TestDCat 3.0: catalog of test debt subtypes and management activities

Bruno S. Aragão1 · Rossana M. C. Andrade1 · Ismayle S. Santos1 · Rute N. S. Castro1 · Valéria Lelli1 · Ticianne G. R. Darin1

© Springer Science+Business Media, LLC, part of Springer Nature 2021

Abstract
When deadlines and resources of software projects become scarce, testing is usually in the first row to have its activities aborted or reduced; however, if defects cannot be found, product quality can be affected. In the 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) and, when the TDs impact testing activities, they are called Test Debt. Although there are several studies dealing with Test Debt, 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 then proposes a Test Debt Catalog, called TestDCat, 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 these software development projects.

Keywords
Technical Debt · Test Debt · TD management activity

Bruno S. Aragão
aragaobs@gmail.com
Rossana M. C. Andrade
rossana@great.ufc.br
Ismayle S. Santos
ismaylesantos@great.ufc.br
Rute N. S. Castro
rute@great.ufc.br
Valéria Lelli
valerialelly@great.ufc.br
Ticianne G. R. Darin
ticiannedarin@great.ufc.br

1 Group of Computer Networks, Software Engineering, and Systems (GREat), Federal University of Ceará (UFC), Ceará, Fortaleza, Brazil
1 Introduction

Technical Debts are technical commitments generated in software projects that can bring short-term benefits, but which, in the long run, can be detrimental to project quality Li et al. (2015). This concept was first used by Cunningham (1992), who related the characterization of Technical Debts to problems in the code and the need for refactoring to pay the debts acquired. Other studies have addressed Technical Debts in other activities of the software development process (e.g., tests, requirements, usability) and provided solutions to manage them in software projects Alves et al. (2016) Li et al. (2015).

When the Technical Debts concern the software testing, they are known as Test Debts. They arise due to inadequate decisions regarding testing activities (e.g., lack of tests, test estimation errors) Samarthyam et al. (2017). For instance, 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).

According to Samartyam 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.

Although there are a significant number of works that deal 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.

Thus, our work intends to fill the following gaps identified in the systematic mappings of Alves et al. (2016) and Li et al. (2015) and the literature review that is also done in this work: (i) lack of studies dealing with other types of Technical Debts (e.g., Test Debts); (ii) need for identifying non-code-related Technical Debts; (iii) few studies with more global approaches that cover more causes of Test Debts; and (iv) need for validated solutions in industry projects.

The knowledge acquired as a result of performing this work was organized in a catalog format and made available for the use of practitioners who work with software testing and Test Debts. The decision for this kind of format came from the fact that catalogs are ways of organizing the information and know-how that originates from practitioners Chung et al. (2012) and can be used as a source of information on how to deal with Technical Debt Kruchten et al. (2012) Ozkaya et al. (2011).

In a previous work Aragão et al. (2019), we proposed the catalog, which we called TestDCat, with subtypes of Test Debts and management activities that can be used to help in the management of Test Debts. To build this catalog, we first investigated, through an empirical study in industry, problems that occurred in a testing factory, which performed tests in applications and systems, and then we identified that these problems could be formulated and treated as Test Debt. Next, the catalog itself (TestDCat 1.0) was developed based on the information gathered from semi-structured interviews and a literature review about Test Debts and their management. After analyzing the results of the first evaluation, we proposed TestDCat 2.0 to improve the first version. This second version was also evaluated and changes and improvements were suggested. The results of the evaluation presented evidence
that the information organized in the catalog could support the management of Test Debts. This last evaluation also brings out the need of new improvements to our catalog.

So, this paper extends our aforementioned previous work by presenting the third version of our catalog, TestDCat 3.0, and a case study to evaluate it. This new version includes improvements that we identified from the results of the previous evaluation of TestDCat 2.0 Aragão et al. (2019). Regarding the case study, it was conducted in software projects of the industry to assess the uses of the catalog to assist the Test Debt management and its usability. Additionally, we present a summary of our empirical study conducted in a test factory to build the TestDCat.

This paper is divided into seven sections. In Section 2, we introduce the methodology and steps used to build the catalog. TestDCat 3.0 is described in Section 3. In Section 4, we present the case study conducted and the related works are discussed in Section 5. Finally, in Section 6, we present the final considerations and perspectives for future work.

2 Methodology

The construction of the catalog follows a methodology (see Fig. 1) that is partially based on the technology transfer model presented by Gorschek et al. (2006). This model favors cooperation between academia and industry and can be beneficial to both. In short, it gives researchers the opportunity to study relevant industry issues and validate their results in a real environment and, in return, professionals receive knowledge about new technologies that can, for example, optimize their processes.

Regarding the first step of the methodology, the step called identify the problem consists of the observation, by the researcher, of the activities performed in the industry in order to identify real problems/issues. In our case, we identified our problem from the experience in developing software in a successful long-term partnership with the industry Andrade et al. (2017) in Research, Development and Innovation (R&D&I) projects that occur in the environment where this work has been developed. In this kind of projects, the GREat Test

---

1 GREat stands for Group of Computer Networks, Software Engineering and Systems and works on research and development software projects, developing web and mobile tools that are continually being tested by the GREat Test Factory team.
factory (GTF) team has followed a testing process as documented in Andrade et al. (2017). Once we identified our problem, we conducted an empirical study to identify the main issues faced by GTF. (The results of this empirical study are also presented in Aragão et al. (2019).) As a result of the first step, the problem was identified and formulated as Test Debt.

The second step is called perform the literature review (see also details in Section 3 and 5). The goal of this step is to better understand the problem, in our case, Test Debts, and identify how to 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. (2015) and Alves et al. (2016). Based on the principle of Snowballing Wohlin (2014), we also performed a search for studies that cited the set of previously selected papers.

The third step is to propose the candidate solution to tackle the identified problem. In our previous work, two candidate versions are presented Aragão et al. (2019). The first version was 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 was then organized in a catalog format. The second version of the candidate solution was obtained from the result of the evaluation performed in TestDCat 1.0 and the information collected in a new series of semi-structured interviews conducted with software developers who also performed testing activities. This work presents a third candidate solution, which was developed from the analysis of the results of the evaluation of the previous versions.

The next steps of the methodology refer to the static and dynamic validation (i.e., perform the static validation and perform the dynamic validation). For the static validation, a survey is applied to evaluate the first version of the catalog, and a focus group Krueger and Casey (2002) is conducted to evaluate the second version. The dynamic validation is executed as case study Runeson and Höst (2009) with industry projects (five applications) at the GREat Test Factory.

In the last step, release the solution, we released the third version of the catalog, TestDCat 3.0, and used it in industry projects.

3 TestDCat 3.0

This section presents the third version of TestDCat, which was built based on the evaluations made in the catalog previous versions Aragão et al. (2019). The steps and details for building TestDCat 3.0 can be seen in Aragão et al. (2019).

In TestDCat 3.0, improvements have been made to the existing catalog versions that are related to the addition or removal of actions as well as occasional improvements in the descriptions of each action (e.g., change of the person responsible for the action). The structure of the catalog remains the same, using the 5W1H model Fernandes et al. (2013), commonly used to conduct action plans. The data is categorized according to TD management activities and subtypes of Test Debts, both mapped by Li et al. (2015). In addition to the subtypes identified by Li et al., two other subtypes have been identified: Inadequate equipment and Inadequate allocation.

Regarding the subtype Inadequate Equipment, 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.

Figure 2 presents the catalog, emphasizing with listed circles parts of it. 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 the example shown in Fig. 3, 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/she wants to handle. In this example, we choose “Low Code Coverage” that 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. In circle number four, we present the functionality in which the user can select which actions are most related to the context. A customized action plan is then created.

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

To use 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/she 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. Both forms of use are available on the website and can be changed by practitioners as needed.

3.1 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.

3.1.1 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 1 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.

3.1.2 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 2 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.

3.1.3 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.

---

2 TestDCat Catalog’s website: https://great-ufc.github.io/TestDCat/index.html
Table 1  Example of identification actions

| TD management activity: Identification Test Debt subtype: Lack of Tests | What? | Why? | Where? | How? | Who? | When? |
|---|---|---|---|---|---|---|
| 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 texting | 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 test in that version. | Development and test template | Periodically or whenever new versions have been released for testing. |

| TD management activity: Identification Test Debt subtype: Defects not found in tests | Analyze feedback from end-users | Users may submit defects found by them via email or via product reviews in any app/systems store and social networks | Email, product reviews and social networks | Analyze emails and user reviews of released products to identify possible defects not found at development/test team | Customer support, developments/test team | Periodic reports or from the analysis of some specific metrics (e.g., bugs found by the end-user) |
**Table 2  Example of measurement action**

| TD management activity | What? | Why? | Where? | How? | Who? | When? |
|------------------------|-------|------|--------|------|------|-------|
| Measurement Test debt sub-type: 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 | 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). |
| What? | Why? | Where? | How? | Who? | When? |
|-------|------|--------|------|------|-------|
| TD management activity: Prioritization Test Debt subtype: 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 debt 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 |
| What? | Why? | Where? | How? | Who? | When? |
|-------|------|--------|------|------|-------|
| 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/use | 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 |
Table 3 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.

3.1.4 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 4 presents two examples of actions. The first action concerns the use of follow-up meetings to report identified debts. The second action is about the use of communication tools (e.g., Skype, email) to report the existence of debts.

3.1.5 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 5 presents some of the examples of the actions 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.

3.1.6 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 6 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.
Table 4  Example of communication actions

| TD management activity: Communication Test Debt subtype: All Test Debt Subtypes |
| --- |
| **What?** | **Why?** | **Where?** | **How?** | **Who?** | **When?** |
| Conduct follow-up meetings | Follow-up meetings happen 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 |
Table 5  Example of monitoring actions

| What? | Why? | Where? | How? | Who? | When? |
|-------|------|--------|------|------|-------|
| 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 |
| What? | Why? | Where? | How? | Who? | When? |
|-------|------|--------|------|------|-------|
| 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 functionality that had identified defects - 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 |
### Table 6  Example of repayment actions

| What? | Why? | Where? | How? | Who? | When? |
|-------|------|--------|------|------|-------|
| 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 |
| What?                                                                 | Why?                                                                 | Where?                                                           | How?                                                                 | Who?                                   | When?                                |
|---------------------------------------------------------------------|----------------------------------------------------------------------|------------------------------------------------------------------|---------------------------------------------------------------------|----------------------------------------|---------------------------------------|
| 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 |                                                      |
Table 7 Example of documentation actions

| TD management activity: Documentation Test Debt subtype: All Test Debt Subtypes | What? | Why? | Where? | How? | Who? | When? |
|---|---|---|---|---|---|---|
| Standardize debt documentation | Using a standard facilitates documentation and understanding of the identified debt | Project tracking tool, WIKI and email | 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 |
3.1.7 Documentation

The documentation management activity is intended to document identified TDs for the knowledge of all stakeholders and for future consultation. Table 7 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.

3.1.8 Prevention

This management activity aims to prevent the occurrence of TDs. Table 8 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.

3.1.9 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 9 presents some examples of these types of actions in different management activities and Test Debt subtypes.

3.2 Limitations

The catalog proposed in this work aims to assist professionals and researchers 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 can be used in different kinds 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; then, the generality becomes a limitation of TestDCat.

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 is a limitation of the work since it can be 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.
| TD management activity: Prevention Test Debt subtype: All Test Debt Subtypes | What? | Why? | Where? | How? | Who? | When? |
|---|---|---|---|---|---|---|
| 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 estimation can be performed. A second person can identify errors in this estimation and propose changes. | Development/test team | Whenever there is a test creation or change |
### Table 9: Example of test process adjustments actions

| What? | Why? | Where? | How? | Who? | When? |
|-------|------|--------|------|------|-------|
| 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 |
| What? | Why? | Where? | How? | Who? | When? |
|-------|------|--------|------|------|-------|
| **TD management activity:** Repayment | 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. |
4 Case study

Following the research methodology used in this work (see Section 2), a dynamic validation was planned and conducted. This validation was performed by a case study at GREat Test Factory (GTF) and aimed to validate the use of the catalog in real projects with the industry.

4.1 Case study design

Case studies are commonly used to study phenomena that occur in their natural context Runeson and Höst (2009). So, this work 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 case study guide proposed by Runeson and Höst (2009), which is a relevant guide for conducting case studies in software engineering.

4.1.1 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 GTF participants on Test Debts, to measure the current status of the occurrence of Test Debts in GTF and to investigate the impact of 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:

- **RQ1. Does the use of the catalog assist 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 GTF as well as the verification of indicators collected after the use of the catalog. Table 10 summarizes these metrics and their respective calculation formulas.

- **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.

4.1.2 Study objects

The objects were two software products developed in partnership with the industry. The selected products were the ones that demanded tests for the GTF during the period of the case study.

Table 11 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 confidentiality reasons. A total of five uses of the catalog were analyzed.
| 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 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                                  | \[
\frac{DT}{(DT + DU)} * 100
\] (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                                | \[
\frac{\text{Actual effort} - \text{estimated effort}}{\text{estimated effort}} \times 100
\] | 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                                                 |
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 Android versions to ensure the correct functioning of the application on these various versions in which it should operate.

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.

### 4.1.3 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, who 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.

In addition to practitioners, two researchers with knowledge of TestDCat and its actions as well as a vast knowledge of Test Debts, participated in the case study. These researchers made observations and notes during the execution of the case study for further discussion.

### 4.2 Execution of the case study

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

The catalog was applied in P1 and P2 software products when testing demands were requested to GTF. The use of the catalog followed the step by step defined by one of the researchers together with the GTF leader:

- The GTF team would meet to analyze the catalog and identify which subtypes of Test Debt would be managed in that demand. In addition, at this meeting, it was also defined which actions would be taken from each management activity of the previously chosen Test Debt subtype.
- Analyze the chosen actions and adapt the activities of the test process established in GTF to manage the chosen Test Debt subtypes.
- Perform the actions together with the activities of the GTF testing process.

| Tool | Platform | #Releases | #Loc | #Test Cases |
|------|----------|-----------|------|-------------|
| P1   | Mobile   | 3         | 77.494 | 1.336       |
| P2   | Web      | 2         | 230.874 | 1.491       |
Collect indicators to identify Test Debts and indicators already collected in GTF (e.g., number of bugs identified by demand, number of test cases by demand, planned effort versus actual effort). The indicators collected will be detailed in subsection 4.3.

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 same meeting that defines the current demand test scope. The adaptations of the activities performed in 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 in the next subsections.

4.2.1 First catalog use

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

After starting the steps established for the use 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.

For the first catalog use, the team chose to deal 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 GTF. It is worth noting that this subtype was the first Test Debt subtype identified in the test factory even before the existence of the catalog.

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 GTF to perform the management of the Test Debt subtypes.

Table 12 presents a summary of the actions chosen in this first use of the catalog. It can be seen that the team prioritized actions that were already fully or partially performed by GTF and that would have less impact on the activities already performed by the team.

4.2.2 Second catalog use

The second use of the catalog still occurred in the P1 project. 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; (ii) Lack of tests; and (iii) Test effort estimation errors. All these subtypes were identified in the empirical study carried out in GTF and it was known to the team that they frequently occurred in GTF. Thus, the team decided to manage them in this second use of the catalog.

After defining the Test Debt subtypes, the team defined which actions would be performed to manage the chosen subtypes. Table 13 shows the actions selected for this new catalog use. It is worth noting that the management activities in which the actions were not altered in relation to the previous use were omitted in the table, yet all the management activities were performed.
Table 12  First use of TestDCat

| SELECTED TEST DEBT SUBTYPES | SELECTED ACTIONS |
|-----------------------------|------------------|
| Defects Not found in tests  |                  |

| 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 | Monitor changes in the cost/benefit ratio of the identified debt meetings | Change test cases by analyzing defects | Standardize debt documentation | Present already identified debts |

| REASONS FOR THE CHOICE       |                  |                 |                          |                          |                          |                          |                          |
|------------------------------|-----------------|----------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|
| Identify defects that affect the end-user | 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 has already used to deal with problems of this type | Facilitate understanding of TDs | Mitigate the events of identical or very similar TDs |

| CHANGES IN ACTIVITIES        |                  |                 |                          |                          |                          |                          |                          |
|------------------------------|-----------------|----------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|
### Table 12 (continued)

| SELECTED TEST DEBT SUBTYPES |
|-----------------------------|
| Defects Not found in tests  |

#### SELECTED ACTIONS

| IDENTIFICATION | MEASUREMENT | PRIORITIZATION | COMMUNICATION | MONITORING | REPAYMENT | DOCUMENTATION | PREVENTION |
|----------------|-------------|----------------|---------------|------------|-----------|----------------|------------|
| 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. |
Table 13  Second use of TestDCat

| SELECTED TEST DEBT SUBTYPES | SELECTED ACTIONS |
|-----------------------------|------------------|
| Defects not found in test   | 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 |
| Lack of test                | 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 |
| Test effort estimation errors | To identify Test effort estimation errors: -Analyze reported effort |

| 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 | CHANGES IN ACTIVITIES |
|------------------------|-----------------------|
| -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 |
### Table 13 (continued)

#### SELECTED TEST DEBT SUBTYPES

| Defects not found in test | Lack of test | Test effort estimation errors |
|---------------------------|--------------|------------------------------|

#### SELECTED ACTIONS

| The following activities were added to the testing process: (i) Analyze emails 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 |
### Table 14  Third use of TestDCat

**SELECTED TEST DEBT SUBTYPES**
- Test effort estimation errors
- Inadequate equipment
- Inadequate allocations

**SELECTED ACTIONS**

| IDENTIFICATION | PRIORITIZATION | COMMUNICATION | REPAYMENT |
|----------------|----------------|---------------|-----------|
| - 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 | 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**
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**
| Table 14 (continued) |
|-----------------------|

**SELECTED TEST DEBT SUBTYPES**

- Test effort estimation errors
- Inadequate equipment
- Inadequate allocations

**SELECTED ACTIONS**

| IDENTIFICATION | PRIORITIZATION | COMMUNICATION | REPAYMENT |
|----------------|----------------|---------------|-----------|
| 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 |
4.2.3 Third catalog use

The third use of the catalog was performed in the P1 product. In this use, 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 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 14 presents details of the actions chosen to manage the subtypes of Test Debts selected in this third use. 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 catalog uses.

4.2.4 Fourth catalog use

The fourth use of the catalog was carried out in the P2 software product. For this use, 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 the previous uses 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 TestDCat). The subtype Inadequate allocation was chosen because a new GTF member was working in the test of this product.

Table 15 shows the actions chosen to manage the previously selected Test Debt subtypes. As occurred in the explanation of the previous uses, the management activities whose actions were not changed in relation to the previous uses were omitted in the table.

4.2.5 Fifth catalog use

The fifth and last use of the catalog in the scope of this case study was performed in the P2 product. For this use, 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 use, no new actions were performed other than those already carried out in the previous uses performed in the P2 product.

4.3 Results

During the use of the catalog, some metrics were collected and a survey was applied with the participants of the case study in order to collect feedback on, for example, the catalog usability and effectiveness.
Table 15  Fourth use of TestDCat

**SELECTED TEST DEBT SUBTYPES**

Deferring tests

Lack of test

Test estimation errors

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

**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.
| 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                 | 25                | 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            | Test Case Point (TCP)                       | 7.2               | 6                  | 8.4               | 3.6               | 1.8               |
|                        | Number of tests run per time period         | 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 |
4.3.1 Collected metrics

The metrics collected were part of the GTF process and are based on articles available in white literature Seela and Yackel (2017) and academic Lazic and Mastorakis (2008). In addition to the debts already collected in GTF, the metric *Number of Test Debts* was also collected to measure the amount of Test Debts identified during the test demands.

Table 16 presents the results of the metrics collected after the use of the catalog in each of the test demands for projects P1 and P2.

![Graph showing test debt status in releases P1 and P2](image)
With regard to Test Debts, Fig. 3 presents the number of Test Debts identified, repaid (refers to carrying out activities to resolve or mitigate a known Test Debt), and not repaid during the releases of P1 and P2 products.

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

### 4.3.2 Survey results

A survey was applied with the participants of the case study. 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 in 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 the case study.

**The participants profile.** As stated in subsection 4.1.3, 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 17 presents the profile of the participants who responded to the survey.

**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.

**General questions about the catalog.** 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.

---

3 Survey applied after conducting the case study - https://forms.gle/4Gn1jqpV9cHmgvej8
Figure 4 presents the answers collected from the third part of the applied survey.

4.4 Discussion

This section discusses 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.

4.4.1 Collected metrics

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

In the first use, 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 use, 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 use, 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 use 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 use, 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.

### 4.4.2 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. (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. (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.

4.4.3 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.

**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.

**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.

4.4.4 Threats to validity

Regarding the threats to validity, we discuss threats related to Internal Validity and External Validity. According to Wohlin et al. (2012) and Runeson and 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.

5 Related work

Regarding Test Debt, Samartyam et al. (2017) present an overview of TDs, the 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, the authors 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 for test automation and in the accomplishment of testing activities. These authors also describe two case studies in industry and report their experiences with Test Debts. However, they do not propose an integration of these steps with an existing testing process in the organization. Also, they do not detail the two case studies and the impacts of paying their Test Debts.

Regarding Test Debt management in a testing process, the work of Sousa (2016) aims to identify, within the software testing process, problems that could be considered as Technical Debt. Sousa presents a set of 22 TDs related to the software testing process collected from technical literature, besides their eventual causes and indicators. To evaluate these TDs, they conducted a survey with test professionals who answered if they agreed with each TD, its causes and indicators. Furthermore, Sousa presents a map with the Technical Debts identified in the literature 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. Sousa, however, does not verify other important activities of TD 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.

Other studies have investigated the impact of testing types or techniques in generating TDs. For example, 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 TD
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.

Likewise, Wiklund et al. (2012) present studies that indicate a general trend of high Technical Debt in many test execution automation implementations, for example, leading to problems for the use, extension, and maintenance of 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 authors 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 those debts.

Table 18 presents a comparison of the aforementioned related work, regarding the following characteristics: (i) Comprehensive approach, which indicates that the study involves comprehensive strategies to manage Test Debts in the software industry considering the testing types and test process activities; (ii) Target, which refers to the Test Debts target of the study; (iii) Test Debt management in a testing process, which denotes that the work proposes Technical Debt management activities within test process (e.g., changes in a test process in order to manage the Test Debt); and (iv) Conduction of a case study, which indicates whether the work conducted a case study in which some Test Debt management activity, approach or strategy has been applied.

None of the works presents a comprehensive approach. 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). Besides that, the Test Debt target varies from test types Samartyam et al. (2017); Shah et al. (2014) and test process Sousa (2016) to how the testing is performed Wiklund et al. (2012).

With regard to the characteristic Test Debt management in a testing process, only the work of Sousa et al. (2016) proposes the management of Test Debts in an integrated manner with the testing process. Regarding the conduction of a case study, only the work of Samartyam et al. (2017) presents case studies applying the approaches presented for Test Debt management.

Therefore, we have concluded that none of the studies cover all the characteristics presented or implement them only partially, different of our work that proposes the TestDCat 3.0, a comprehensive catalog to manage the Technical Debts related to both test subtypes and process. It proposed management activities based on the test process presented by ISO/IEC 29119 and its effectiveness was measured through a case study performed with industry products.

6 Conclusion

Test Debts have a high impact on software quality. Therefore, 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 practitioners in the management of Test Debts, a catalog, called TestDCat, was created. The 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. They answered a survey regarding clarity, ease of use, and completeness. In the second evaluation, a focus group was organized to analyze in detail 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 (3.0) of the TestDCat, a case study was conducted in the GTF. It aimed to evaluate the application of TestDCat in real products. Five applications of the catalog were performed in two software products, 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.

From the results of this work, the main future research directions are described as follows:

- Improve the catalog by inserting more specifics for each proposed action (e.g., tools and steps taken to perform the action). These improvements can be identified when the practitioner/researcher uses in real situations the actions in their projects and documents all activities, artifacts and tools used to carry out the action.
- In order to make the catalog more comprehensive and possible to use in several organizations, and to identify new Test Debts, it is interesting to expand the literature review carried out with a systematic review of Test Debts and their management.
- There are different types and subtypes of Technical Debt, but some of them are related in some way (e.g., dependency). Thus, it is important to 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.
- 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.

With the realization of these future works, the catalog would be more complete and with more specific practices and tools to assist practitioners in managing their Test Debts.

References

Alves, N. S., Mendes, T. S., de Mendonca, M. G., Spinola, R. O., Shull, F., Seaman, C. (2016). Identification and management of Technical Debt: A systematic mapping study. Information and Software Technology, 70, 100–121.
Andrade, R. M., Lelli, V., Castro, R. N., Santos, I. S. (2017). 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) (pp. 10-16). IEEE. https://doi.org/10.1109/SER-IP.2017.2.

Andrade, R. M. C., Santos, I. S., Lelli, V., de Oliveira, K. M. da Rocha, A. R. C. (2017). Software testing process in a test factory - from ad hoc activities to an organizational standard. In ICEPT.

Aragão, B. S., de Castro, R. N. S., de Sousa Santos, I., Lelli, V., Andrade, R. M. (2019). Test Debts identification in a test factory. In Proceedings of the XVIII Brazilian Symposium on Software Quality (pp. 296-305). Association for Computing Machinery, New York, USA. https://doi.org/10.1145/3364641.3364676.

Aragão, B. S., Andrade, R. M. C., Santos, I. S., Castro, R. N. S., Lelli, V., Darin, T. G. R. (2019). Test-dcat: Catalog of Test Debt subtypes and management activities. In C. Gaston, N. Kosmatov, P. Le Gall (Eds.) Testing Software and Systems (pp. 279-295). Springer International Publishing, Cham.

Bangor, A., Kortum, P., & Miller, J.A. (2008). The System Usability Scale (SUS): An Empirical Evaluation, International Journal of Human-Computer Interaction, 24(6).

Brooke, J. (1996). Sus-a quick and dirty usability scale. Usability evaluation in industry, 189(194), 4–7.

Chung, L., Nixon, B. A., Yu, E., Mylopoulos, J. (2012). Non-functional requirements in software engineering (Vol. 5). Springer Science & Business Media.

Cunningham, W. (1992). The wycash portfolio management system. SIGPLAN OOPS Mess., 4(2), 29–30. https://doi.org/10.1145/157710.157715.

Fernandes, F., Sousa, S., Lopes, I. D. S. (2013). On the use of quality tools: a case study. In World Congress on Engineering 2013 (Vol. 1, pp. 634-639). Newwood Limited Publisher.

Gorschek, T., Garre, P., Larsson, S., Wohlin, C. (2006). A model for technology transfer in practice. IEEE software, 23(6), 88–95.

Kruchten, P., Nord, R. L., Ozkaya, I., Visser, J. (2012). Technical Debt in software development: From metaphor to theory report on the third international workshop on managing Technical Debt. SIGSOFT Softw. Eng. Notes, 37(5), 36–38. https://doi.org/10.1145/2347696.2347698.

Krueger, R. A., & Casey, M. A. (2002). Designing and conducting focus group interviews. In Designing and Conducting Focus Group Interviews, 2nd ed. Sage Publications, Inc.

Lazic, L., & Mastorakis, N. (2008). Cost effective software test metrics. WSEAS Transactions on Computers, 7(6), 599–619.

Li, Z., Avgeriou, P., Liang, P. (2015). A systematic mapping study on Technical Debt and its management. Journal of Systems and Software, 101, 193–220.

Ozkaya, I., Kruchten, P., Nord, R. L., Brown, N. (2011). Managing Technical Debt in software development: Report on the 2nd international workshop on managing Technical Debt, held at icse 2011. SIGSOFT Softw. Eng. Notes, 36(5), 33-35. https://doi.acm.org/10.1145/2020976.2020979.

Runezon, P., & Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. Empirical software engineering, 14(2), 131.

Samarthyam, G., Muralidharan, M., & Anna, R. K. (2017). Understanding Test Debt. In Trends in Software Testing (pp. 1-17). Springer.

Seela, S., & Yackel, R. (2017). 64 essential testing metrics for measuring quality assurance success. https://www.qasymphony.com/blog/64-test-metrics/.

Shah, S. M. A., Torchiano, M., Vetro, A., Morisio, M. (2014). Exploratory testing as a source of Technical Debt. IT Professional, 16(3), 44–51.

Sousa, C. L. D. (2016). Mapa de apoio à gestão de dívida técnica no processo de teste de software. Master thesis, Universidade Federal de Pernambuco.

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

Wiklund, K., Eldh, S., Sundmark, D., Lundqvist, K. (2012). Technical Debt in test automation. In 2012 IEEE Fifth International Conference on Software Testing, Verification and Validation (pp. 887-892). https://doi.org/10.1109/ICST.2012.192.

Wiklund, K., Eldh, S., Sundmark, D., Lundqvist, K. (2017). Impediments for software test automation: A systematic literature review. Software Testing, Verification and Reliability, 27(8).

Wohlin, C. (2014) Guidelines for snowballing in systematic literature studies and a replica- tion in software engineering. In Proceedings of the 18th international conference on evaluation and assessment in software engineering, EASE 14 (pp. 38:1-38:10). https://doi.org/10.1145/2601248.2601268.

Wohlin, C., Runezon, P., Hst, M., Ohlsson, M. C., Regnell, B., Wessln, A. (2012). Experimentation in Software Engineering. Incorporated: Springer Publishing Company.
Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Bruno Sabóia Aragão received the M.Sc in Computer Science from the Federal University of Ceara (UFC) and a Bachelor’s degree in Computer Science at the University of Ceara (UECE). He is currently project manager at GREat (Software and Systems Engineering Network Group) at UFC. He has experience in projects using Scrum, Software Testing, Web Systems Development, and Mobile Applications. He is also Scrum Master Certified (Scrum Alliance).

Rossana Maria de Castro Andrade is an associate professor at Federal University of Ceará, Department of Computer Science, and the Founding Director of the GREat Research Group. She received her Ph.D from University of Ottawa, SITE, in 2001. Her research interests are computer networks and software engineering, specifically, she is looking at the challenges of developing and testing ubiquitous and mobile applications as well as alternative solutions to increase systems security. Over the past 15 years, she has coordinated several research projects with the government and the industry, supervised master and PhD students, and maintained a steady and relevant level of publications. She is also a member of ACM, IEEE and the Brazilian Computer Society. Contact her at rossana@ufc.br.

Ismayle de Sousa Santos received the M.Sc. and Ph.D degrees in Computer Science from the Federal University of Ceará (UFC), Brazil, in 2013 and 2017. His research interests are Software Testing, Adaptive Systems and Experimental Software Engineering. He is currently postdoctoral researcher at the Group of Computer Networks, Software Engineering and Systems (GREat) lab.
Rute Castro received the M.Sc in Computer Science from the Federal University of Ceará (2006) and a Bachelor’s degree in Computer Science from the Federal University of Ceará (2003). She currently works at GREat/UFC (Computer Networks, Software and Systems Engineering Group) as Project Manager. She has experience in Software Engineering and emphasis on software standards, project management of application development for mobile devices and natural language processing.

Valéria Lelli received her Ph.D in Informatics from the National Institute of Applied Sciences (INSA) in Rennes, France. She holds a master’s degree and a bachelor’s degree in Computer Science from the Federal University of Ceará (UFC). She has experience in Software Engineering and Mobile Computing, working mainly in software testing with emphasis on mobile applications and graphical user interfaces. She worked as researcher and project manager at GREat/UFC in software development projects and collaborated in the implementation of the testing process at the GREat/UFC testing factory. Currently, she is an assistant professor at the Federal University of Ceará.

Ticianne Darin has a Master degree in Computer Science from the Federal University of Ceará (2011) and a PhD degree in Computer Science from the same university (2018), with a sandwich period at the Universidad de Chile as a student visiting researcher. She is currently professor at the Federal University of Ceará and teaches Human-Computer Interaction, Design Theory and Criticism, Emotional Design, Evaluation of Interaction and History of Design in the Bachelor’s Degree in Digital Systems and Media. Her research interests focus especially on IHC evaluation in multimodal interfaces and games, besides Interaction Design, Cognitive Psychology and Multimedia Design.