented. These lessons are divided into identification and prevention lessons. It is worth noting that some of these lessons were used in TestDCat.
- **TestDCat Website**. In order to facilitate the use of the catalog and make it more interactive, a website was created to present TestDCat. In this website, the user can choose which actions he/she wants to use to manage the test debts and generate an action plan from the selected actions. Also, in this website, the user can propose new test debt management actions that can be added to TestDCat.

Furthermore, two papers were published in conferences from the research performed in this work. Table 25 presents the references of these papers. The first article presents (ARAGÃO *et al.*, 2019) the second version of the catalog and the steps taken to develop this version. The second article (ARAGAO *et al.*, 2019) presents the empirical study carried out in the GTF. The results that came from this article were one of the inputs for building the TestDCat.

Three articles related to the theme of this master's thesis were also published. Table 26 presents them.

Tabela 25 – Papers from this master work

Reference	Qualis	Status
ARAGAO, B. ; ANDRADE, R. M. C. ; SANTOS, I. S. ; CASTRO, R. N. S. ; DANTAS, V. L. L. ; TEIXEIRA, T. R.. **TestDCat: Catalog of Test Debt Subtypes and Management Activities**. In: 31ST IFIP International Conference on Testing Software and Systems (IFIP-ICTSS), 2019, Paris. Testing Software and Systems. ICTSS 2019. Lecture Notes in Computer Science, 2019. v. 11812.	B2	Published
ARAGAO, B. ; CASTRO, R. N. S. ; SANTOS, I. S. ; DANTAS, V. L. L. ; ANDRADE, R. M. C. . **Test debts identification in a test factory**. In: Simpósio Brasileiro de Qualidade de Software, 2019, Fortaleza, CE. Anais do XVIII Simpósio Brasileiro de Qualidade de Software, 2019.	B3	Published

Fonte: The Author

Tabela 26 – Published papers related to the theme of this master's thesis

Reference	Qualis	Status
SANTOS, E. B. ; COSTA, L. S. ; **ARAGAO, B.** ; SANTOS, I. S. ; ANDRADE, R. M. C.. **Extraction of test cases procedures from textual use cases to reduce test effort: Test Factory Experience Report**. In: Simpósio Brasileiro de Qualidade de Software, 2019. Anais do XVIII Simpósio Brasileiro de Qualidade de Software, 2019.	B2	Published
VIEIRA, L. ; LIMA, C. ; SANTOS, E. ; **ARAGÃO, B. S.** ; SANTOS, I. S. ; ANDRADE, R. M. C.. **Automação de Testes em uma Fábrica de Testes: Um Relato de Experiência**. In: Simpósio Brasileiro de Sistemas de Informação (SBSI), 2018, Caxias do Sul/RS. Anais do XIV Simpósio Brasileiro de Sistemas de Informação, 2018.	B2	Published
DE SOUSA, AMANDA OLIVEIRA ; DE SOUSA SANTOS, ISMAYLE ; **ARAGÃO, BRUNO SABÓIA** ; DE CASTRO ANDRADE, ROSSANA M. . **Towards an automatic approach to estimating test effort**. In: the 17th Brazilian Symposium, 2018, Curitiba. Proceedings of the 17th Brazilian Symposium on Software Quality - SBQS, 2018. p. 305.	B3	Published

Fonte: The Author

During the period of this master's degree, some studies were also published, not directly related to this master's thesis, but important for the author of this master's thesis to acquire more knowledge about scientific research. Table 27 presents these articles.

7.3 Limitations

The catalog proposed in this master's thesis aims to assist professionals who perform software testing activities to manage test debts. However, TestDCat has some limitations, that are presented as follows.

One of the limitations is the generality of the actions in the catalog. This occurred due to the forms of TestDCat to be used in different kind of organizations, so it was an option to produce more generic actions. With this, users of the catalog could make the necessary changes in the actions to meet the specifications of their organization. However, some users of the catalog may miss more specific actions, which becomes a limitation of TestDCat.

Tabela 27 – Other published works

Reference	Qualis	Status
BORGES, B. ; SILVA, R. A. S. ; PAIVA, J. O. V. ; **ARAGAO, B.** ; SANTOS, I. S. ; ANDRADE, R. M. C. . **Design e avaliação de um aplicativo móvel complementar para um jogo de cartas educacional**. In: Workshop sobre Interação e Pesquisa de Usuários no Desenvolvimento de Jogos (WIPlay), 2019, Vitória,ES. Anais do I Workshop sobre Interação e Pesquisa de Usuários no Desenvolvimento de Jogos (WIPlay), 2019.	B2	Published
BEPPE, THIAGO A. ; DE ARAÚJO, ÍTALO LINHARES ; **ARAGÃO, BRUNO SABÓIA** ; DE SOUSA SANTOS, ISMAYLE ; XIMENES, DAVI ; ANDRADE, ROSSANA M. CASTRO . **GreaTest: Card Game to Motivate the Software Testing Learning**. In: the XXXII Brazilian Symposium, 2018, Sao Carlos. Proceedings of the XXXII Brazilian Symposium on Software Engineering - SBES '18, 2018. p. 298.	B2	Published
ARAGÃO, B. S.; SANTOS, I. S. ; NOGUEIRA, T. P. ; MESQUITA, L. B. M. ; ANDRADE, R. M. C.. **Modelagem Interativa de um Processo de Desenvolvimento com Base na Percepção da Equipe: Um Relato de Experiência**. In: XIII Brazilian Symposium on Information Systems, 2017, Lavras, Minas Gerais. Proceedings [of the] XIII Brazilian Symposium on Information Systems SBSI 2017, 2017. p. 428-435.	B2	Published
ANDRADE, R. M. C. ; SANTOS, I. S. ; ARAUJO, I. L. ; **ARAGÃO, B.** S. ; SIEWERDT, F.. **Retrospective for the Last 10 years of Teaching Software Engineering in UFC's Computer Department**. In: Brazilian Simposium of Software Engineering, 2017, Fortaleza, Ceará. Proceedings of XXXI Brazilian Simposium of Software Engineering, 2017.	B2	Published

Fonte: The Author

Another limitation is with regard to the possible bias of the catalog. This is due to the fact that one of the TestDCat inputs was generated from semi-structured interviews with industry professionals, but only professionals from the same organization were interviewed. This fact may be a limitation of the work since it has become biased. However, as previously stated, it was decided to include in the catalog more generic actions for the management of test debt in order to reduce the bias of the inserted actions. Besides, the catalog also contains actions identified in the literature that have a different context from the context of the organization of the professionals interviewed.

Finally, in the case study, no comparison was made between the execution of tests with and without the use of the TestDCat. This occurred because the demands of the same product are different from each other (for example, due to change in scope or change in the team that will test the demand), and the comparison could not be performed with the metrics already collected in the GTF. Therefore, this is a limitation of the case study carried out, but it is intended to correct it in the continuation of the work presented in this master's thesis.

7.4 Future Work

From the results of this master's thesis, the main future research directions are described as follows:

- In order to make the catalog more general and to identify new test debts, it is necessary to expand the literature review conducted with a systematic review on test debts and their management.
- Analyze how test debt subtypes relate to each other and how test debts can impact other types of technical debt.
- In the current version, the use of the catalog is entirely manual, research can be done to study ways to automate some actions proposed in the catalog in order to facilitate its use.
- Most of the work that uses tools to help manage test debt are code-related. As a result, there is a lack of tools that use artifacts and that are not code-related. Therefore, it is important to study how such tools could be implemented.
- Although the present study utilized the System Usability Scale (SUS) questionnaire (BROOKE *et al.*, 1996) to evaluate the usability of the technical debt catalog for tests, future evaluations might consider applying the Technology Acceptance Model (TAM) questionnaire (DAVIS, 1989). TAM is a widely recognized model that examines user acceptance of technology, focusing on two main factors: Perceived Usefulness and Perceived Ease of Use. Applying TAM could provide deeper insights into how users perceive the effectiveness and ease of use of the catalog, as well as offer a more comprehensive understanding of the catalog acceptance and potential integration in different organizational contexts.
- The 5W1H method was used to present the test debt catalog. For future work, the 5W2H method could be also used, since it adds "How much"to the existing questions (What, Why, Where, When, Who and How). This addition could provide a more comprehensive analysis by incorporating cost or effort considerations associated with managing test debt items. This approach increases understanding of the practical implications and feasibility of different strategies for managing test debt, resulting eventually in improved practices.
- Different types of applications (e.g., web applications, mobile applications, desktop software, and embedded systems) may present unique challenges and requirements that influence how technical debt is managed and prioritized. By investigating these contextual factors, it is possible to determine whether specific adaptations or extensions to the catalog are necessary to better serve different application domains. Such an analysis would provide

valuable insights into the applicability and adaptability of the test debt catalog, ultimately enhancing its utility and effectiveness across diverse software development environments.

– Regarding the case study, it is important to apply the TestDCat in other organization that do testing activities. Also, metrics should be studied to perform the comparison between demands that use and do not use TestDCat.

REFERÊNCIAS

ALVES, N. S.; MENDES, T. S.; MENDONÇA, M. G. de; SPÍNOLA, R. O.; SHULL, F.; SEAMAN, C. Identification and management of technical debt: A systematic mapping study. **Information and Software Technology**, Elsevier, v. 70, p. 100–121, 2016.

ANDRADE, R. M. C.; LELLI, V.; CASTRO, R. N. S.; SANTOS, I. S. Fifteen years of industry and academia partnership: Lessons learned from a brazilian research group. In: **2017 IEEE/ACM 4th International Workshop on Software Engineering Research and Industrial Practice (SER IP)**. [*S. l.*: *s. n.*], 2017. p. 10–16.

ANDRADE, R. M. de C.; SANTOS, I. de S.; LELLI, V.; OLIVEIRA, K. M. de; ROCHA, A. R. C. da. Software testing process in a test factory - from ad hoc activities to an organizational standard. In: **ICEIS**. [*S. l.*: *s. n.*], 2017.

ARAGAO, B.; CASTRO, R. N. S.; SANTOS, I. S.; DANTAS, V. L. L.; ANDRADE, R. M. C. Test debts identification in a test factory. In: **Proceedings of the 18th Brazilian Symposium on Software Quality**. New York, NY, USA: ACM, 2019. (SBQS).

ARAGÃO, B. S.; ANDRADE, R. M. C.; SANTOS, I. S.; CASTRO, R. N. S.; LELLI, V.; DARIN, T. G. R. Testdcat: Catalog of test debt subtypes and management activities. In: GASTON, C.; KOSMATOV, N.; GALL, P. L. (Ed.). **Testing Software and Systems**. Cham: Springer International Publishing, 2019. p. 279–295. ISBN 978-3-030-31280-0.

BANGOR; AARON; KORTUM, P.; T., P.; MILLER; T., J. The system usability scale (sus): an empirical evaluation. **International Journal of Human-Computer Interaction**, v. 24, p. 574–, 08 2008.

BROOKE, J. *et al.* Sus-a quick and dirty usability scale. **Usability evaluation in industry**, London–, v. 189, n. 194, p. 4–7, 1996.

BROWN, N.; CAI, Y.; GUO, Y.; KAZMAN, R.; KIM, M.; KRUCHTEN, P.; LIM, E.; MACCORMACK, A.; NORD, R.; OZKAYA, I. *et al.* Managing technical debt in software-reliant systems. In: ACM. **Proceedings of the FSE/SDP workshop on Future of software engineering research**. [*S. l.*], 2010. p. 47–52.

BURNARD, P. A method of analysing interview transcripts in qualitative research. **Nurse education today**, Elsevier, v. 11, n. 6, p. 461–466, 1991.

Chernak, Y. Validating and improving test-case effectiveness. **IEEE Software**, v. 18, n. 1, p. 81–86, Jan 2001.

CHUNG, L.; NIXON, B. A.; YU, E.; MYLOPOULOS, J. **Non-functional requirements in software engineering**. [*S. l.*]: Springer Science & Business Media, 2012. v. 5.

COYNE, I. Sampling in qualitative research. purposeful and theoretical sampling; merging or clear boundaries? **Journal of advanced nursing**, v. 26, n. 3, p. 623–630, 1997.

CUNNINGHAM, W. The wycash portfolio management system. **SIGPLAN OOPS Mess.**, ACM, New York, NY, USA, v. 4, n. 2, p. 29–30, dez. 1992. ISSN 1055-6400. Disponível em: http://doi.acm.org/10.1145/157710.157715.

DAVIS, F. D. Perceived usefulness, perceived ease of use, and user acceptance of information technology. **MIS Quarterly**, Management Information Systems Research Center, University of Minnesota, v. 13, n. 3, p. 319–340, 1989. ISSN 02767783, 21629730. Disponível em: http://www.jstor.org/stable/249008.

FERNANDES, F.; SOUSA, S.; LOPES, I. d. S. On the use of quality tools: a case study. In: NEWSWOOD LIMITED PUBLISHER. **World Congress on Engineering 2013**. [*S. l.*], 2013. v. 1, p. 634–639.

GORSCHEK, T.; GARRE, P.; LARSSON, S.; WOHLIN, C. A model for technology transfer in practice. **IEEE software**, IEEE, v. 23, n. 6, p. 88–95, 2006.

GUO, Y.; SEAMAN, C.; SILVA, F. Q. da. Costs and obstacles encountered in technical debt management–a case study. **Journal of Systems and Software**, Elsevier, v. 120, p. 156–169, 2016.

HASS, A. **Guide to Advanced Software Testing**. Artech House, 2014. ISBN 9781608078042. Disponível em: https://books.google.com.br/books?id=mhr7oAEACAAJ.

HOVE, S. E.; ANDA, B. Experiences from conducting semi-structured interviews in empirical software engineering research. In: IEEE. **Software metrics, 2005. 11th ieee international symposium**. [*S. l.*], 2005. p. 10–pp.

ISO/IEC29119-1. Iso/iec/ieee international standard - software and systems engineering –software testing –part 1:concepts and definitions. **ISO/IEC/IEEE 29119-1:2013(E)**, p. 1–64, Sept 2013.

ISO/IEC29119-2. Iso/iec/ieee international standard - software and systems engineering –software testing –part 2:test processes. **ISO/IEC/IEEE 29119-2:2013(E)**, Sept 2013.

ISTQB. **Standard glossary of terms used in Software Testing**. 2012.

KRUCHTEN, P.; NORD, R. L.; OZKAYA, I.; VISSER, J. Technical debt in software development: From metaphor to theory report on the third international workshop on managing technical debt. **SIGSOFT Softw. Eng. Notes**, ACM, New York, NY, USA, v. 37, n. 5, p. 36–38, set. 2012. ISSN 0163-5948. Disponível em: http://doi-acm-org.ez11.periodicos.capes.gov.br/10.1145/2347696.2347698.

KRUEGER, R. A.; CASEY, M. A. **Designing and conducting focus group interviews**. 2002.

LAZAR, J.; FENG, J. H.; HOCHHEISER, H. **Research methods in human-computer interaction**. [*S. l.*]: Morgan Kaufmann, 2017.

LAZIC, L.; MASTORAKIS, N. Cost effective software test metrics. **WSEAS Transactions on Computers**, World Scientific and Engineering Academy and Society (WSEAS), v. 7, n. 6, p. 599–619, 2008.

LI, Z.; AVGERIOU, P.; LIANG, P. A systematic mapping study on technical debt and its management. **Journal of Systems and Software**, Elsevier, v. 101, p. 193–220, 2015.

LIM, E.; TAKSANDE, N.; SEAMAN, C. A balancing act: What software practitioners have to say about technical debt. v. 29, p. 22–27, 11 2012.

METTE, A.; HASS, J. Testing processes. In: IEEE. **Software Testing Verification and Validation Workshop, 2008. ICSTW'08. IEEE International Conference on**. [*S. l.*], 2008. p. 321–327.

MYERS, G. J.; SANDLER, C.; BADGETT, T. **The Art of Software Testing**. 3rd. ed. [*S. l.*]: Wiley Publishing, 2011. ISBN 1118031962, 9781118031964.

ORSO, A.; ROTHERMEL, G. Software testing: a research travelogue (2000–2014). In: ACM. **Proceedings of the on Future of Software Engineering**. [*S. l.*], 2014. p. 117–132.

OZKAYA, I.; KRUCHTEN, P.; NORD, R. L.; BROWN, N. Managing technical debt in software development: Report on the 2nd international workshop on managing technical debt, held at icse 2011. **SIGSOFT Softw. Eng. Notes**, ACM, New York, NY, USA, v. 36, n. 5, p. 33–35, set. 2011. ISSN 0163-5948. Disponível em: http://doi-acm-org.ez11.periodicos.capes.gov.br/10.1145/2020976.2020979.

PERRY, D. E.; PORTER, A. A.; VOTTA, L. G. Empirical studies of software engineering: a roadmap. In: ACM. **Proceedings of the conference on The future of Software engineering**. [*S. l.*], 2000. p. 345–355.

PETUNOVA, O.; BĒRZIŠA, S. Test case review processes in software testing. **Information Technology and Management Science**, De Gruyter Open, v. 20, n. 1, p. 48–53, 2017.

ROCHA, A.; MONTONI, M.; SANTOS, G.; MAFRA, S.; FIGUEIREDO, S.; ALBUQUERQUE, A.; MIAN, P. Reference model for software process improvement: A brazilian experience. In: . [*S. l.*: *s. n.*], 2005. v. 3792, p. 130–141.

ROGERS, Y.; SHARP, H.; PREECE, J. **Interaction design: beyond human-computer interaction**. [*S. l.*]: John Wiley & Sons, 2011.

RUBIN, K. S. **Essential Scrum: A practical guide to the most popular Agile process**. [*S. l.*]: Addison-Wesley, 2012.

RUNESON, P.; HÖST, M. Guidelines for conducting and reporting case study research in software engineering. **Empirical software engineering**, Springer, v. 14, n. 2, p. 131, 2009.

SABINO, G. **Como conduzir uma sessão de lições aprendidas**. 2016. Disponível em: http://www.radardeprojetos.com.br/2016/02/como-conduzir-uma-sessao-de-licoes.html. Acesso em: 16 oct. 2019.

SAMARTHYAM, G.; MURALIDHARAN, M.; ANNA, R. K. Understanding test debt. In: **Trends in Software Testing**. [*S. l.*]: Springer, 2017. p. 1–17.

SANZ, A.; GARCIA, J.; SALDANA, J.; AMESCUA, A. A proposal of a process model to create a test factory. In: IEEE. **Software Quality, 2009. WOSQ'09. ICSE Workshop on**. [*S. l.*], 2009. p. 65–70.

SEELA, S.; YACKEL, R. **64 Essential Testing Metrics for Measuring Quality Assurance Success**. 2017. Disponível em: https://www.qasymphony.com/blog/64-test-metrics/. Acesso em: 08 july 2019.

SHAH, S. M. A.; TORCHIANO, M.; VETRO, A.; MORISIO, M. Exploratory testing as a source of technical debt. **IT Professional**, IEEE, v. 16, n. 3, p. 44–51, 2014.

SHNEIDERMAN, B.; PLAISANT, C.; COHEN, M.; JACOBS, S.; ELMQVIST, N.; DIAKOPOULOS, N. **Designing the user interface: strategies for effective human-computer interaction**. [*S. l.*]: Pearson, 2016.

SLAUGHTER, S. A.; HARTER, D. E.; KRISHNAN, M. S. Evaluating the cost of software quality. **Communications of the ACM**, ACM, v. 41, n. 8, p. 67–73, 1998.

SMITH, J.; TESSLER, J.; KRAMER, E.; LIN, C. Using peer review to teach software testing. In: **Proceedings of the Ninth Annual International Conference on International Computing Education Research**. New York, NY, USA: ACM, 2012. (ICER '12), p. 93–98. ISBN 978-1-4503-1604-0. Disponível em: http://doi.acm.org/10.1145/2361276.2361295.

SOUSA, A. O. de; SANTOS, I. de S.; aO, B. S. A.; ANDRADE, R. M. de C. Towards an automatic approach to estimating test effort: An experience report. In: **Proceedings of the 17th Brazilian Symposium on Software Quality**. New York, NY, USA: ACM, 2018. (SBQS), p. 305–314. ISBN 978-1-4503-6565-9. Disponível em: http://doi.acm.org/10.1145/3275245.3275273.

SOUSA, C. L. d. **Mapa de apoio à gestão de dívida técnica no processo de teste de software**. Dissertação (Mestrado) – Universidade Federal de Pernambuco, 2016.

SPILLNER, A.; LINZ, T.; SCHAEFER, H. **Software testing foundations: a study guide for the certified tester exam**. [*S. l.*]: Rocky Nook, Inc., 2014.

STANDARD, I. Ergonomic requirements for office work with visual display terminals (vdts)–part 11: Guidance on usability. iso standard 9241-11: 1998. **International Organization for Standardization**, 1998.

STUFFLEBEAM, D. L. Guidelines for developing evaluation checklists: the checklists development checklist (cdc). **Kalamazoo, MI: The Evaluation Center**, 2000.

SUTHERLAND, J.; SCHWABER, K. The scrum guide. **The definitive guide to scrum: The rules of the game. Scrum. org**, v. 268, 2013.

TECHTERMS. **Hotfix Definition**. 2015. Disponível em: https://techterms.com/definition/hotfix. Acesso em: 16 oct. 2019.

TURNER, D. Qualitative interview design: A practical guide for novice investigators. **Qualitative Report**, v. 15, 05 2010.

TYNAN, C.; DRAYTON, J. Conducting focus groups — a guide for first time users. **Marketing Intelligence Planning**, v. 6, p. 5–9, 12 1988.

VIEIRA, L. S.; BARRETO, C. G. L.; SANTOS, E. B. dos; ARAGãO, B. S.; SANTOS, I. de S.; ANDRADE, R. M. C. Automação de testes em uma fábrica de testes: Um relato de experiência. In: SBC. **Anais do XIV Simpósio Brasileiro de Sistemas de Informação**. [*S. l.*], 2018. p. 80–73.

WIKLUND, K.; ELDH, S.; SUNDMARK, D.; LUNDQVIST, K. Technical debt in test automation. In: **2012 IEEE Fifth International Conference on Software Testing, Verification and Validation**. [*S. l.*: *s. n.*], 2012. p. 887–892. ISSN 2159-4848.

WIKLUND, K.; ELDH, S.; SUNDMARK, D.; LUNDQVIST, K. Impediments for software test automation: A systematic literature review. **Software Testing, Verification and Reliability**, Wiley Online Library, v. 27, n. 8, 2017.

WOHLIN, C. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In: **Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering**. [*S. n.*], 2014. (EASE '14), p. 38:1–38:10. ISBN 978-1-4503-2476-2. Disponível em: http://doi.acm.org/10.1145/2601248.2601268.

WOHLIN, C.; RUNESON, P.; HST, M.; OHLSSON, M. C.; REGNELL, B.; WESSLN, A. **Experimentation in Software Engineering**. [*S. l.*]: Springer Publishing Company, Incorporated, 2012. ISBN 3642290434, 9783642290435.

YLI-HUUMO, J.; MAGLYAS, A.; SMOLANDER, K. How do software development teams manage technical debt?–an empirical study. **Journal of Systems and Software**, Elsevier, v. 120, p. 195–218, 2016.

APÊNDICE A – INTERVIEW SCRIPT

Table 1 presents the script used to conduct the interviews.

Tabela 1 – Interview Script

Interview steps	Topic	Sample questions
1. Interviewee Identification	Academic background	1.1 What is your academic background?
	Profession	1.2 What is your profession?
	Activities carried out in the project	1.3 What is your role in the team?
		1.4 What test activities do you perform?
	Knowledge of Test Debts	1.5 What is your understanding of the concept of technical debt?
2. Questions about test technical debt	Indication of the existence of test TDs	2.1 In cases of pressure for delivery, is there an impact on testing activities?
		2.1.1 How does this affect your activities?
		2.1.2 Are there cases of low code coverage?
		2.1.3 Are there cases of deferring testing?
		2.1.4 Are there cases of lack of test?
		2.1.5 Are there cases of lack of test automation?
		2.1.6 Are there cases of defects not found in tests?
		2.1.7 Are there expensive test cases?
		2.1.8 Do cases of estimation errors occur?
3. Questions about TD management activities	Identification	3.1.1 How is the existence of <Test Debt subtype> identified?
		3.1.2 Would you have any suggestions for improving the process or activities that could assist in this identification?
	Measurement	3.2.1 Are there any measurement to quantify cost/benefit of these identified <Test Debt subtype>?
		3.2.2 Would you have any suggestions for improving the process or activities that could assist in this measurement?
	Prioritization	3.3.1 Are there any rankings to facilitate the choice of which <Test Debt subtype> should be done first?
		3.3.2 Would you have any suggestions for improving the process or activities that could assist in this priorization?
	Communication	3.4.1 Is the entire team aware of the identified <Test Debt subtype>?
		3.4.2 Would you have any suggestions for improving the process or testing activities that could assist in this communication?
	Monitoring	3.5.1 Is there any monitoring to identify changes in the cost/benefit of the identified <Test Debt subtype>?
		3.5.2 Do you have any suggestions for improving the process or activities that could assist in this monitoring?
	Repayment	3.6.1 Is the identified <Test Debt subtype> repayment sometimes in the project life cycle?
		3.6.2 Do you have any suggestions for improving the process or activities that could help in this activity?
	Documentation	3.7.1 How the existence of these <Test Debt subtype> are documented?
		3.7.2 Do you have any suggestions for improving the process or activities that could assist in this documentation?
	Prevention	3.8.1 Is anything done to prevent this <Test Debt subtype> from happening again in other demands?
		3.8.2 Do you have any suggestions for improving the process or activities that could help in this activity?
4. Slowdown	Final considerations	If the aforementioned activities (e.g., identification, communication, repayment) are not performed, does this impact the quality of the tests that are done?
		If there is a catalog with information that could help you to perform these activities, would you use it?

Fonte: The Author