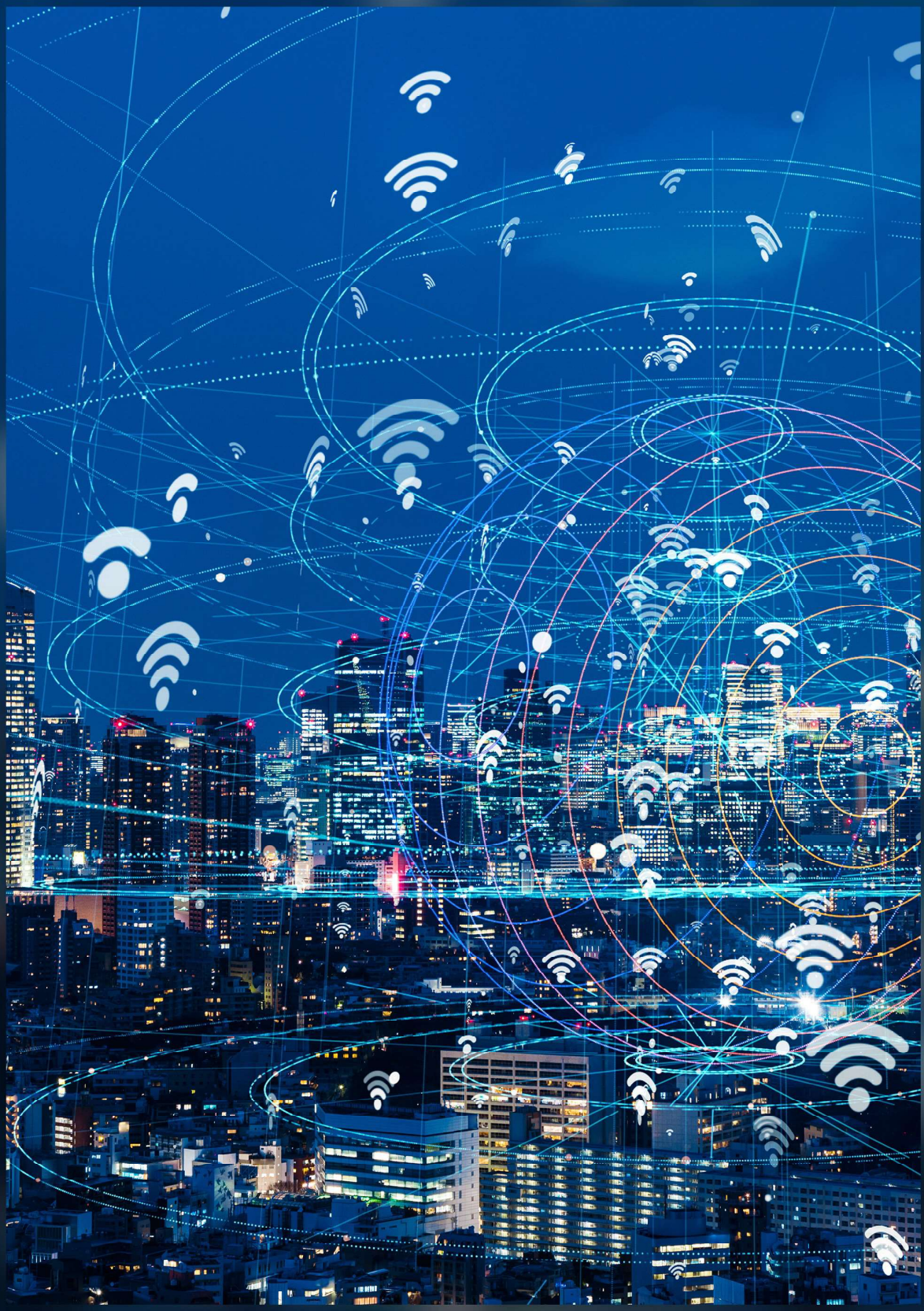
SINGULAR®

ENGENHARIA, TECNOLOGIA E GESTÃO



ANO 1, V.1, Nº 2 - OUTUBRO DE 2019

CEULP/ULBRA
CENTRO UNIVERSITÁRIO LUTERANO DE PALMAS

Singular Engenharia, Tecnologia e Gestão

Vol. 1, N. 2, Outubro, 2019
eISSN: 2596-2604
https://doi.org/10.33911/singular-etg.v1i2

Editada em outubro de 2019.
Última edição em outubro de 2019.
Publicada em outubro de 2019.

Nota da Editora: Os autores são responsáveis pela apresentação dos fatos contidos e opiniões expressas nesta obra.

Dados Internacionais de Catalogação na Publicação na (CIP)

R454 Singular Engenharia, Tecnologia e Gestão [recurso eletrônico] / Centro Universitário Luterano de Palmas. – Volume 1, n. 2 (out./2019). Dados eletrônicos. – Palmas: Ceulp, 2019-

Semestral.
Modo de Acesso: World Wide Web:
<http://ulbra-to.br/singular/index.php/SingularETG/>
Título varia: Revista Singular ETG.
Descrição baseada em: Volume 1, n. 2 (out.2019).
eISSN: 2596-2604
doi: 10.33911/singular-etg.v1i2

I. Interdisciplinar. II. Computação. III. Engenharia. IV. Gestão. V. Título: Singular Engenharia, Tecnologia e Gestão.

CDU: 001

Ficha catalográfica elaborada pela bibliotecária Thaís Fernandes, CRB-2/1680

SINGULAR ENGENHARIA, TECNOLOGIA E GESTÃO
CENTRO UNIVERSITÁRIO LUTERANO DE PALMAS
Avenida Teotônio Segurado 1501 Sul
Palmas - TO CEP 77.019-900 Caixa Postal nº 85
Fone: (63) 3219 8125email: revista.singular@ceulp.edu.br

EDITORIAL

A Singular Engenharia, Tecnologia e Gestão apresenta à comunidade científica sua segunda edição. Fruto de um trabalho incansável de sua editoria, esta revista busca oferecer um espaço a mais para apresentação de artigos que contribuam significativamente para estudos nestes campos.

A Singular Engenharia, Tecnologia e Gestão é uma publicação semestral, que se propõem a contribuir na difusão de debates e ideias no espaço acadêmico e, com isso, criar uma rede de pesquisadores com perspectivas teóricas e metodológicas diversas, propiciando a troca de informações e o debate sobre as principais questões nesses campos.

A partir desta segunda edição reafirmamos o compromisso com a comunidade científica no objetivo de buscar atender aos seus anseios e necessidades no que diz respeito à um espaço de publicação e apresentação de resultados das pesquisas realizadas. Estamos abertos à todo tipo de contribuição neste sentido.

Agradecemos aos pesquisadores que submeteram seus trabalhos para esta edição.

Boa leitura.

Palmas-TO, outubro de 2019.

SUMÁRIO

SINGULAR®
ENGENHARIA, TECNOLOGIA E GESTÃO

Publicação: Outubro de 2019

Digital Object Identifier https://doi.org/10.33911/singular-etg.v1i2.55

Android Applications based on software repository analysis

TAYSE VIRGULINO RIBEIRO[1], MÁRCIO LOPES CORNÉLIO[2]

1-Master of Science in Computer Science - Center of Informatic (CIn), Universidade Federal de Pernambuco - UFPE, Recife/PE - Brazil (e-mail: tvr@cin.ufpe.br)
2- Assistant Professor Centro de Informática · Universidade Federal de Pernambuco PhD in Computer Science, Centro de Informática (e-mail: mlc2@cin.ufpe.br)
Autor Correspondente: Tayse Virgulino Ribeiro (e-mail: tvr@cin.ufpe.br).

Context: Software repositories have been a source for studies about software evolution and its relation to software defects. In addition, the context of repositories have also been used for the purpose of analyzing refactoring practiced by programmers throughout the development process. **Objective**: Our objective is based on android projects stored in software repositories, to determine what types of transformations, that is, which refactoring are used, seeking to relate them to quality and security factors. **Method**: This research uses as an approach an exploratory study of a qualitative character, based on a systematic review of the literature, which will be carried out between the period from 2015 to 2019, as well as application of research and quality criteria regarding the work context. In addition, develop a case study with projects for Android, relating refactoring quality criteria to non-aggregated projects in software repositories, glimpsing comparative and resulting factors. **Expected results**: It is expected with this review an analysis and a summary of existing literature on Code Quality in the process of Software Refactoring for Android projects. **Conclusions**: The research is guided by this approach in identifying the types of refactoring practiced and extracting the related quality factors in the development process. We believe that our results will benefit in the updating and summary of the literature in the context of refactoring, glimpsing comparative factors.

Software projects, Android projects, software refactoring, metrics quality repositories software.

I. INTRODUCTION

The research related to Systematic Review of Literature (SRL) is necessary as software repositories have been a source for studies that establish the relationship between evolution activities on software defects [1], which allow the measurement of contributions from developers [2] in software projects. In addition, repositories have also been used to analyze refactoring practiced by programmers throughout the development process [3]. In the case of mobile platforms, in particular for Android systems, applications have been analyzed with various static analysis tools in order to determine, for example, the excess permissions or potential bugs in different versions [4].

Understanding how software is created and preserved is essential to define how to make it faster, cheaper, and with higher quality. One way to get valuable information about the development process is to analyze existing projects: source code, how the application has evolved over time, and attributes such as security, failure, and size [4].

Different program understanding studies show that programmers rely on good software documentation [5]. In addition to improving, standardizing the quality of documentation and ensuring information security.

Out, the problem of this research is focused on identifying which refactoring are involved in the development process to achieve the software quality factor? Also, what refactoring are related to a specific quality model?

In addition, the work uses as an approach an exploratory study of qualitative character, based on

the accomplishment of a rapid systematic review of the literature. As well, a case study with projects for Android is carried out in the context of software repositories, looking for applicability of quality criteria.

This paper is organized as follows. In Section 2, we introduce fundamental concepts about of software repositories and software quality. In Section 3, we describe the research method we adopt. In section 4 and 5, we present results and considerations of the exploratory study presented. Finally, in Section 6, we present out conclusions.

II. CONCEPTUAL BACKGROUND AND RELATED WORKS

This section presents the main concepts, theories and research challenges directed at the systematic review and the case study of this qualitative research.

A. REFACTORING

Refactoring is the process of changing a software system in such a way that it does not alter the external behavior of the code yet improves its internal structure. It is a disciplined way to clean up code that minimizes the chances of introducing bugs. In essence when you refactor you are improving the design of the code after it has been written [6].

"Improving the design after it has been written." That's an odd turn of phrase. In our current understanding of software development, we believe that we design and then we code. A good design comes first, and the coding comes second. Over time the code will be modified, and the integrity of the system, its structure according to that design, gradually fades. The code slowly sinks from engineering to hacking [6].

Refactoring is the opposite of this practice. With refactoring you can take a bad design, chaos even, and rework it into well-designed code. Each step is simple, even simplistic. You move a field from one class to another, pull some code out of a method to make into its own method, and push some code up or down a hierarchy. Yet the cumulative effect of these small changes can radically improve the design. It is the exact reverse of the normal notion of software decay [6].

With refactoring you find the balance of work changes. You find that design, rather than occurring all up front, occurs continuously during development. You learn from building the system how to improve the design. The resulting interaction leads to a program with a design that stays good as development continues [6].

In 1992, Willian Opdyke's thesis was the first work in the context of refactoring. The focus of the thesis was to automate the refactoring in a way that preserved the behavior of a program. This thesis establishes a set of refactoring operations that support the design, evolution and reuse of frameworks of object-oriented applications [17].

Contextualizes the design of reusable software in an especially difficult way [17]. In general, reusable Software is the result of many design iterations. These occur after the software has been reused and the resulting changes affect not only the design but also the design of other software that is using it. Therefore, making the software easier to modify facilitates subsequent design iterations and makes the software more reusable [17].

It is generally addressed in Opdyke's thesis that, for some refactoring, one or more of its preconditions are undecidable. I n this thesis he defines 23 primitive refactoring and shows three examples of compound refactoring. For primitives, a set of preconditions provides the notion of software behavior. The set of complex refactoring is defined in detail: generalization of the inheritance hierarchy, specialization of the inheritance hierarchy and use of aggregations to model relationships between classes [17].

Finally, this thesis explores several operations for applicability in the process of evolution and development of object-oriented applications. As well, it provides some conservative algorithms to determine if a program satisfies these constraints and describes how to use this design information to refactor a program [17].

In 1999, Don Roberts in his doctoral thesis continues the work of Opdyke [17], adding postconditions and developing the first refactoring tool. In his thesis, [18] provides:

> [...] a new definition of refactoring that focuses on preconditions and postconditions of refactorings rather than on program transformation itself. Preconditions are statements that a program must satisfy for refactoring to be applied, and postconditions specify how the assertions are transformed by refactoring. Post-conditions can be used for a number of purposes: to reduce the amount of analysis that subsequent refactorings must perform, derive preconditions for compound refactorings, and calculate dependencies between

refactorings.

In addition to examining techniques to aid refactoring, it presents the Refactoring Browser design, a Smalltalk refactoring tool that is used by commercial software developers and identifies the criteria necessary for any refactoring tool to be successful [18].

Thus, it redefines the refactorings proposed by [17], dividing them into three groups: class refactoring, method refactoring, and variable refactoring [19].

Refactoring, reuse e reality

Refactoring is an overhead activity [6]:

- Tools and technologies are available to allow refactoring to be done quickly and relatively painlessly
- Experiences reported by some object-oriented programmers suggest that the overhead of refactoring is more than compensated by reduced efforts and intervals in other phases of program development.
- Although refactoring may seem a bit awkward and an overhead item at first, as it becomes part of a software development regimen, it stops feeling like overhead and starts feeling like an essential.

How does one safely refactor? There are several options [6]:

- Trust your coding abilities.
- Trust that your compiler will catch errors that you miss.
- Trust that your test suite will catch errors that you and your compiler miss.
- Trust that code review will catch errors that you, your compiler, and your test suite miss.

The real-world concerns regarding a reuse program are similar to those related to refactoring [6].

- Technical staff may not understand what to reuse or how to reuse it.
- Technical staff may not be motivated to apply a reuse approach unless short-term benefits can be achieved.
- Overhead, learning curve, and discovery cost issues must be addressed for a reuse approach to be successfully adopted.
- Adopting a reuse approach should not be disruptive to a project; there may be Strong pressures to leverage existing assets or implementation albeit with legacy constraints. New implementations should interwork or be backward compatible with existing systems.

B. QUALITY OF SOFTWARE

1) Project of software: android

Android has grown to be the world's most popular mobile platform with apps that are capable of doing everything from checking sports scores to purchasing stocks. In order to assist researchers and developers in better understanding the development process as well as the current state of the apps themselves, we present a large dataset of analyzed open-source Android applications and provide a brief analysis of the data, demonstrating potential usefulness [4].

Android has become an extremely popular mobile platform, and Android apps are not immune to the problems which have hindered traditional software — especially security vulnerabilities, high maintenance costs, and bugs. Understanding how software is created and maintained is paramount in determining how to produce it faster, cheaper, and of higher quality. One way to gain valuable insight into the development process is to examine existing projects: source code, how the app has evolved over time, and attributes such as its security, defects, and size. App source code may be analyzed using static analysis tools, providing data about the software's security risk level, possible defects, or even lack of adherence to coding standards [4].

A dataset of Android applications with the results of the static analysis tools we have created is an important tool for understanding how Android applications are developed and maintained [4].

2) Software testing repositories: software bugs

Studying the evolution and understanding the structure of large legacy systems are two key issues in software industry that are being tackled by academic research. These problems are strongly coupled for various reasons: (i) examining the structure of subsystems allows us to gain a better understanding of the whole system evolution; (ii) the histories of software entities can reveal hidden relationships among them and (iii) analyzing several versions of a system improves our understanding of it [7].

3) Repositories of software: excess permissions

According [8], one of the most important principles of good computer security is the principle of least privilege: A user should have no more access to data and systems than is necessary for their task. Too often, security problems result from users having excessive privileges and excessive access to data.

In the area of software refactoring, some of the related works are worth referencing:

Firouzi [12] discusses automated refactoring tools in Visual Studio, analyzing their effectiveness with the success rate of the builds and version control system. This paper examines the actual impact of using ReSharper tools on the results of test runs, Builds and Version control commands to evaluate their impact using some features such as Quick-fixes, Context actions, and Refactorings. In this study, they investigate whether the above tools meet the expectations. In order to obtain the necessary information, Firouzi used the Data Set Enriched Event Flows provided by the MSR challenge in 2017. In the pre-process phase an application was developed that iterated all user zip files and all JSON files and converted the dataset into a relational database, and information stored in the SQL Server database.

It is a study focused on the Enhanced Events Stream Dataset and the presented results may not generalize well in other contexts, regardless of statistical justifications and significance calculations. In addition, other factors may influence the results, but due to the abstraction of the data set or not being collected could not be investigated. The results suggest that the automated refactoring tools of Visual Studio, ReSharper, may not consider an overview of the solutions [12].

In [13] software metrics are employed in the development and maintenance of software to evaluate different quality attributes, software design, test and reengineering processes. In this case, the software metrics used in relation to the project standards went through an analysis process. According to Derezińska [13], they focused on "classical" design patterns (DP, in short) applied to object-oriented software. Different metrics are used in evaluating the impact of the design pattern on selected quality attributes, for example, software maintenance, flexibility to change input, performance, susceptibility to failure, among others. One of the quality models of object-oriented software is QMOOD. Different object-oriented features are associated with quantitative measures such as SIZE, NOC, DIT, DAM, CBO, CAM, MOA, MFA, NOP, RFC, WMPC.

To automate the introduction of design patterns in the existing code, a refactoring process was proposed, this process is based on relevance metrics that support the recommendation of standards. The metrics were implemented in a prototype tool that supports refactoring to design standards. The tool extends the Eclipse environment and transforms Java programs. The main contribution of the paper is a new approach based on metrics for refactoring. The approach can be used as a recommendation presented to a user or can provide partial or fully automated refactoring. Preliminary experiments with the prototype on the refactoring of Java programs to design patterns have confirmed the profitability of the approach [16].

In article [14] the effect of clone refactoring (CR) on the size of unit test cases in object-oriented (OO) software is evaluated empirically. In general, the contributions of this paper are: (1) strong and positive correlation between the CR code and the reduction in the size of the unit test cases, (2) showed how the code quality attributes related to the testability of the classes are significantly improved when clones are refactored (3) the size of unit test cases can be significantly reduced when CR is applied, and (4) complexity/size measures are commonly associated with variations in the size of unit test cases when compared to coupling. The selected source code metrics address the size, complexity, and coupling attributes.

In the methodology was collected data from two open source Java software systems, ANT and ARCHIVA, which were refactored in clones. As reported by Badri [14], to quantify the size of unit test cases, we used two test code metrics already used in many previous empirical studies and the size of unit test cases. In addition, the results were based on three different techniques: single-exit cross-validation (LOOCV), ten-fold cross-validation (10FCV) and cross-project validation (IPCV).

In this article [15] explores better understanding of performance issues in mobile applications by focusing our study on iOS. We conducted an empirical study of 225 performance problem reports on four free software iOS applications written in the Swift programming language to study common types of performance problems. IOS applications are generally developed using the Model-View-Controller (MVC) design pattern. The layers in the MVC help to abstract the underlying device differences, such as screen sizes, and simplify application development. The study of problem reports found that inefficient user interface design, memory problems, and inefficient thread manipulation are the most common types of performance problems. These four anti-patterns were documented and a static analysis tool, called iPerfDetector, was developed to detect these patterns, evaluated in 11 applications in IOS.

In [16], it is discussed the importance of refactoring in software engineering and the difficulties that can be faced with the application of refactoring. In addition to providing an overview of the refactoring

process and discussing the critical components of this practice. Quality metrics are used to discover design flaws in software systems or measure the quality of the code in specific ways. In the literature [16], "Bad Smells" is a term used to describe common structural problems in the code that need to be eliminated to make the code more sustainable. To achieve this, an appropriate refactoring operation needs to be performed. Examples of "Bad smells" metrics: Duplicated code, Long method, Larg class, and Long paramter list.

For the most part, the refactoring deals with the static relationships between the properties of the units in the source code. For small software systems, structure or design can be considered as an aesthetic question. For a larger software design and company design, they become much more important because of their direct impact on cost. Therefore, a higher quality project should be the goal and this project needs continuous refinements to preserve the initial quality against software evolution. We briefly discuss the refactoring process and discuss several examples of "Bad smell", design problems that degrade software quality and proposed refactoring solutions [16].

Four different types of very important tools for successful refactoring have been listed: static analysis tools, visualization tools designed to support refactoring, refactoring tools for performing the refactoring process, and automated testing tools to verify such refactoring operations. Refactoring actually preserve the behavior of the system. The place of refactoring in industry and academia was addressed. In addition to highlighting that refactoring is one of the main practices in agile development processes, such as extreme programming [16].

III. METHOD

In a different direction, this research aims to explore activities based on android projects stored in software repositories, in search of which types of refactorings are used in relation to quality factors in the development and measurement of the code. From this perspective analyze and summary the existing literature, as well as conduct a case study with Android projects, relating to refactoring quality criteria in order to glimpse comparative factors. Our research question is formulated as follows:

Based on android projects, what types of refactoring are used, trying to relate them to quality factors?

From this, the research question of this work is derived from the definition of the following elements:

- *Context: projects for android*
- *Intervention: quality factors*
- *Result: types of refactorings*

This study is based on a systematic review, addressing a qualitative exploratory case study. Figure 1 shows the flow of the methodological proposal that this research will guide.

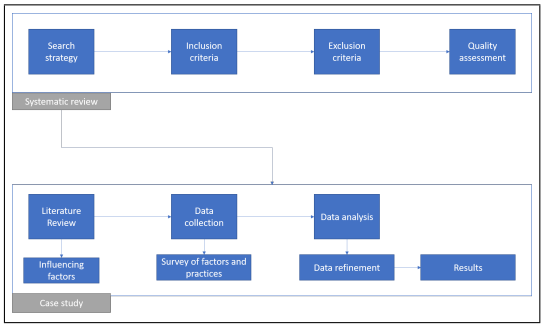


Fig. 1: Methodological flow of research

The research will comprise two phases, as shown in Figure 1, Systematic review and Case study.

In the first stage we understand Systematic review. First, in this process the Search strategy step must be carried out, which will follow the basic scope of definition of the search methods, construction of the search string and definition of the search source. The initial proposal of the protocol is based on a manual search in specific sources to obtain a set of known articles that allow us to measure the level of completeness of our research and help in the construction of our search string that will be used in the automatic search. This research aimed to analyze periodic and events related to Code Refactoring, as well as studies classified as Software Repositories of Mining (MSR), with the help of the use of Springer, IEEE and ACM databases.

Already in the automatic search we have as aid the use of the search engine Scopus. After this process of searching the engines, we be carried out the process of selection of studies. Some inclusion criteria can be performed to filter the most relevant studies for the research.

In the strategy of the research, we still designated some criteria to categorize the study. According to Hulley [9], the inclusion and exclusion criteria are based on a standard and necessary practice in the elaboration of high-quality protocols. In addition, characteristics of the target unit of analysis that researchers will use to answer the study question. While the exclusion criteria include eligible characteristics that make them have a high chance of loss of follow-up.

For this research, the following inclusion criteria will be assigned, taking into account that the researchers seek to identify the maximum of applied trends in code quality and software refactoring process. The protocol of this research chose to include studies written only in English, and not to differentiate in profiles of students and professionals. With regard to the publication time limit, publications from 2011 up to the current year will be included. The inclusion criteria of the studies are presented in Table 1:

1lightgray

Tab. 1: Inclusion Criterion.

Criterion	Inclusion Criterion Description
CI1	Language of the article: English
CI2	Publish date limit: 2011 to 2019
CI3	Primary studies that address the area of research knowledge
CI4	Target population: software engineers, software developers, programmers and related areas of computing.

On the other hand, the studies that do not have a focus on the area of computing and related or that do not have relevance to the research will be excluded. As well, studies without theoretical foundation and that are not related to the research question. As well as articles that were previously published the year 2011. The exclusion criteria for the retrieved studies will be presented in Table 2.

1lightgray

Tab. 2: Exclusion Criterion.

Criterion	Description of Exclusion Criterion
CE1	Articles with no theoretical basis and not related to the research question
CE2	Articles that are not focused on computing or that have no relevance
CE3	Articles before 2011
CE4	Language of the article: Portuguese.

In addition to these criteria, in this process a quality evaluation is performed, that is, a list of criteria necessary to meet the quality of the study. Initially, to carry out this quality analysis, 6 criteria were defined and evaluated according to a scale composed of the following values:

- (0) when the criterion is not met;
- (1) when the criterion is implicit;
- (2) when the criterion is fully met;

The quality criteria (QC) will be listed below 3:

1lightgray

Tab. 3: Quality Criterion.

Criterion	Description of Quality Criterion
CQ1	Is there a clear definition of the objectives of the study?
CQ2	Is there a clear definition of the study's justifications?
CQ3	Is there a clear definition of the study research question?
CQ4	Is there a clear description of the context in which the research was carried out?
CQ5	Does the study provide clear and reliable results?
CQ6	Is the study relevant to research or professional practice?

The criteria presented in Table 3 are defined to evaluate and prove, regarding reliability, quality of the selected studies and why they are required.

After the first step of systematic review, the case study is initiated. This will comprise of the steps of literature review, data collection and data analysis. In the literature review stage, the studies recovered from the systematic review stage will be analyzed according to the main influencing factors of the research, according to the research question. Afterwards, the data collection stage will be carried out in order to be based on a source of evidence, such as physical artifact analysis, in order to analyze tools, instruments or technological devices. In this case, with the objective of collecting information about projects developed for Android relating to quality criteria in code refactoring.

Finally, the data analysis step is performed, which in turn comprises refining the collected data. But also, it is worth mentioning that this step is executed in a parallel way with the collection of data and in a systematic way. In addition, it contributes to conclusions drawn from the data, maintaining a clear chain of evidence. According to [10], in this process the data are codified; the coded material can be combined with comments by the researcher; the researcher uses this material to identify the first set of hypotheses; an iterative approach is performed, so that data collection and analysis are performed in parallel; and a set of generalizations can be formulated from the analysis performed. Finally, a report is generated of the whole process, that is, the result of what was analyzed in the study. The study report is responsible for presenting the results obtained, besides being the main source of information to judge the quality of the study.

IV. THREATS TO VALIDATY

This section presents the set of possible threats, as listed by Wohlin [11] for the study, as well as the actions of controls in order to minimize them.

Listed as threats to internal validity we have: Including only journals articles and events classified by Mining Software Repositories (MSR) in one of the search processes as belonging to software engineering limits the possibility of generalizing the results to other forums in which refactoring technologies are published. Also limiting the selection of articles in other search engines may interfere with a more consistent result. This also introduces the risk of lack of technologies and evaluations published in conference proceedings, reports, workshops, etc. However, since automatic search is performed in a way that does not limit the type of publication classification, other main journals in the area of software engineering may be included in the review, so it is necessary that this threat is limited.

In the article selection process, first, the search

procedure used introduces a threat, since relevant work may be missing for inclusion in the review. Even if the research has lost its work, it should not introduce any systematic bias towards the results. Secondly, the inclusion criterion is applied, the abstracts of the selected studies are read. This introduces a threat because the summary may not necessarily reflect what is actually presented in the articles.

Finally, as threats to external validity, the following are punctuated: Already in the process of extracting the studies, a potential threat to validity is the judgment used to include/exclude a study and extract information from these included studies. With this in mind, in order to limit this threat, a pilot with the classification and inclusion, exclusion and quality criteria can be carried out, and these can be changed before use. In addition, the aspects used for data extraction are derived from the research question. These aspects are subject to variation. To limit this threat, the studies were classified giving the researchers the benefit of the doubt, ie, the studies were classified according to what is mentioned in the articles.

After the systematic review, the case study is started. Thus, during this stage, data collection and analysis phases identified in the previous phase will be performed. Inconsistent collection and analysis of this information may interfere with the extraction of the criteria from the case study phase. This is to generate a report of refinement of the whole process carried out based on the research question.

The threats to validity presented for the protocol of this research are an outline for the study, since it is a proposal, they may change during the application.

V. EXPECTED RESULTS

In this section the expected results related to the development of this research will be presented, in order to discuss the stages of the study carried out based on the steps of the methodology performed in this work.

Based on the studies captured during the review phase, these will be filtered based on the criteria presented above, as presented in Tables 1, 2 and 3. These criteria aim to extract studies with theoretical basis and related to the research question. Therefore, in this phase of systematic review, we hope to extract a summary of the studies related to refactorings in the development process. As well, explore specific quality factors for Android applications.

In addition, it is also expected with the case study stage to present a relationship between the collected studies, with the objective of collecting quality metrics on projects developed for Android. Finally, the data analysis step is performed, which in turn comprises refining the collected data. The accomplishment of this research comprises the analysis of studies in the context of software refactoring. In order to determine which types of transformations, that is, which refactorings are used, trying to relate them to quality and safety factors in the refactoring process.

VI. CONCLUSIONS

We believe that this work can provide an identification of the types of refactoring practiced when developing for the mobile applications area. As well, propose an extraction of the refactoring quality factors related to the development process. We believe that our results will be positive regarding the presentation of an update of the literature review in the context of refactoring, glimpsing comparative factors on the use of refactoring types in android development environments.

In addition, it is expected that the proposal of this work will serve as an input to support the research developed for the context of code refactoring. Therefore, we intend to contribute in a positive way in updating the systematic review, seeking to extract the quality metrics in the process of refactoring android application code.

REFERÊNCIAS

[1] Jacek Ratzinger, Thomas Sigmund, and Harald C. Gall. On the relation of refactorings and software defect prediction. In Proceedings of the 2008 international working conference on Mining software repositories (MSR '08). ACM, New York, NY, USA, 35-38. 2008.

[2] Georgios Gousios, Eirini Kalliamvakou, and Diomidis Spinellis. Measuring developer contribution from software repository data. In Proceedings of the 2008 international working conference on Mining software repositories (MSR '08). ACM, New York, NY, USA, 129-132. 2008.

[3] Gustavo Soares et al. Analyzing Refactorings on Software Repositories. Software Engineering (SBES), 2011 25th Brazilian Symposium on Software Engineering. 2011.

[4] Daniel E. Krutz et al. A dataset of open-source Android applications. In Proceedings of the 12th Working Conference on Mining Software Repositories (MSR '15). IEEE Press, Piscataway, NJ, USA, 522-525. 2015.

[5] Paul W. McBurney and Collin McMillan. Automatic Documentation Generation via Source Code Summarization of Method Context. In Proceedings of the ICPC'14, June 2-3, 2014, Hyderabad, India.

[6] Martin Fowler, Kent Beck (Contributor), John Brant (Contributor), William Opdyke, don Roberts. Refactoring: Improving the Design of Existing Code. Another stupid release 2002.

[7] Marco D'Ambros and Michele Lanza. Software Bugs and Evolution: A Visual Approach to Uncover Their Relationship. Proceedings of the Conference on Software Maintenance and Reengineering (CSMR'06), 2006.

[8] Larry Seltzer for Zero Day. Excess privilege makes companies and data insecure. ZDNet. October 22, 2013. Topic: Security. Link: https://www.zdnet.com/article/excess-privilege-makes-companies-and-data-insecure/
[9] Hulley SB, Cummings SR, Browner WS, Grady DG, Newman TB. Designing Clinical Research. 3rd ed, Philadelphia, PA: Lippincott Williams and Wilkins; 2007.
[10] Robson C. Case Studies for Software Engineers. (2002) Real World Research. Blackwell, (2nd edition).
[11] Wohlin, C., Runeson, P., Hst, M., Ohlsson, M. C., Regnell, B., and Wesslén, A.. 2012. Experimentation in Software Engineering. Springer Publishing Company, Incorporated.
[12] Firouzi, E., Sami, A. Visual Studio Automated Refactoring Tool Should Improve Development Time, but ReSharper Led to More Solution-Build Failures. MAINT 2019, Hangzhou, China.
[13] Derezińska A. (2019) Metrics in Software Development and Evolution with Design Patterns. In: Silhavy R. (eds) Software Engineering and Algorithms in Intelligent Systems. CSOC2018 2018. Advances in Intelligent Systems and Computing, vol 763. Springer, Cham.
[14] Badri, M., Badri, L., Hachemane, O., Ouellet, A. Measuring the effect of clone refactoring on the size of unit test cases in object-oriented software: an empirical study. Innovations in Systems and Software Engineering. Springer-Verlag London Ltd., part of Springer Nature 2019.
[15] Afjehei, S.S., Chen, T-H. P., Tsantalis, N. iPerfDetector:Characterizinganddetecting performanceanti-patternsiniOSapplications. EmpiricalSoftwareEngineering. SpringerScience+BusinessMedia,LLC,partofSpringerNature 2019.
[16] Kaya, M., Conley, S., Othman, Z.S., Varol, A. Effective Software Refactoring Process. 2018 6th International Symposium on Digital Forensic and Security (ISDFS). Antalya, Turkey.
[17] OPDYKE, William F. Refactoring object-oriented frameworks. 1992.
[18] ROBERTS, Donald Bradley; JOHNSON, Ralph. Practical analysis for refactoring. University of Illinois at Urbana-Champaign, 1999.
[19] MINUZZI, Tiago da Silva. Ustory-Refactory: ferramenta de refatoração de requisitos aplicada em cartões user stories (CRC Cards). 2007.

TAYSE VIRGULINO RIBEIRO Possui graduação em Sistemas de Informação pelo Centro Universitário Luterano de Palmas (2018). Mestranda em Ciência da Computação no CIn-UFPE, na Área de Engenharia de Software. Atualmente é Gestora do setor de TIC e Professora na Unibra - Centro Universitário Brasileiro. Tem experiência na área de Ciência da Computação, com ênfase em Engenharia de Software, atuando principalmente nos seguintes temas: processos de desenvolvimento de software, engenharia de software, informática na educação e interação homem computador. Com pouco mais de 6 anos de experiência no mercado, centraliza suas atividades na área de Engenharia de Software e afins (Líder de projetos, Scrum Master e Analista de Sistemas).

MÁRCIO LOPES CORNÉLIO Possui graduação em Ciência da Computação (Bacharelado) pela Universidade Federal da Paraíba (1996), mestrado em Ciências da Computação pela Universidade Federal de Pernambuco (1998) e doutorado em Ciência da Computação pela Universidade Federal de Pernambuco (2004). Atualmente é professor adjunto da Universidade Federal de Pernambuco. Tem experiência na área de Ciência da Computação, com ênfase em Engenharia de Software, atuando principalmente nos seguintes temas: métodos formais, refatoração e transformação de programas.

Publicação: Outubro de 2019

Digital Object Identifier https://doi.org/10.33911/singular-etg.v1i2.59

Efficient Fault-Tolerant Transactions for Distributed Graph Database

RAY NEIHEISER[1], ROLAND SCHMITZ[2], LUCIANA RECH[3], MANFREDO MANFREDINI[4]

1- Federal University of Santa Catarina, Graduate Program at Automation and Systems Engineering, CTC/DAS/PPGEAS, Florianópolis, Brazil (neiheiser.r@posgrad.ufsc.br)
2 - Stuttgart Media University, Faculty of Print and Media, Germany (schmitz@hdm-stuttgart.de)
3 - Federal University of Santa Catarina, Graduate Program at Computer Science CTC/INE/PPGCC, Florianópolis, Brazil (luciana.rech@ufsc.br)
4 - The University of Auckland, School of Architecture and Planning, Auckland, New Zealand. and College of Architecture and Urban Planning, Hunan University, China. (m.manfredini@auckland.ax.nz)
Autor Correspondente: Ray Neiheiser (e-mail: ray.neiheiser@gmail.com).

Through the ongoing trend in graph technologies due to the massive growth of linked data produced by social networks graph databases gained popularity. Replication, a common approach to increase availability in databases, is also used by diverse graph database solutions. Few approaches implementing fault-tolerance in graph databases have been proposed yet. This paper considers deferred update replication using atomic broadcast in order to implement fault-tolerance in distributed graph databases. The main contribution of this paper is a deferred update algorithm adapted to graph databases offering a more scalable and faster solution, showing a performance advantage of over 30% compared to existing approaches.

Graph Databases, Fault Tolerance, Distributed Systems.

I. INTRODUCTION

Current data is usually stored in relational databases with the help of Relational Database Management Systems (RDBMS). RDBMS like Oracle [16], MySQL [12], and Microsoft SQL Server [11] are the most common way of storing data. But, with the large amounts of highly connected data, relational data-bases and also NoSQL do not offer a solution for storing data with plausible query times [23]. However, graph databases, offer an excellent way to store and retrieve highly connected data. Vicknair in [23] compares Neo4j [15], a graph database, with MySQL, a relational database. In their experiments, they discover that graph databases show great performance advantages for queries traversing the graph or searching strings. For instance, while MySQL needs 69.8 seconds to traverse the graph to a depth of 128, counting the number of reachable nodes, Neo4j only needs 18 seconds. Considering Facebook present dimensions of over 2 billion monthly active users and around 54 million pages, suggests that this amount of data and requests cannot be handled by a single computer. Even graph databases cannot cover this amount of data on a single computer or handle immense request numbers. Thus, as well as normal databases, they need to distribute their data over various machines.

Unfortunately, highly distributed systems tend to errors and failures making it necessary to implement fault-tolerance mechanisms for distributed systems, in order to be able to offer high dependability for users. Therefore, a significant amount of work has been done to explore the field of fault-tolerance leading to a high diversity of literature, see e.g., [6] or [9]. Hence, universities and companies have worked intensively in order to develop solutions for this important problem causing the emergence of a wide selection of different approaches in the last decade [3], [4], [13], [14], [21].

Due to the importance and impact of relational

databases several studies have implemented fault-tolerant replication in relational databases as in [2], [7] and [8]. Also, NoSQL databases—which came up in the last years—often provide fault-tolerant services as *Cassandra* or *MongoDB*. Some of the graph databases also offer fault-tolerance yet relying on a central coordinator or implementing state-machine replication, which does not scale well. Our main contribution is an algorithm for fault-tolerance which offers better performance than existing solutions and still scales well.

The next section will offer an overview of the related work highlighting the gap in the literature and proving the importance of the protocol presented in this paper. Section 3 presents the proposed algorithm, the system model and the correctness. Section 4 gives some experimental results on the performance of our algorithm showing its advantages over Neo4j.

II. RELATED WORK

Fault-tolerance, a topic that emerged almost 45 years ago with Randell in [19], brought up a wide range of famous approaches, e.g., Huang in [5] or Rabin in [18]. Thompson in [22], Cai in [2] or Kopetz in [7] who were the first implementing fault-tolerance for relational databases. Since then, the field of fault-tolerance has evolved further. Various solutions have been developed over the last years and due to the NoSQL trend fault-tolerance approaches emerged for different systems, e.g., graph processing as in [24].

Protocols for fault-tolerance differ in various features. Existing protocols implementing crash-fault tolerance rely on $2f + 1$ replicas where f is the number of accepted failures. Protocols may be divided into centralized and decentralized approaches. While centralized approaches offer write-access only over one central replica, decentralized protocols achieve higher throughput through multiple write masters. In order to guarantee consistency in the environment of multiple write masters, most existing protocols offer state-machine replication propagating updates directly to the other replicas. State-machine replication generally comes with the disadvantage of sequential single-threaded execution and therefore suffers a low throughput.

Marandi, in [10], proposes a solution offering parallel execution while still using state-machine replication. Unfortunately, through the immediate propagation of every write and read the distributed data-base easily suffers global locks, which may slow down or block other transactions.

Sciascia, in [20] works with deferred update which propagates updates only on commit time and therefore represents a more scalable and better-performing solution.

Current graph database solutions, as mentioned above, already offer replication schemes. Neo4j—the most popular graph database—offers single-master replication, where the single write master represents a bottleneck with an increasing number of clients. OrientDB [17], another popular solution, offers multiple write masters but relies on state-machine replication and therefore also does not scale well.

Therefore, for our protocol, we will follow the approach of deferred update in order to guarantee a high throughput system avoiding global locks.

III. PROPOSED ALGORITHM

A. SYSTEM MODEL AND DEFINITIONS

This section describes the system model as well as assumptions required for the algorithm.

Let $C = c_1, c_2, ..., c_N$ be a set of client processes and $P = p_1, p_2, ...p_I$ a set of server processes.

Processes don't have access to shared memory and each server process p_I is an exact replica. Only crash failures are accepted in the system, byzantine failures—caused by a bug or intruder—which may affect the consistency of a single replica are assumed to over-strain the system.

A process p_I or c_N that never crashes is correct, otherwise, it is faulty. Every process holds a complete version of the database. Processes use asynchronous uni- or multicast communication primitives [4].

In the case of a correctly working sender and receiver a sent message will eventually reach the receiver. Atomic-multicast through ring-paxos is used to ensure delivery and correct ordering. The used database is a native graph database $D = (d_1, d_2, ..., d_J)$. Data items are nodes $N = (n_1, n_2, ..., n_J)$, relationships $R = (r_1, r_2, ..., r_K)$ and their properties. Relationships are always directed and connect exactly two nodes $n_J \leftarrow n_L$. A relationship r_k connecting node n_J with n_L is therefore a different relationship than the one connecting n_L with n_K. A node n_J may or may not have any relationship. Transactions use locks to apply consistent read and write operations and follow a commit or abort to apply changes. Every node n_J and relationship r_K may be identified by a unique ID or by the unique combination of their properties.

Every transactions contains a read-, write-, delete- and create-set called rs, ws, ds and respec-

tively cs. These sets consist of the unique identifiers describing the nodes and relationships. The consistency model used is serializability [1], meaning that sequential or concurrent executions of transactions lead to exactly the same end results.

B. ALGORITHM

Because of the advantages mentioned above, our algorithm uses deferred update replication with atomic broadcast through *Ring Paxos*. It has a client and server side component, which will now be explained in detail.

Algorithm 1 Client code part 1

```
1:  function BEGIN(t)                          ▷ Start Transaction
2:      t.RS ← ∅                               ▷ Initialize read-set
3:      t.WS ← ∅                               ▷ Initialize write-set
4:      t.CS ← ∅                               ▷ Initialize create-set
5:      t.DS ← ∅                               ▷ Initialize delete-set
6:  end function

7:  function READ(t, tid, i)                   ▷ Read operation
8:      SEND(read, t, i, tid)                  ▷ Send to server
9:      on receive(i,v,rid)                    ▷ Wait for response
10:     if lid = null then                     ▷ If first read
11:         lid ← rid                          ▷ Set Snapshot-id
12:         return v                           ▷ return value
13:     end if
14:     if i ∈ t.CS then
15:         return t.CS(i)                     ▷ return create-set
16:     end if
17:     if i ∈ t.WS then
18:         return t.WS(i)                     ▷ return write-set
19:     end if
20: end function
```

Algorithm 2 Client code part 2

```
1:  function WRITE(op, t, i, v)
            ▷ op = create,update,delete operation
2:      if op = create then                    ▷ On create
3:          t.CS ← (i, v)
4:      else if op = update then               ▷ On update
5:          if i ∈ t.CS then ▷ If create-set contains
6:              t.CS ← (i, v)                  ▷ Update create-set
7:          else
8:              t.WS ← (i, v)                  ▷ Update read-set
9:          end if
10:     else if op = delete then               ▷ On delete
11:         if i ∈ t.WS then ▷ If write-set contains
12:             t.WS.remove(i)
13:             t.DS ← (i, v)                  ▷ Save in delete-set
14:         else if i ∈ t.CS then
15:             t.CS.remove(i)
16:         else
17:             t.DS ← (i, v)                  ▷ Save in delete-set
18:         end if
19:     end if
20: end function

21: function COMMIT(t,lid)
22:     if t.WS = null ∧ t.CS = null ∧ t.DS =
    null then                                  ▷ If read-only transaction
23:         return commit
24:     else
25:         SEND(commit,t,lid)
26:         on receive(outcome)
27:         return outcome
28:     end if
29: end function
```

1) Algorithm in detail

The client code in algorithm 1 shows the part of the algorithm which has to be executed on client side. It starts with the creation of read-, write-, create-, and delete-set. Since the creation of new nodes does not interfere in any way with read or write processes we divide requests into write and create requests. In addition, create-sets also do not interfere with other create-sets. On *BEGIN* a transaction initializes the previously explained sets. In the case of a read operation the client will call the function *READ*—line 6—with its parameters t, lid and i where t is the transaction, lid the local timestamp and i the unique index of the node or relationship.

Since most graph databases handle their indexing automatically, these indices should not be saved externally since they may change during processing. Therefore we identify nodes by their label/type and properties and relationships by starting/end nodes and the relationships properties and relationship type.

On the first $READ$ a transaction t sends to the server, the server returns a timestamp rid. All read requests to the server after the initialization will contain this unique timestamp in order to guarantee that any transaction t_i which has been committed since the start of t won't interfere with it. After the server responded the client's $READ$ request the local create-set and write-set are searched for data matching the unique id i.

The second part of the client code shown in algorithm 2 contains write requests $WRITE$—line 1— which never reach the server. Depending on the operation op the read-, write-, create- or delete-

Algorithm 3 Server code part 1

```
1:  gid = 0                                   ▷ Initiate global snapshot Id
2:  cws < it, ws > ← ∅                        ▷ Committed writes
3:  ptl < it, v > ← ∅                         ▷ Pending transactions
4:  ltl < it, v > ← ∅                         ▷ Local transactions

5:  on receive (read, t, i, sid)
6:  READ(t, i, tid)

7:  function READ(t,i,tid)
8:      if tid = null then                    ▷ On first read
9:          lid ← gid                         ▷ Fill local snapshot Id
10:         ltl ← t                           ▷ Add to local transaction
11:     else
12:         lid ← tid
13:     end if
14:     v = RETRIEVE(i,lid)
15:     SEND(i,v,lid)                         ▷ Send response to client
16: end function

17: on receive (commit, t, tid)
18: COMMIT(t, tid)

19: function COMMIT(t,tid)
20:     ab-cast(commit, t, tid)
21:     outcome = CONFLICTHANDLER(t, tid)
22:     if outcome = commit then
23:         cws ← (tid, t.WS)                 ▷ Store write-set
24:         ptl ← (tid, t)
25:     end if
26:     SEND(outcome)                         ▷ Send outcome to client
27:     for ti ∈ ltl do                       ▷ Check local transactions
28:         if ti.Rs ⋂ ptl[0] then
29:             ti.abort                      ▷ If conflict, abort
30:         else
31:             Executet[0]
32:             gid = tid
33:         end if
34:     end for
35: end function

36: function CONFLICT-HANDLER(t,tid)
37:     for ti ∈ ptl ← it > tid do
38:         if !ti.WS[i] ⋂ t.RS then          ▷ On conflict
39:             return commit
40:         else
41:             return abort
42:         end if
43:     end for
44: end function
```

set are updated with value v and id i. As in the case of the $READ$ request t represents the transaction object. On *COMMIT*—line 21—we analyze the write-, create- and delete-set. If all of these sets are empty, transaction t is a read-only transaction and can be committed immediately. Else, the client sends his commit request to the server. The client will then wait for a response from the server indicating whether the transaction has been committed or aborted.

The client requests are directed to the server—as described in algorithm 3. The server initially has a global snapshot id gid which increases with each committed transactions, the set of committed write-sets cws, a list of pending transactions ptl and a list of local transactions ltl.

When the server receives a read request from the client—line 5—it calls function *READ* with the transaction t, the index i and the transactions snapshot id tid as parameters. In $READ$—line 7—it initializes snapshot id sid if it is the first read request of the transaction, retrieves the requested data from the database and responds by sending the index i, the value v and the local snapshot id lid to the client.

On commit, the server will execute the function *COMMIT*—line 20— with the transaction t and its snapshot id tid as parameters. $COMMIT$ will distribute the $COMMIT$ request to all other replicas. and start the *ConflictHandler*. The *ConflictHandler* in line 36 calculates if there are any conflicts with previously committed transactions and will, depending on the result, return a commit or an abort. If it returns commit the write-set will be stored in the committed write-sets and the transaction will be added to the pending transaction list ptl. Following, a search for conflicts in the locally running transactions to abort these if necessary. After a transaction has been committed successfully —line 31— the server will increase his global snapshot id.

C. CORRECTNESS

Ring Paxos, the protocol we deployed for atomic broadcast, guarantees total ordered transactions. Therefore, all transactions are executed at all replicas in the same order which guarantees serializability [1].

Old messages are stored so that recovering servers are able to replay lost transactions to regain consistency after a crash. In Ring Paxos, at least three replicas must be alive in order to guarantee an agreement between the servers. So, $2f + 1$ replicas are required in order to tolerate f failures. Never-

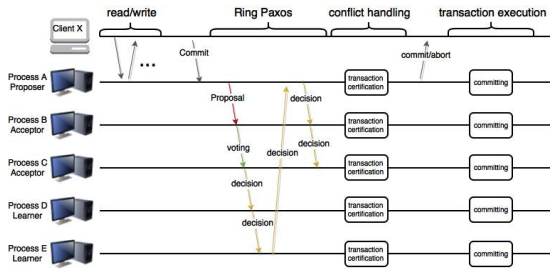


Fig. 1: Deferred Update replication - Ring Paxos

theless, at least three replicas have to keep working correctly in order to maintain the services availability. On transaction commit at the local replica, it will be distributed to the other replicas and if there are no conflicts detected it will be committed by all correct replicas or none of them.

On transaction abort, no changes will be written to the database. All transactions run completely isolated, which means that during the run time of a transaction T_1 the results of any concurrently running transaction T_i will not interfere in any way with T_1. All concurrently executed transactions, therefore, have the same result, being executed concurrently or sequentially. This is guaranteed through storing time stamps in the form of Ring Paxos instance numbers into the properties of nodes and relationships, letting local transactions access only the changes made by the transactions which have been committed previous to the first read request to the database.

Interference between local transactions in the executing stage and global transactions in the committing stage are automatically detected and the local transaction will be aborted by the server. The local transaction initiates on the first read request which returns the mentioned timestamp.

After the replicas have decided on a commit, transactions in the case of Neo4j cannot be executed concurrently due to Neo4j's automatic indexing mechanism. Neo4j creates a unique index for each new node and relationship. The programmer cannot access these mechanisms and therefore, in order to guarantee a consistent state, create requests have to be executed sequentially at every local replica.

Figure 1 shows the sequence of events initiated by our algorithm and Ring Paxos. In the first phase, the client executes various read and write requests on the server until it sends the commit request. On commit, the termination protocol starts and Ring Paxos executes atomic broadcast. The commit runs through the ring of servers until the servers agree on an order. Once the message arrives, all server proceed in the same order: the conflict handler starts and returns—in the case of a conflict— abort, else commit. The server then sends the outcome to the client and finally writes the changes to the database.

IV. IMPLEMENTATION AND TESTING

All tests ran on 4 Linux 64 bit machines with Intel-core i7-3770k quad-cores, hyper threading and 4 gigabyte RAM. The algorithm has been implemented in Java in the programming environment of IntelliJ using the java 1.8.0_31 jdk. Since Neo4j is a popular graph database, we chose Neo4j high availability mode in order to compare the performance of our algorithm. Neo4j only offers writes through a single master. OrientDB offers multiple write masters but relies on a key-value store and is therefore not a native graph database which doesn't make it suitable for a performance comparison. For testing purposes, the same client and server socket is used to avoid results manipulated by socket related connection delays. We initiate with an empty database and then, in order to test the performance, a test function has been designed which sends 32 transactions to the server creating a random number of nodes and relationships between them and then updates the nodes and relationships. While running the function conflicts between the transactions may happen which have to be resolved by each algorithm. After the test program has finished, the databases will be compared in order to search for inconsistencies. Two sets of experiments have been conducted in order to investigate the differences between both solutions.

A. VARYING THE NUMBER OF MACHINES

The first set of experiments runs four server processes on two, three and four physical machines. On each of these three setups, ten replications have been performed in order to achieve statistically relevant results. Four clients then access these four server processes and print out the required time as soon as they are finished.

Observing figure 2, which compares Neo4j one the left and our developed algorithm on the right our solution shows a significantly better performance than the High availability mode Neo4j offers. Neo4j even shows a slightly worse performance if distributed on more servers, while our algorithm runs even faster.

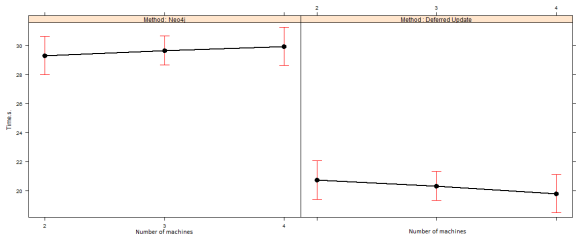


Fig. 2: Four server processes on two to four physical machines

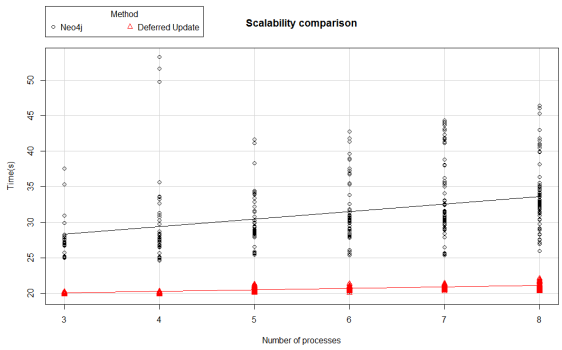


Fig. 3: Three to eight server processes on four physical machines

B. VARYING THE NUMBER OF PROCESSES

The second set of experiments runs three to eight server processes on our four physical machines. Again, three to eight client processes access these server processes and each setup has been executed ten times.

In the second experiment—whose results are displayed in figure 3—our algorithm is represented by the lower line, while Neo4j high availability mode is represented by the upper one. As in the first experiment our algorithm performs significantly better than the other solution. Observing the slope of the lines shows that our algorithm exhibits a much better scalability with respect to a higher number of replicas.

V. CONCLUSION

The deferred update algorithm for distributed graph databases presented in this paper shows great performance advantages compared to the solution Neo4j–the market leader–offers. As future research, it is possible to investigate clearer performance advantages through access to Neo4j's indexing mechanism or by conducting experiments on a bigger number of machines. Nevertheless, deferred update also has disadvantages. For the deferred update algorithm to work each replica requires the whole database and long running-transactions may easily suffer starvation. However, nowadays disk space is not a problem in that it is easy to store terabytes of data at a local replica. Also, in our opinion, long-running transactions easily lead to global locks and should, therefore, be avoided. Graph processing engines, which are build for offline graph analysis, should be used. Our solution, therefore, may offer a great opportunity for other graph database providers to implement scalable fault-tolerance into their systems helping graph databases to take more influence dealing with the massive growth of highly connected data. In future research, we would like to add a system for avoiding transaction starvation.

Acknowledgments: This paper is dedicated to the memory of our dear colleague Prof. Lau Cheuk Lung who motivated us to pursue this work.

REFERÊNCIAS

[1] P. A. Bernstein, V. Hadzilacos, and N. Goodman. Concurrency Control and Recovery in Database Systems. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1987.
[2] J. Cai, L. A. Hemachandra, and J. Vyskoč. Promises and fault-tolerant database access. Technical report, ., Rochester, NY, USA, 1993.
[3] R. Ekwall and A. Schiper. A fault-tolerant token-based atomic broadcast algorithm. Dependable and Secure Computing, IEEE Transactions on, 8(5):625–639, Sept 2011.
[4] V. Hadzilacos and S. Toueg. A modular approach to fault-tolerant broadcasts and related problems. Technical report, 1994.
[5] K.-H. Huang and J. Abraham. Algorithm-based fault tolerance for matrix operations. Computers, IEEE Transactions on, C-33(6):518–528, June 1984.
[6] P. Jalote. Fault Tolerance in Distributed Systems. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1994.
[7] H. Kopetz, A. Damm, C. Koza, M. Mulazzani, W. Schwabl, and C. Senft. Distributed fault-tolerant real-time systems: the mars approach. Micro, IEEE, 9(1):25–40, Feb 1989.
[8] A. F. Luiz, L. C. Lung, and M. Correia. Byzantine fault-tolerant transaction processing for replicated databases. In 10th IEEE NCA, pages 83–90, Aug 2011.
[9] M. R. Lyu. Software Fault Tolerance. John Wiley & Sons, Inc., New York, NY, USA, 1995.
[10] P. Marandi, C. Bezerra, and F. Pedone. Rethinking state-machine replication for parallelism. In 34th IEEE ICDCS, pages 368–377, June 2014.
[11] Microsoft. Microsoft sql server, 2019.
[12] MySQL. Mysql enterprise edition, 2019.
[13] R. Neiheiser, M. Bravo, L. Rodrigues, and L. Rech. Policy-based adaptation of a byzantine fault tolerant distributed graph database. In SRDS. IEEE, 2018.
[14] R. Neiheiser, D. Presser, L. Rech, M. Bravo, L. Rodrigues, and M. Correia. Fireplug: Flexible and robust n-version geo-replication of graph databases. In ICOIN, pages 110–115, Jan 2018.
[15] Neo4j. Neo4j graph platform, 2019.
[16] Oracle. Main site, 2019.
[17] OrientDB. Main site, 2019.
[18] M. O. Rabin. Efficient dispersal of information for security, load balancing, and fault tolerance. J. ACM, 36(2):335–348, Apr. 1989.

[19] B. Randell. System structure for software fault tolerance. SIGPLAN Not., 10(6):437–449, Apr. 1975.
[20] D. Sciascia, F. Pedone, and F. Junqueira. Scalable deferred update replication. In 42nd Annual IEEE/IFIP DSN, pages 1–12, June 2012.
[21] P. Sutra and M. Shapiro. Fault-tolerant partial replication in large-scale database systems. In Euro-Par 2008 – Parallel Processing, volume 5168 of LNCS, pages 404–413. Springer Berlin Heidelberg, 2008.
[22] G. Thompson and et al. Fault-tolerant distributed database system and method for the management of correctable subtransaction faults by the global transaction source node, Sept. 21 1993. US Patent 5,247,664.
[23] C. Vicknair and et al. A comparison of a graph database and a relational database: A data provenance perspective. In 48th Annual Southeast Regional Conference, ACM SE '10, pages 42:1–42:6, New York, NY, USA, 2010.
[24] P. Wang, K. Zhang, R. Chen, H. Chen, and H. Guan. Replication-based fault-tolerance for large-scale graph processing. In 44th Annual IEEE/IFIP DSN, pages 562–573, June 2014.

RAY NEIHEISER has earned his master's degree in Computer Science from the Federal University of Santa Catarina (UFSC) and he is now a PhD student at the same university. His area of interest is the construction of reliable distributed systems. In the context of this, he specifically focuses on byzantine fault tolerant consensus and distributed ledger technology.

DR. ROLAND SCHMITZ is a professor of Internet Security at Stuttgart Media University since 2001. Before that, he was a senior researcher at the research center of Deutsche Telekom in Darmstadt, Germany, where he worked on mobile communications security and digital signature standardization. Roland Schmitz has authored or co-authored more than 70 journal papers, conference papers and books. Roland Schmitz holds a diploma degree and a Ph.D. in mathematics, both from the technical university of Braunschweig, Germany. He is a member of IEEE.

DR. LUCIANA RECH is Associate Professor at the Informatics and Statistics Department (INE) of the Federal University of Santa Catarina (UFSC) and a member of the Distributed Systems Research Laboratory (LAPESD) since 2009. Luciana Rech holds a diploma degree and Master´s degree in Computer Science, and Ph.D. in Electrical Engineering. Her core area of expertise is Distributed Systems, Intelligent Systems, Real Time Systems and Applied Informatics.

DR. MANFREDO MANFREDINI is Director and Senior Lecturer at the School of Architecture and Planning of the University of Auckland and Honorary Professor at the College of Architecture and Urban Planning, Hunan University. His core area of expertise is at the intersection of comparative urbanism and architectural design. He taught design and theory courses at leading global schools (e.g. Tsinghua University Beijing and Milan Technical University) and was invited as keynote speaker in international conferences (e.g. 9th China Housing Congress and 7th Arte-Polis Conference). His research leadership has been recognised by publications (100+ papers), invited presentations in high impact journals, leading global universities (e.g. University of Stuttgart and Chinese University of Hong Kong), major international events (e.g., UN-Habitat 3, Rome Biennale of Public Space and Bi-City Biennale of Architecture and Urbanism in Shenzhen and Hong Kong) and important and awards (e.g. first prize at the Biennale di Venezia). Over the years he collaborated with prominent educators and designers, such as Colin Fournier (Archigram and The Bartlett, UCL), Andrea Branzi (Archizoom and Milan Technical University) and Ulisse Staccioli (Brera Fine Arts College, Milan).

Publicação: Outubro de 2019

Digital Object Identifier https://doi.org/10.33911/singular-etg.v1i2.60

Fertirrigação no cultivo de capim e a diversidade microbiana do solo do Cerrado antes e após a produção de biomassa vegetal

DR. JOSÉ GERALDO DELVAUX SILVA[1],
DR. JOSÉ MARIA RODRIGUES DA LUZ[2],
DRA. SÔNIA SALGUEIRO MACHADO[3],
JOSÉ EXPEDITO C. DA SILVA[4].

1- Engenheiro agrônomo. Servidor público do Estado do Tocantins.
2- Professor Visitante do Instituto de Ciência Farmacêuticas. Laboratório de Enzimologia Universidade Federal de Alagoas (UFAL).
3- Professor do Instituto de Química. Laboratório de Enzimologia. UFAL.
4- Professor da Universidade Federal do Tocantins.
(e-mail:delvaux2013@hotmail.com, josemarodrigues@yahoo.com.br, salgueiro.sonia355@gmail.com, jecs@mail.uft.edu.br)
Autor Correspondente: José Geraldo Delvaux Silva (e-mail: josemarodrigues@yahoo.com.br).

RESUMO - A fertirrigação é uma técnica de aplicação de nutrientes para plantas via água de irrigação. As águas residuárias domésticas (ARD) têm elevadas concentrações de nitrogênio (N), fósforo (P) e potássio (K), com isso surge a alternativa do reuso dessas águas como fonte de nutrientes no cultivo agrícola, reduzindo os impactos ambientes gerado por esse resíduo e diminuindo os custos na aquisição de adubos químicos na implantação das culturas. Contudo, as principais limitações do uso de ARD são a presença de sódio, e microrganismos coliformes fecais. Entretanto, estudos realizados na Universidade Federal de Tocantins têm demonstrado que o uso de ARD no solo do Cerrado para cultivo de capim tem baixo potencial de sanilização e sodificação do solo, sem alterações significativas nas propriedades físicas e químicas do solo e a biomassa vegetal não apresenta crescimento desses microrganismos. Assim, o objetivo desse estudo foi avaliar as alterações na fertilidade e na diversidade microbiana (bactérias fixadoras de nitrogênio e fungos micorrízicos) do solo do Cerrado após aplicação de ARD no cultivo de capim Brachiaria brizantha cv Marandu. Os manejos de irrigação utilizados continha 0, 20, 40 e 60 % (m/v) de fertilizantes NPK oriundos de ARD que foi aplicado após o crescimento da plântula. As composições químicas e a diversidade microbiana foram determinadas nas amostras do solo. Nas amostras da planta determinou-se a massa seca e o potencial nutricional do capim. A adição de ARD alterou a abundância bactérias e fungos, mas a diversidade não teve alterações. Esse resultado pode ser devido à maior disponibilidade de NPK no solo. Não foram observadas alterações na composição nutricional do capim após a fertirrigação. Portanto, a ARD tem potencial para ser utilizado na fertirrigação de capim em solo do Cerrado.

PALAVRAS-CHAVE - esgoto doméstico, capim, diversidade microbiana, fertilizantes, ração.

I. INTRODUÇÃO

A fertirrigação com águas residuárias proveniente do esgoto doméstico pode ser uma boa alternativa para o plantio de capim Brachiaria brizantha em solo do Cerrado ([15], [18], [16], [17], [19]). Esse processo pode diminuir o uso de água de melhor qualidade na irrigação, a aplicação de fertilizantes sintéticos e os desmatamentos para plantio de pastagens.

A baixa fertilidade dos solos do Cerrado se caracteriza por possuírem elevados teores de alumínio e manganês trocáveis, baixa capacidade de reter fertilizantes à base de potássio e deficiência de micronutrientes [12]. Além disso, os teores de matéria

orgânica na maioria dos solos dos Cerrados são considerados baixos [1]. Assim, os nutrientes presentes nas águas residuárias podem ser disponibilizados para o crescimento vegetal nesse tipo de solo ([16] e [19].

As técnicas de manejo e fertilização contornam os principais problemas dos solos do Cerrado que tem contribuído para a ocupação de grandes extensões pela agricultura moderna [19].

No Brasil, o cultivo de Brachiaria brizantha cobre 30 milhões de ha, o que equivale a cerca de 50% das gramíneas cultivadas na região do Cerrado (ABRASEM, 2012). A expansão de áreas pastagens cultivadas com Brachiaria sp no Brasil tem se verificado em proporções jamais igualadas por outras forrageiras, em qualquer outro país de clima tropical ([3] e [21]). Além disso, os grandes interesses dos pecuaristas por esta espécie incluem o fato de a mesma ser uma planta de alta produção de massa seca, ter boa adaptabilidade aos solos do Cerrado, responder bem à adubação fosfatada, ter facilidade para estabelecimento e persistência, apresentar alto valor nutritivo para ração e poucos problemas de doenças e mostrar um bom crescimento durante a maior parte do ano [21].

A microbiota do solo é a principal responsável pela decomposição dos resíduos orgânicos, pela ciclagem dos nutrientes e pelo fluxo de energia dentro do solo ([24], [6]). Assim, seria muito importante analisar o efeito do uso de águas residuárias para fertirrigação sobre os microrganismos que contribuem para a fertilidade do solo. Os fungos micorrízicos arbusculares (FMAs) e as bactérias fixadoras de nitrogênio (BFN) podem ser um bom parâmetro para investigar essas possíveis alterações. Além disso, esses microrganismos podem contribuir para evitar o acúmulo de nutrientes, incluindo o fósforo e nitrogênio, e de metais pesados oriundos da água residuária doméstica no solo.

A utilização águas residuárias para fertirrigação de B. brizantha parece ser uma boa alternativa para produção de alimento animal em solos do Cerrado que apresentam deficiência hídrica e nutricional. Entretanto, estudos sobre as alterações dos atributos físico-químicos e microbianos do solo é necessário. Assim, o objetivo desse estudo foi avaliar as alterações na fertilidade e na diversidade microbiana (bactérias fixadoras de nitrogênio e fungos micorrízicos) do solo do cerrado após aplicação de ARD e cultivo de capim B. brizantha cv Marandu.

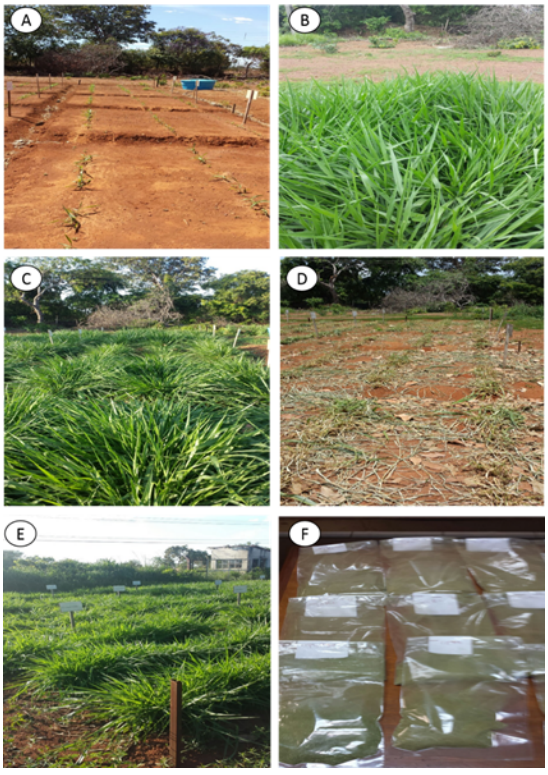


Fig. 1: Ilustrações da evolução do experimento de fertirigação do capim Brachiaria brizantha cv. Marandu em solo do Cerrado com água residuária doméstica. A- Capim Brachiaria Brizantha cv Marandu aos 57 dias. B - Capim em novembro antes do 1o. corte. C - Capim na véspera do 2o. corte. D- Um dia após o 20 corte. E – Capim na véspera do 3o. corte. F –Amostras secas para análises químicas-bromatológica [18].

II. MATERIAL E MÉTODOS

A pesquisa foi conduzida na Área Experimental do Laboratório de Recursos Hídricos do Centro Universitário Luterano de Palmas (CEULP/ULBRA). Em uma área experimental de 180 m2 cultivou a gramínea Brachiaria brizantha cv Marandu, com a utilização do efluente sanitário do CEULP/ULBRA (Figura 1A). A composição do manejo de irrigação foi baseada na adição de 0, 20, 40 e 60% (m/v) de NPK contido na ARD da CEULP/ULBRA. Essas composições estão de acordo com os estudos de Silva et al, [19] realizados na Universidade Federal do Tocantins. O tratamento controle (T0) não continha NPK do ARD. Os tratamentos T1, T2, T3 e T4 continha doses acrescente de NPK do ARD.

As metodologias de plantio, irrigação e as análises das amostras do solo e do capim foram realizadas conforme os estudos de Silva [16]e Silva

et al [18].

As sementes foram aplicadas a lanço e utilização de um rastelo para posterior enterrio das mesmas no solo. A aplicação da água residuária foi feita através de um regador de 10 litros, sobre as folhas quando o capim apresentava cerca de 20 cm de comprimento (Figura 1B).

A coleta de amostras do solo (20 g) ocorreu antes do plantio das sementes e após o corte do o capim para análise da diversidade microbiana. A contagem de microrganismos viáveis foi realizada de acordo com método de Sabino [14]. Os resultados das contagens microbianas foram expressos em logarítmo da unidade formadoras de colônia (UFC) por grama de solo.

A diversidade de bactérias fixadoras de nitrogênio (BFN) e fungos micorrízicos arbusculares (FMAs) foram analisadas pela técnica de PCR-DGGE após amplificações dos genes 16S rDNA, nif H e 18S rDNA e os AM1 [8] e NS31 [20].

Os cortes ocorreram após 140 dias da semeadura das sementes do capim com o auxílio de uma tesoura de poda (Figura 1CDE). Nesse estudo, foram realizados três cortes. Após isso, pesou-se a massa verde produzida. A determinação da composição química foi realizada após as amostras serem secadas em estufas a 60 oC até peso constante (Figura 1F) de acordo com as técnicas descritas pela Embrapa [4].

O experimento foi conduzido no delineamento inteiramente casualizado (DIC) com quatro manejos de irrigação e quatro repetições para cada manejo e

III. RESULTADOS E DISCUSSÃO

Os teores de sódio das ARD foram menores que a concentração limite para contaminações, salinização e ou sodificação do solo e do lençol freático ([5], [7]). Assim, a possibilidade de acumulação desses minerais efluente doméstico do CEULP/ULBRA é mínima. O pH da água residuária foi básico. Esse valor contribui para a correção da acidez do solo que favorável à germinação da semente e crescimento da plântula. Alem disso, com exceção do índice de saturação de bases, pH e areia todas as outras variáveis físico-químicas analisadas nas amostras do solo reduziram teores em função da profundidade (Tabela 1) .

Após a fertirrigação observou-se um aumento significativo na saturação de base e uma redução significativa na saturação de alumínio ($p< 0,05$). Segundo Koura et al. [9], alterações nas propriedades físicas e químicas do solo dependem do tempo de aplicação, do tipo de cultura utilizada e das características do

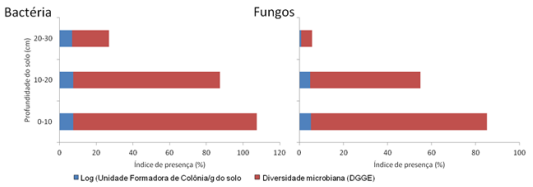


Fig. 2: Contagem de células viáveis e diversidade microbina por eletroforese em gel de gradiente desnaturante (DGGE) antes a fertirrigação em diferentes profundidades do solo do cerrado de Tocantins.

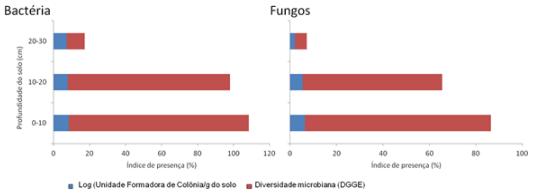


Fig. 3: Contagem de células viáveis e diversidade microbina por eletroforese em gel de gradiente desnaturante (DGGE) após a fertirrigação em diferentes profundidades do solo do cerrado de Tocantins.

solo e do efluente. Além disso, após a fertirrigação o solo permaneceu ácido nas profundidades estudadas. Duarte et al [4]) também não constataram diferença significativa do pH do solo antes e após a aplicação de água residuária no solo.

A aplicação da água residuária também aumentou a disponibilidade de matéria orgânica no solo (Tabela 1).

A contagem de células viáveis e a diversidade microbiana foram alteradas em função da profundidade do solo (Figuras 2 e 3). Esse resultado por ser devido à disponibilidade nutriente e água que diminui nas profundidades do solo. Além disso, quando se compara os resultados das Figuras 2 e 3, a água residuária alterou a abundância microbiana, mas não a diversidade de BFNs e FMAs no solo. Nesse caso, esse resultado evidencia o potencial de uso dessas águas na fertirrigação em solo do cerrado.

Independente da profundidade, a comunidade de bactérias foi maior que à de fungos filamentosos antes e após a fertirrigação (Figuras 2 e 3). O predomínio da comunidade bacteriana em relação à de fungos também foi observada no solo sob vegetação nativa localizado na região Sul do Brasil [13]).

A diversidade de gene nif H foi maior nas profundidades de 0 a 10 cm e 10-20 cm (Figuras 2 e 3).

Tab. 1: Sat.base = Saturação de base, Sat. Al = Saturação de Alumínio. * Os valores desta tabela representam a média de 4 repetições. Esses valores foram comparados pela análise de variância (Anova) seguida do teste de Tukey a 5% probabilidade (veja os anexos). Os resultados dessas análises estatísticas estão apresentados no texto indicado, pelo nível significativo p < 0,05. Fonte: autores

Variáveis		Profundidade do solo (cm)								
		0-10			10 – 20			20-30		
pH (CaCl2)		5	±	0,87	4,8	±	0,78	4,625	±	0,5
Argila	%	27	±	0	35,75	±	3,95	42	±	4,24
Areia	%	50	±	3,46	40,75	±	5,62	35	±	2
Limo	%	23	±	3,46	23,5	±	1,73	23	±	4,69
Ca	cmolc/dm3	2,7	±	2,35	1,65	±	1,65	0,975	±	1,02
Mg	cmolc/dm3	0,975	±	1,03	0,5	±	0,38	0,45	±	0,38
Al	cmolc/dm3	0,1	±	0,14	0,05	±	0,06	0,125	±	0,1
H+Al	cmolc/dm3	3,35	±	2,06	3,6	±	1,88	3,05	±	1,66
K	cmolc/dm3	0,042	±	0,01	0,069	±	0,07	0,04	±	0,01
CTC	cmolc/dm3	7,05	±	1,77	5,8	±	1,43	4,5	±	1,49
Matéria Orgânica	g/kg	31	±	3,27	28	±	3,83	26	±	3,83
Sat. Base	%	49,05	±	36,25	37,05	±	28,84	32,05	±	24,4
Sat. Al	%	8,275	±	12,44	5,975	±	6,9	14	±	11,65
Na	mg/dm³	1,25	±	0,5	1,25	±	0,5	1,5	±	0,58
Zn	mg/dm³	1,2	±	0,41	1,15	±	0,1	0,925	±	0,34
B	mg/dm³	0,225	±	0,05	0,25	±	0,06	0,175	±	0,05
Cu	mg/dm³	0,15	±	0,06	0,125	±	0,05	0,1	±	0
Fe	mg/dm³	36	±	3,74	34	±	8,12	24,75	±	3,59
Mn	mg/dm³	20,25	±	14,03	7	±	3,46	10	±	12,08
K	mg/dm³	16,5	±	5,26	27	±	28,77	15,5	±	4,73
P (Melich I)	mg/dm³	2,75	±	0,96	2,25	±	1,26	1,5	±	0,58

Esse resultado pode ser explicado devido à maior quantidade de células de microrganismos na região da rizosfera [2].

A presença de NPK das águas residuárias pode ter contribuído para germinação de esporos e crescimento de células microbianas no solo (Figura 3). Esse resultado pode ser observado na quantidade maior de células viáveis após a aplicação da água residuaria. Os microrganismos do solo auxiliam a planta na absorção de água e nutrientes, principalmente o fósforo do solo ([22], [23]). Segundo Machado et al. [10], o teor de água e nutriente aumenta a taxa de germinação de esporos microbianos o que confirma o potencial de uso de água residuária para irrigação de culturas agrícolas, com efeito positivo sobre os microrganismos do solo (Figuras 2 e 3).

Neste estudo, a produtividade de massa seca do capim B. brizantha cv Marandu foi diretamente proporcional à concentração de NPK proveniente da água residuária. Silva et al. [17] também observaram que as maiores produções de massa seca foliar por hectare (ha) foram observadas nos manejos de irrigação que receberam água residuária. Esses autores sugeriram que NPK comercial está menos disponível para planta que o NPK da água residuária. Além disso, o efluente doméstico não alterou a composição nutricional do capim Marandu (Figura 4).

As análises de componentes principais das

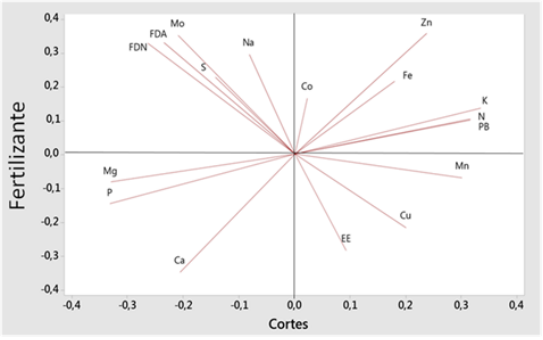


Fig. 4: Análise de componente principal da composição bromatológica do capim Brachiaria brizantha cv Marandu em função da adição de águas residurias (fertilizante) no solo do cerrado [18].

amostras do capim mostraram que a fibra de detergente neutro (FDN), a fibra de detergente ácida (FDA) e as concentrações de molibdênio, enxofre, sódio, magnésio, fósforo e cálcio tiveram um efeito negativo em função dos cortes (Figura 4). Assim, essas variáveis o primeiro corte teve melhor resultado que pode ser devido à redução dos teores de NPK nos intervalos dos cortes. Entretanto, o manganês, o cobre, o extrato etéreo, o zinco, o ferro, o potássio, o nitrogênio e a proteína bruta tem efeito contrário (Figura 4). Apenas os teores de cobalto não têm nenhum efeito em relação ao tratamento e os cortes que parece ser característica da fisiológ-

ica da planta. Portanto, essas variações negativas, neutras e positivas na composição bromatológica no capim Marundu mostram que a concentração de NPK adicionada no solo pode contribuir para uma melhor composição nutricional do capim.

IV. CONCLUSÕES

A fertirrigação mostrou ser uma alternativa viável para utillização de águas residuárias domésticas na agricultura no solo do Cerrado;

O uso de fertilizantes oriundos da água residuária doméstica contribuiu positivamente com a abundância de bactérias fixadoras de nitrogênio e fungos microrrízicos arbusculares no solo;

A fertirrigação aumentou a produtividade de massa seca foliar indicando que o NPK da água residuária é mais disponível para a planta que o NPK comercial;

REFERÊNCIAS

[1] ADÁMOLI J, MACEDO J, AZEVEDO LG, NETTO JM. Characterization of Cerrado Region. In. Goedebert, W. J. (Ed) Cerrado soil: Technologies and Management Strategies, Nobel, EMBRAPA, São Paulo. pp. 33-74, 1985.

[2] DA SILVA MRSS. Diversidade de comunidades bacterianas de solo de Cerrado em resposta a diferentes alterações dos ecossistemas. Tese, Universidade de Brasília, 140p, 2012.

[3] DIAS-FILHO MB. Diagnóstico das Pastagens no Brasil. Documento 402. Empresa Brasileira de Pesquisa Agropecuária (Embrapa), 2014.

[4] DUARTE AS, AIROLDI RPS, FOLEGATTI MV, BOTREL TA, SOARES TM. Effects of application of treated wastewater in soil: pH, organic matter, phosphorus and potassium. Revista Brasileira de Engenharia Agricola e Ambiental, 12:3, 2008.

[5] EMPRESA BRASILEIRA DE PESQUISA AGROPECUÁRIA (Embrapa). Manual de métodos de análises do solo/Centro Nacional de Pesquisa de Solos. 2a edição revisada e atualizada, Rio de Janeiro, 212p., ISBN-85-85864-03-6, 1999.

[6] FONSECA SPP, SOARES AA, MATOS AT, PEREIRA OG. Nutritional value and fecal contamination of coastcross grass grown in a unit of wastewater overland flow treatment. Revista Brasileira de Engenharia Agricola e Ambiental, 21:293-301, 2001.

[7] GAMA-RODRIGUES EF, BARROS NF, GAMA-RODRIGUES AC, SANTOS GA. Carbon, nitrogen and activity of microbial biomass in soil under eucalypt plantations. Revista Brasileira de Ciência do Solo, 29-893-901, 2005.

[8] GLOAGUEN TV, GONÇALVES RAB, FORTI MC, LUCAS Y, MONTES CR. Irrigation with domestic wastewater: a multivariate analysis of main soil changes. Revista Brasileira de Ciência do Solo, 34:1427-1434, 2010.

[9] HELGASON T, FITTER AH, YOUNG JPW. Ploughing up the wood-wide web? Nature, 394-431, 1998.

[10] KOURAA A, FETHI F, LAHLOU A, OUAZZANII N. Reuse of urban wastewater by combined stabilization pond system en Benslimane (Marocco). Urban Water, 4: 373-378, 2002.

[11] MACHADO CTT, PEREIRA CD, LOPES V. Fungos micorrízicos arbusculares: pesquisa e desenvolvimento para a agricultura. In: FALEIRO, FG, DE ANDRADE, SRM. Biotecnologia: estado da arte e aplicações na agropecuária. Embrapa Cerrados, 730 p, 2011.

[12] MALAVOLTA E, KLIEMANN HJ. Nutritional disorders in the Cerrado. Piracicaba: Associação Brasileira para Pesquisa da Potassa e do Fosfato, 136p, 1985.

[13] RECH M, PANSERA MR, SARTORI VC, RIBEIRO RTS. Microbiota do solo em vinhedos agroecológico e convencional e sob vegetação nativa em Caxias do Sul, RS. Revista Brasileira de Agroecologia, 8: 3, 2013.

[14] SABINO DCC. Interação planta-bactéria diazotrófica na cultura de arroz. Tese de Doutorado. Universidade Federal Rural do Ruio de Janeiro. Seropédica, RJ, 54p, 2007.

[15] SILVA JGD. Chemical composition and productivity of Mombaça grass grown in different effluent blades of primary sewage treatment. Dissertação. Universidade Federal de Viçosa, 67p, 2010.

[16] SILVA JGD, MATOS AT, BORGES AC, PREVIERO CA. Chemical bromatological composition and productivity of Mombaça grass (Panicum maximum cv. mombaca) submitted to different primary wastewater treatments of sanitary sewage. Revista Ceres, 59:606-613, 2012.

[17] SILVA, JGD, CARVALHO, JJ; DA LUZ, JMR; SILVA, JEC. Fertigation with domestic wastewater: Uses and implications. African Journal of Biotechnology, 15: 807-815, 2016.

[18] SILVA, JGD. Biotecnologia do uso de água residuária doméstica em solo do cerrado no cultivo do capim Brachiaria brizantha cv marandu.Universidade Federal do Tocantins, 97p, 2017.

[19] SILVA, JGD; LUZ, JMR, HENRIQUE, J, CARVALHO, JJ; SILVA, JEC. Domestic Wastewater for Forage Cultivation in Cerrado Soil. Journal of Agricultural Science, 10: 248-254., 2018.

[20] SIMON L, LALONDE M, BRUNS TD. Specific amplification of 18S fungal ribosomal genes from vesicular-arbuscular endomycorrhizal fungi colonizing roots. Applied and Environmental Microbiology, 58: 291-295, 1992.

[21] SOARES FILHO CV, RODRIGUES LRA, PERRI SHV. Produção e valor nutritivo de dez gramíneas forrageiras na região Noroeste do Estado de São Paulo. Acta Scientiarum, 24-5-1377-1384, 2002.

[22] SOUZA JAA, BATISTA RO, RAMOS MM, SOARES AA. Microbiological contamination of soil by sewage. Acta Scientiarum Technology, 33-5-8, 2011.

[23] THONGTHA S, TEAMKAO P, BOONAPATCHAROEN N, TRIPETCHKUL S,TECHKARNJARARUK S, THIRAVETYAN P. Phosphorus removal from domestic wastewater by Nelumbo nucifera Gaertn and Cyperusal ternifolius L. Jounal of Environmental Management, 137-54-60, 2014.

[24] VITOUSEK PM, HÄTTENSCHWILER S, OLANDER L, ALLISON S. Nitrogen and Nature. Ambio: A Journal of the Human Environment, 31-97-101, 2002.

DR. JOSÉ GERALDO DELVAUX SILVA Possui graduação em Agronomia pela Universidade Federal de Viçosa (1993), graduação em Química pela Universidade do Tocantins (2005), mestrado em Engenharia Agrícola pela Universidade Federal de Viçosa (2010) e doutorado em Biodiversidade e Biotecnologia pela Universidade Federal do Tocantins (2017). Atualmente é professor Titular do Centro Universitário Brasileiro de Palmas e engenheiro agrônomo- quadro geral - SETAS.

JOSÉ MARIA RODRIGUES DA LUZ Tem experiência na área de Bioquímica, com ênfase em microbiologia, atuando principalmente nos seguintes temas: cogumelos comestíveis, atividade enzimática, pinhão-manso e tratamentos de resíduos sólidos.

DRA. SÔNIA SALGUEIRO MACHADO PhD em Enzimologia - Delft University of Technology, Pós-doutorado no Departamento de Medicina Molecular do The Scripps Research Institute-TSRI em expressão heterologa, purificação e cristalização de P450 monooxigenase (CYP9) de figado de coelho (2000). Pesquisador visitante no The Scripps Research Institute-TSRI (2010). Atualmente é professor associado (Bioquímica) da Universidade Federal de Alagoas e desenvolve pesquisas em Enzimologia objetivando caracterização e estudos em vivo e in vitro de enzimas como colinesterases de animais como peixes e camundongos para estudos de toxicológicos de monitoramento ambiental e neurofisiológicos.

JOSÉ EXPEDITO C. DA SILVA Professor da Universidade Federal do Tocantins.

Publicação: Outubro de 2019

Digital Object Identifier https://doi.org/10.33911/singular-etg.v1i2.60

Gestão do Processo de Cálculo do Capital para Risco de Crédito com Foco na Melhoria da Eficiência Operacional

TIAGO ENY RELIM DE JESUS GARCIA[1],
RÔMULO DE MEDEIROS PALMEIRA[1],
JOÃO VICENTE PEREIRA[1],
SIMONE BORGES SIMÃO MONTEIRO[1].

1- Departamento de Ciência da Computação - Universidade de Brasília, Caixa Postal 4466, 70910-900, Brasília–DF, Brasil.
(e-mail:tiagoenyrelim@hotmail.com, romulopalmeira@gmail.com, joao.vicentep@gmail.com, simoneborges@unb.br)
Autor Correspondente: Rômulo de Medeiros Palmeira (e-mail: romulopalmeira@gmail.com).

RESUMO - Este estudo apresenta uma abordagem prática para gestão do processo de cálculo do capital para risco de crédito em instituições financeiras. A metodologia utilizada foi a de estudo de caso. O processo contribui para a melhoria da eficiência operacional e confiabilidade do resultado do cálculo. Além disso, constitui ponto de partida para gestão de outros processos da organização.

PALAVRAS-CHAVE - Risco de Crédito, Gestão de Processos, ISO 31000, Matriz GUT.

I. INTRODUÇÃO

Em um ambiente empresarial cada vez mais competitivo, a busca pela eficiência operacional, isto é, fazer mais com menos recursos, constitui etapa chave do processo de gestão [1]. Reduzir gastos sem comprometer a produtividade tem sido um grande desafio para empresas de diversos setores [2].

Um aspecto chave que nem sempre tem a devida importância, é a gestão dos riscos associados aos processos operacionais. Seja pela necessidade de cumprimento de prazos curtos, seja pelo fato de se ter que agregar uma série de assuntos diferentes em uma mesma equipe, é comum que algumas empresas adotem mecanismos frágeis de gestão dos riscos ou mesmo os subestime, podendo incorrer em perdas para a organização.

Neste estudo, será apresentada uma abordagem prática para gestão do processo de cálculo do capital para risco de crédito em instituições financeiras (IF), processo crítico de sucesso para este tipo de organização.

O método utilizado foi o de estudo de caso, por meio de entrevista com a área de interesse e a aplicação in loco das técnicas estudadas.

II. CAPITAL PARA RISCO DE CRÉDITO

A atividade financeira envolve diversos riscos, que são geridos de acordo com sua relevância em termos de impacto econômico para a organização. Dentre os riscos mais relevantes, destacam-se o operacional, liquidez, socioambiental, variação de taxa de juros, mercado e de crédito [3].

O risco de crédito é o de maior impacto potencial, pois está associado à principal atividade bancária, que é a de emprestar dinheiro. Se materializa na ocorrência de inadimplência por parte do devedor, que não honrando sua obrigação junto ao banco, pode gerar perda financeira para a instituição.

Para fazer frente a este risco, os órgãos reguladores estabelecem uma série de requisitos que as IFs devem cumprir, em geral resultando em necessidade de alocação de recursos financeiros por meio de duas fontes principais:

a) Provisões, associadas às perdas esperadas;

b) Capital, associado às perdas inesperadas.

Esses recursos não podem ser utilizados para outras finalidades e representam um custo de oportunidade para os investidores. Se por um lado os recursos para provisões de perda esperada são oriun-

dos do preço das operações, por outro lado o capital é proveniente do patrimônio líquido, que representa o recurso alocado na instituição por seus acionistas.

Portanto, a correta mensuração da necessidade de capital para risco de crédito é fator crítico de sucesso para instituições financeiras de qualquer tipo. Por meio dela, é possível orientar uma série de ações estratégicas, potencializando a busca por portfólios que apresentem maiores retornos com menores riscos.

A. PROCESSO DE CÁLCULO DO CAPITAL PARA RISCO DE CRÉDITO

O cálculo do capital para risco de crédito por meio da abordagem padronizada é definido por uma série de requisitos regulatórios, que são unificados para as IFs e estão detalhados na Circular Bacen n.º 3644/13.

Embora o processo de cálculo de capital possa variar entre IFs, seus requisitos são sempre os mesmos. Desta forma, o processo depende de informações dos clientes, das características das operações e dos riscos a eles associados. Este aspecto torna o processo um tanto complexo, pois é muito comum que estas informações provenham de diferentes fontes dentro da mesma IF.

Com foco na garantia da qualidade do processo de gerenciamento de riscos e de capital em instituições financeiras, em fevereiro de 2017 o Banco Central do Brasil publicou a Resolução do Conselho Monetário Nacional (CMN) n.º 4.557, que entre outras coisas, estabelece:

Art. 7º A estrutura de gerenciamento de riscos deve prever:

(...)

III - sistemas, rotinas e procedimentos para o gerenciamento de riscos;

IV - avaliação periódica da adequação dos sistemas, rotinas e procedimentos de que trata o inciso III;

V - políticas, processos e controles adequados para assegurar a identificação prévia dos riscos inerentes a: (...)

c) mudanças significativas em processos, sistemas, operações e modelo de negócio da instituição;

Os procedimentos apresentados no presente estudo buscam, além de suprir esses requisitos regulatórios, contribuir para a melhoria da qualidade da execução do processo de cálculo do capital, diminuindo o retrabalho e minimizando a possibilidade de erros operacionais que venham a comprometer a integridade dos resultados gerados.

III. NBR ISO 31000:2009

A norma ISO 31000 fornece princípios e diretrizes genéricos para a gestão de riscos. Tendo em vista que qualquer atividade organizacional está sujeita a riscos, a ISO 31000 torna-se aplicável a qualquer organização que tenha por objetivo estabelecer um processo formal de gestão de riscos [4].

A norma deixa claro ainda que sua aplicação é possível a qualquer tipo de risco, independentemente de sua natureza, incluindo estratégias, decisões, operações, processos, funções, projetos, serviços e ativos. A figura a seguir exemplifica de que forma o processo de gestão dos riscos é visto no contexto da Norma:

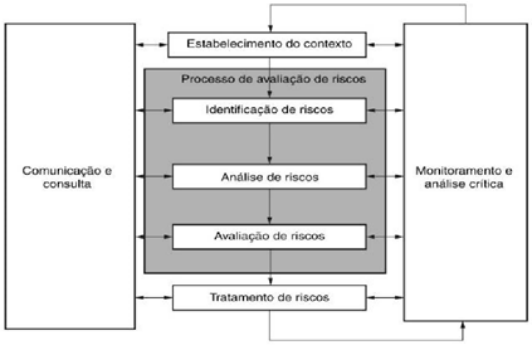


Fig. 1: Processo de Gestão de Riscos. Fonte: ABNT NBR ISO 31010 [5]

Um ponto chave da aplicação da ISO 31000 é o processo de estabelecimento do contexto no qual a gestão de riscos será aplicada. Uma vez estabelecido de forma correta, todo o processo de avaliação dos riscos é feito de forma intuitiva e abrangente.

O processo apresentado na Figura 1 serviu de ponto de partida para o desenvolvimento da gestão do processo de cálculo do capital para risco de crédito e seus aspectos serão tratados com mais detalhe no estudo de caso apresentado nos tópicos a seguir.

IV. GESTÃO DE RISCO DO PROCESSO DE CÁLCULO DO CAPITAL PARA RISCO DE CRÉDITO EM UMA INSTITUIÇÃO FINANCEIRA

O presente estudo de caso é resultado da aplicação do processo de gerenciamento de riscos baseado na Norma ISO 31000 em uma instituição financeira brasileira de grande porte. Refere-se, conforme já mencionado, a um processo central para a tomada de decisões estratégicas nos diversos níveis da organização.

A. ESTABELECIMENTO DO CONTEXTO INTERNO E EXTERNO

Esta etapa busca refletir os aspectos internos e externos que influenciam no atingimento dos objetivos da organização para o contexto abordado; neste estudo, o capital para risco de crédito.

Sakamoto [6] realizou estudo empírico com empresas de software e desenvolveu o conceito de cadeia de valor para explicar o desdobramento dos processos em quatro principais estruturas: macroprocesso, processo, atividade e tarefa. Dessas estruturas, o processo é um grupo de atividades interligadas logicamente que fazem uso dos recursos da organização para gerar resultados e adicionar valor. Baseado em seu estudo, foi construída a cadeia de valor da IF, conforme se observa a seguir:

Fig. 2: Cadeia de Valor da Instituição Financeira

O processo de gestão de riscos, posicionado na base da cadeia de valor, representa sua importância fundamental para alicerçar as decisões estratégicas e de continuidade dos negócios. O cálculo do capital é um dos diversos processos ali inseridos, sendo, no entanto, o de maior relevância econômica para a organização.

O Guia para o Gerenciamento de Processos de Negócio – Corpo Comum de Conhecimento [7] descreve as principais etapas do ciclo de vida dos processos: modelagem, análise, desenho, gerenciamento do desempenho e transformação.

A etapa de modelagem começa com o planejamento, que visa assegurar o alinhamento do contexto dos processos de negócio com os objetivos estratégicos da organização. Planejar a modelagem é entender os processos de negócio no escopo da organização como um todo e pode ser feito com o mapeamento do processo, com a descrição, o desenho e a identificação da inter-relação entre as atividades, como mostra a figura 3.

Para o processo de cálculo de capital, foram utilizadas entrevistas e análise da documentação como técnicas de mapeamento e a ferramenta Bizagi Modeler para a fluxogramação que detalha o nível de execução do trabalho.

A execução é iniciada com a disponibilização das bases de dados, que passa pelas equipes de teste de estresse, perda esperada, modelagem, mitigadores e validação, responsáveis pelas atividades de modelagem, fornecimento de diretrizes, parâmetros, informações de garantias, validação da base e finalmente retorna à equipe de teste de estresse para o cálculo final.

Dessa forma, o processo de cálculo é baseado em dados fornecidos por cinco áreas diferentes e seus resultados são insumo para diversas outras áreas, sendo complexa a tarefa de atender às mais variadas expectativas que mudam constantemente.

Tendo em vista as informações detalhadas até aqui, foram estabelecidos seus contextos interno e externo.

O estabelecimento destes contextos auxilia no desenvolvimento de atividades que atendam de forma adequada tanto às exigências regulamentares e da organização, quanto às expectativas dos usuários da informação, evitando que se tenha conhecimento da existência de áreas intervenientes que não tenham sido contempladas somente após a conclusão do trabalho.

B. ANÁLISE DOS RISCOS

Inicialmente, foi realizado levantamento histórico do tempo de processamento do cálculo do capital, que ocorre mensalmente. Em todas as datas de referência, foi verificada a presença de reprocessamento dos dados. Os motivos são diversos e vão desde a ocorrência de erros operacionais à necessidade de inclusão de novos dados para atender a demandas específicas de outras áreas.

Para tornar mais tangível o efeito danoso do retrabalho, foi associado um custo de R$ 200,00 por hora de processamento, referente ao valor médio da remuneração por hora dos três analistas responsáveis pelo processo.

Como se observa no Quadro 2, o custo médio do reprocessamento superou o custo do processamento em 2015 e 2016, recuando a partir de 2017, embora ainda em valores expressivos.

Foi apresentada à equipe responsável pelo processo, uma planilha em Excel, para que fossem elencados os principais fatores de risco que levaram à necessidade de reprocessamento dos cálculos. Após levantamento dos dados, chegou-se ao seguinte resultado, materializado em riscos:

I Risco Operacional
   a) Erros de disponibilização e tratamento dos dados
   b) Falhas em sistemas de TI

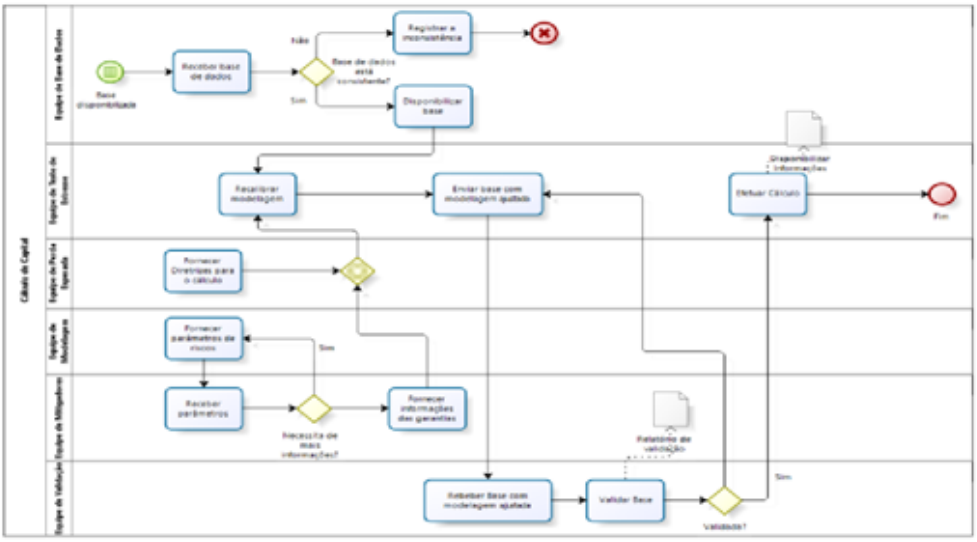


Fig. 3: Processo de Cálculo do Capital

Tab. 1: Custo do processamento (P: Custo do Processamento e R: Custo do Reprocessamento)

		jan	**fev**	**mar**	**abr**	**mai**	**jun**	**jul**	**ago**	**set**	**out**	**nov**	**dez**	**Média**
2.015	P	3.582	3.556	4.734	3.643	3.817	4.178	4.521	5.005	4.593	4.641	3.229	4.277	4.148
2.016	R P	2.427	6.950	2.706	4.310	6.761	12.208	6.216	3.778	7.993	4.955	8.294	8.459	6.255
		4.628	3.693	879	2.221	2.610	2.403	2.458	2.461	2.711	3.082	3.456	2.719	2.777
2.017	R P	1.896	8.476	3.562	1.475	1.279	6.191	1.844	515	2.661	7.102	810	2.798	3.217
		3.049	2.892	2.732	2.967	3.398	3.121	3.201	3.137	2.993	2.712	2.991	2.869	3.005
2.018	R P	2.718	175	1.818	510	3.689	4.508	2.276	549	2.773	2.176	2.739	2.407	2.195
2.993		2.990	3.674	2.640	2.669									

c) Falhas humanas
d) Perda de informações históricas

II Risco Legal
e) Atrasos de entregas regulamentares
f) Divulgação de informação sigilosa

III Risco de Descontinuidade
g) Ausência de documentação adequada
h) Ausência de mais de um profissional para desempenho das funções
i) Ausência de manutenção tempestiva de rotinas e sistemas

C. PRIORIZAÇÃO DOS RISCOS E TRATAMENTO

A Norma ISO 31010 [5] apresenta uma série de ferramentas para o processo de análise e avaliação dos riscos. Para este estudo, optou-se pela utilização da Matriz GUT, por julgá-la mais adequada diante das informações disponíveis.

A Matriz GUT (Gravidade, Urgência e Tendência) é uma técnica de aplicação bastante intuitiva e que tem como benefício a utilização da experiência de profissionais envolvidos na área para a pontuação dos riscos e sua posterior classificação [8].

Foi fornecida para os funcionários da equipe uma planilha em Excel, com cada um dos riscos identificados e uma escala de classificação de 1 a 5, da seguinte forma:

Tab. 2: Pontuação da Matriz GUT. Adaptado de [8]

1	Sem gravidade	Pode esperar	Não irá mudar
2	Pouco grave	Pouco urgente	Irá piorar a longo prazo
3	Grave	O mais rápido possível	Irá piorar
4	Muito Grave	Urgente	Irá piorar em pouco tempo
5	Extremante Grave	Precisa de ação imediata	Irá piorar rapidamente

Após classificação por cada participante, foram calculados os valores médios em cada item e arredondados para o inteiro correspondente. Por fim, os valores foram multiplicados (G x U x T) e ordenados do maior para o menor [10], resultando na seguinte ordem de priorização com seus respectivos tratamentos.

Descontinuidade	Ausência de documentação adequada	Documentar o processo de geração dos dados e validação	5	5	5	125
Operacional	Perda de informações históricas	Realizar backup periódico	5	5	3	75
Descontinuidade	Ausência de mais de um profissional para desempenho das funções	Definir substituto para as funções	5	3	5	75
Legal	Atrasos de entregas regulamentares	Criar alertas para as áreas que fornecem informação e reportar andamento dos trabalhos ao gestor	5	5	2	50
Operacional	Falhas humanas	Impelentar revisão para processos críticos	4	3	2	24
Descontinuidade	Ausência de manutenção tempestiva de rotinas e sistemas	Incluir processo de avaliação se os programas que estão sendo	2	4	3	24
		utilizados são os últimos aprovados				
Operacional	Erros de disponibilização e tratamento dos dados	Validação dos resultados e de acordo do gestor	3	3	1	9
Legal	Divulgação de informação sigilosa	Criar regra de acesso aos dados	3	2	1	6
Operacional	Falhas em sistemas de TI	Quantificar as falhas e reportar à tecnologia	2	1	2	4

D. MONITORAMENTO E REPORTE

Foi proposto um processo contínuo de monitoramento dos riscos e reporte dos resultados, tanto para os profissionais envolvidos no processo quanto para seus demais intervenientes [9]. É importante que os responsáveis pelo processo tenham consciência dos riscos a que estão expostos e, principalmente, que sejam responsabilizados por sua gestão.

Assim, foi sugerido também a definição de indicadores e metas de acompanhamento das ações de tratamento dos riscos, com o objetivo de materializar os esforços pela manutenção do processo e também prevenir eventuais desvios ao longo do tempo.

Tab. 3: Indicadores e Metas de Acompanhamento

Ausência de documentação adequada	Existência de manual atualizado para execução do processo	Número de dias desde a última revisão do manual do processo	Inferior a 360 dias
Perda de informações históricas	Armazenamento das bases processadas	Número de bases de dados históricas disponíveis	60 últimos meses
Ausência de mais de um profissional para desempenho das funções	Quantidade de profissionais responsáveis pelo processo	Quantidade de profissionais aptos a exercer a função	3 funcionários
Atrasos de entregas regulamentares	Número de dias de atrasos em relação à data de entrega regulamentar	Número de dias de atraso na entrega de demanda regulatória	0 dias de atraso

Por fim, foi proposta a meta de reduzir o tempo médio de reprocessamento para o ano de 2018 em 60%, com base no tempo médio observado ano de 2017.

Acredita-se que uma vez aplicados os tratamentos e acompanhamentos aqui propostos, este objetivo possa ser atingido sem maiores problemas.

V. CONCLUSÃO

Este estudo focou no processo de cálculo do capital para risco de crédito de uma instituição financeira. Foram identificados, avaliados, priorizados e definidos tratamentos para os principais riscos que influenciam no atingimento dos objetivos da organização. Os resultados foram apresentados à equipe responsável pelo processo e sugestões de monitoramento e reporte foram apresentadas. Espera-se que o estudo de caso aqui apresentado possa servir de ponto de partida para o gerenciamento de outros processos críticos da organização, contribuindo para sua eficiência operacional, minimizando riscos e custos desnecessários.

REFERÊNCIAS

[1] Ceretta P. S. and Niederauer C. A. P. “Rentabilidade e eficiência no setor bancário brasileiro,” Rev. Adm. Contemp., vol. 5, no. 3, pp. 7–26, 2001.

[2] Fleury A. C. C. and Fleury M. T. L., “Estratégias competitivas e competências essenciais: perspectivas para a internacionalização da indústria no Brasil,” Gestão e Produção, vol. 10, no. 2, pp. 129–144, 2003.

[3] BRASIL. Resolução n. 4557, de 23 de fev. de 2017. Disponível em: <http://www.bcb.gov.br/#!/n/normasbc> Acesso em: 12 jul. 2018.

[4] ABNT. NBR ISO 31000. Gestão de riscos — Princípios e diretrizes. 2009. Rio de Janeiro. Brasil.

[5] ABNT. ISO/IEC 31010. Técnicas para o Processo de Avaliação de riscos. 2012. Rio de Janeiro. Brasil.

[6] Sakamoto, A.; Vasconcellos, M. and Serio, L. Cadeia de valor de empresas de software, a partir de suas inovações: um estudo empírico. Anais dos SIMPOI: Simpósio de Administração da Produção, Logística e Operações Internacionais, 2009.

[7] CBOK, Guia para o Gerenciamento de Processos de Negócio, Corpo Comum de Conhecimento ABPMP BPM CBOK V.3.0, 2013.

[8] Lalonde, C. and Boiral, C. Managing risks through ISO 31000: A critical analysis. Risk Management. 14: p. 272-300. 2012. Disponível em: <https://doi.org/10.1057/rm.2012.9>. Acesso em 16 jun. 2018.

[9] Leitch, M. ISO 31000:2009 – The New International Standard on Risk Management. Risk Analysis. Vol. 30, n. 6. 2010. Disponível em <https://onlinelibrary.wiley.com/doi/full/10.1111/j.15396924.2010.01397.x> Acesso em: 16 jun. 2018.

[10] Oliveira, L.; Filho, F. and Madeira, M. Aplicação da Matriz GUT em uma microempresa de assistência técnica. Encontro Internacional Sobre Gestão Empresarial e do Meio Ambiente. Dez. 2016.

TIAGO ENY RELIM DE JESUS GARCIA Possui graduação em Ciências Atuariais pela Pontifícia Universidade Católica de Minas Gerais (2009) e pós graduação em Matemática pela Faculdades Integradas de Jacarepaguá (2012).

SIMONE BORGES SIMÃO MONTEIRO Possui graduação em Engenharia Química pela Universidade Federal de Uberlândia (1995), mestrado em Engenharia de Produção pela Universidade Federal de São Carlos (1998) e doutorado em Engenharia de Produção pela Universidade Federal de São Carlos (2006). Atualmente é professora adjunto de engenharia de produção da Faculdade de Tecnologia. Tem experiência na área de Engenharia de Produção, com ênfase em Garantia de Controle de Qualidade, atuando principalmente nos seguintes temas: gestão da qualidade, gestão de risco, aprendizagem baseada em projetos (pbl), cadeia de produção agroalimentar e coordenação da qualidade.

RÔMULO DE MEDEIROS PALMEIRA Bacharel em Ciências Econômicas pelo UNICEUB (2001), especialista em Estatística Aplicada pela AEUDF (2013) e mestre em Computação Aplicada pela UnB (2019). Gerenciou microempresa do segmento esportivo no período de 1998 a 2003 (Academia de Ginástica NEW LIFE). Assessor Empresarial no Projeto Basileia Risco de Crédito de 2013 a 2016, contribuindo nos processos de análise do Escopo IRB, estudos de impacto do cálculo de capital, estruturação/homologação de dados corporativos para capital, modelagem da volatilidade para capital econômico e integração na gestão dos projetos de capital regulatório/econômico, teste de estresse, painel de risco e monitoramento de modelos. Desde 2017 exerce a função de Assessor Empresarial na Diretoria de Gestão de Riscos de um banco.

JOÃO VICENTE PEREIRA Graduado em Estatística pela Universidade de Brasília (UnB) e em Ciências Contábeis pela Fundação Universidade Tocantins, com pós-graduação em Estatística pela Universidade Paulista, MBA em Gestão de Crédito pela Fundação Getúlio Vargas (FGV) e Mestrado em Computação Aplicada pela Universidade de Brasília (UnB).

Publicação: Outubro de 2019

Digital Object Identifier https://doi.org/10.33911/singular-etg.v1i2.62

Identificando e Avaliando Dívidas Técnicas no Processo de Testes de Software

CLEYDIANE LIMA DE SOUSA[1], ARILO CLAUDIO DIAS NETO[2].

1 - Centro de Estudos e Sistemas Avançado do Recife (CESAR)
2- Instituto de Computação (IComp), Universidade Federal do Amazonas, Manaus, Brasil
(e-mail:cleydiane.lima@cesar.org.br, arilo@icomp.ufam.edu.br)
Autor Correspondente: Cleydiane Lima de Sousa (e-mail: cleydiane.lima@cesar.org.br).

RESUMO - Dívida Técnica (DT) está relacionada a tarefas que devem ser executadas e são acumuladas ao longo de um projeto para serem realizadas posteriormente. DTs não solucionadas tendem a criar dependências e aumentar o grau de complexidade para correção, resultando em maior esforço, custo e retrabalho em um projeto. A sua identificação e gerenciamento ao longo de um projeto é essencial para minimizar seu impacto e consequências negativas. Estudos apontam teste de software como sendo uma das principais áreas impactadas por DTs em projetos de software. Neste artigo foram mapeadas a partir da literatura técnica 22 possíveis DTs relacionadas ao processo de teste, suas causas e indicadores. Elas foram avaliadas por meio de um survey com os profissionais da área de teste de software. Como resultados do survey, são apresentados os níveis de concordância obtidos para cada DT, suas causas e indicadores sugeridos.

PALAVRAS-CHAVE - Dívida Técnica, Teste de Software, survey.

I. INTRODUÇÃO

O aumento da demanda por novos produtos de software tem desafiado cada vez mais as organizações de software a atender aos prazos e qualidade exigidos pelos seus usuários. Isto requer alguns sacrifícios diante da possível falta de tempo, recursos, profissionais capacitados, dentre outras limitações ou imprevistos em um projeto de software. De acordo com Pressman [1], em geral, nos projetos de software, quando os prazos ou recursos se tornam escassos, as organizações tendem a comprometer tarefas e práticas relacionadas à qualidade de software. Como consequência, problemas de qualidade podem ser observados no produto durante o projeto ou após sua implantação. Tais tarefas comprometidas precisam ser concluídas em algum momento ao longo do projeto, ou se não concluídas, podem gerar uma dívida, ou déficits ao projeto. Assim, surge o conceito de Dívida Técnica (DT: do inglês Technical Debt) dentro da área de Engenharia de Software, que representa um conjunto de tarefas que devem ser executadas e são acumuladas ao longo de um projeto para serem realizadas posteriormente.

O conceito de DT foi criado por Cunningham [2], que definiu como: "A primeira vez que a qualidade do código é comprometida é como se estivesse incorrendo em débito. Um pequeno débito acelera o desenvolvimento até que seja pago por meio da reescrita do código. O perigo ocorre quando o débito não é pago. Cada minuto em que o código é mantido em inconformidade, juros são acrescidos na forma de reescrita". Assim, a DT que não é solucionada com o decorrer do tempo tende a criar dependências e aumentar o grau de complexidade para correção, ou até mesmo ocorrer sua desprioriorização pelo acúmulo de novas dívidas. Este tema vem ganhando importância dentro das diversas áreas da Engenharia de Software. Li et al. [3] apresentam um levantamento das áreas que podem ocasionar DT. Dentre as áreas apresentadas neste estudo, a área de teste de software possui um grande destaque como causadora de DTs. Segundo Bertolino [4], teste de software abrange diversas atividades no decorrer do ciclo de desenvolvimento de software,

nas quais são realizadas observações para verificar o comportamento do que está sendo desenvolvido e validar se está como o esperado. De acordo com a norma ISO/IEC/IEEE 29119 [5], um processo de testes é formado por diversas atividades, incluindo: planejamento, monitoramento e controle, projeto, execução e análise dos resultados dos testes. Ao longo deste processo, a equipe de testes pode deixar de realizar atividades ou deixá-las para outro momento, gerando DT que precisam ser identificadas, controladas e solucionadas ao longo do projeto.

No entanto, a identificação e monitoramento de DTs relacionadas ao processo de teste de software, suas eventuais causas e os indicadores que permitem observá-las em um projeto de software ainda são pouco explorados em pesquisas acadêmicas. Em geral, elas estão dispersas em diversas fontes (artigos científicos e relatos da indústria). Baseado neste cenário, este artigo apresenta um conjunto de 22 DTs relacionadas ao processo de testes de software (reportadas na literatura técnica), as eventuais causas para cada DT e indicadores que possibilitam a sua identificação em um projeto de software. Tais DTs foram organizadas em 2 grupos: DTs relacionadas a atividades gerenciais de teste (planejamento, monitoramento e controle) e DTs relacionadas a atividades técnicas de teste (projeto e execução dos testes). Este conjunto foi avaliado por meio de um survey com a participação de 64 especialistas em teste de software, que responderam o quanto concordam com cada DT, suas causas e indicadores. Tais DTs foram ordenadas de acordo com o grau de concordância obtido ao longo do survey e uma análise quantitativa e qualitativa dos resultados é apresentada. O artigo segue a seguinte estrutura: Seção II apresenta o referencial teórico, descrevendo os conceitos de DT e do processo de TS e apresenta as DTs em TS identificadas a partir da literatura técnica. Seção III descreve o planejamento e execução do survey, com objetivo de avaliar a concordância dos profissionais de TS quanto as DTs identificadas na literatura. Seção IV apresenta os resultados obtidos a partir da análise dos dados coletados ao longo do survey. Finalmente, a seção V apresenta as conclusões e trabalhos futuros.

II. REVISÃO DA LITERATURA

A. DÍVIDA TÉCNICA EM ENGENHARIA DE SOFTWARE

Cunningham [2] é considerado o pioneiro em DT em Engenharia de Software. Para ele, em curto prazo, adquirir uma DT pode se apresentar como um fator positivo, desde que se tenha a visão que no futuro é necessário pagar a dívida para não acumular. Este cenário se apresenta, por exemplo, em situações em que é necessário gerar uma DT de código mal escrito para contornar problema de prazo curto para entrega de um módulo.

Em contrapartida à visão de DT centrada em código, Klinger [6] enfatiza o fato que uma DT não vem somente da parte técnica, de código. Ela vai além disso. Outros fatores interferem em seu surgimento, como diversas funções técnicas e não-técnicas, tais como testes de produtos, estratégia de marca, questões jurídicas, marketing, dentre outros. Observar estes fatores e o que mais envolve todo o processo de um produto ajudam a entender melhor porque uma DT veio a ocorrer e foi ou não paga.

Dentre outras definições, Guo [7] considera DT como "uma metáfora para artefatos imaturos, incompletos ou inadequados no ciclo de vida de desenvolvimento de software que causam maiores custos e menor qualidade". A criação destes artefatos pode acelerar o desenvolvimento em curto prazo. Porém, em longo prazo, a má qualidade dos artefatos tende a gerar um maior custo, devido esforços de manutenção e força de trabalho para correções. Torna-se perceptível os riscos no acúmulo de dívidas técnicas, as dependências entre as dívidas e outros artefatos aumentam gradativamente a complexidade de pagamento destas dívidas.

Para Shull [8], a DT varia de equipe para equipe, havendo a necessidade da equipe refletir sobre qual tipo de dívida se deve considerar com uma maior preocupação e estar sempre monitorando. O monitoramento da DT é necessário para não se perder o controle e não adquirir nova DT sem perceber, acumulando-a ao ponto de quando for percebida ser muito caro para ser paga.

Quanto a relevância do tema para comunidade de pesquisa, Alves et al. [9] apresentam uma revisão sistemática que apontou que o tema DT obteve trabalhos com publicações principalmente a partir de 2010. Porém, ainda é um tema novo e seus diferentes tipos e indicadores ainda não estão organizados. Já Li et al. [3] complementam estes dados por meio de sua pesquisa que identificou trabalhos publicados a partir do ano de 1992, ano em que o conceito de DT foi apresentado, até 2013.

B. PROCESSO DE TESTE DE SOFTWARE

Segundo Mettle e Hass [10], o processo de testes consiste nas atividades do ciclo de vida, tanto estática e dinâmica, preocupadas com planejamento, preparação e avaliação de produtos de software e produtos de trabalho relacionados para determi-

nar que satisfaçam os requisitos especificados, para demonstrar que estão aptos para o efeito e para detectar defeitos. Um processo de teste pode definir melhor as atividades a serem realizadas dentro da área de testes, além de prover melhorias para que as atividades sejam feitas de maneira organizada.

A norma ISO/IEC/IEEE 29119 [5] sugere um processo de testes de software com etapas e artefatos para serem gerados. O processo de testes é dividido em três subprocessos: organizacional, gerenciamento e dinâmico. O processo de teste organizacional deve compreender atividades para a criação, revisão e manutenção de especificações de teste organizacionais. Ele também deve abranger o monitoramento da conformidade organizacional [5]. O objetivo destas atividades dentro das organizações está em ajudar a se ter um processo e controlar melhor o que está sendo realizado para que os testes ocorram com sucesso. Os outros dois subprocessos são formados pelas seguintes atividades [5]:

- Processo de Gerenciamento dos Testes: Planejamento do Processo de Teste, Monitoramento e Controle de Teste e Conclusão de Teste.
- Processo de Teste Dinâmico: Projeto e Implementação dos Testes, Configuração e Manutenção do Ambiente de Teste, Execução dos Testes e Reporte dos Incidentes de Teste.

Tais atividades serão usadas neste trabalho para organização das DTs relacionadas ao processo de testes, indicando o momento em que elas surgem em um projeto.

C. TRABALHOS RELACIONADOS SOBRE DÍVIDA TÉCNICA EM TESTE DE SOFTWARE

A maioria dos trabalhos que abordam DT apresenta seu foco em DTs identificadas na etapa de codificação. Nesta seção, serão apresentados trabalhos centrados em DT na área de teste de software com o objetivo de identificar as principais contribuições.

Shah et al. [11] relatam sobre a DT estar relacionada a diversas atividades, em que, uma delas é a de teste de software. Os autores citam DTs dentro de teste de software, focando em DTs que podem vir a ocorrer quando há execução de somente testes exploratórios. Os autores ainda destacam que a falta de planejamento e documentação tendem a criar uma grande dificuldade para realizar testes de regressão de forma adequada, o que tende a gerar retrabalho e também uma cobertura de defeitos inadequada. Logo, adquire-se uma DT maior a cada vez que se cria mais funcionalidades para o produto, e a regressão não identifica os defeitos.

Bavani [12] descreve uma entrevista com Johanna Rothman e Lisa Crispin (importantes pesquisadores da área de testes ágeis) realizada em 2012 que busca identificar a opinião deles sobre DT em equipes ágeis e distribuídas. Os entrevistados se preocupam em sanar as DTs que aparecem no processo de teste, e uma sugestão é automatizar, mas sem perder tempo com a manutenção. Além disso, para sanar problemas em equipes distribuídas, Crispin e Rothman concordam que as equipes de teste devem ser integradas com o restante da equipe, e que é necessário ter um ritmo sustentável para não surgirem novas DTs.

Wiklund [13] descreve uma pesquisa que visou identificar os contribuintes comuns a DTs acumuladas em testes automatizados, para avaliar a consciência desta dívida em organizações que usam o teste automatizado. A pesquisa foi realizada a partir de uma entrevista semiestruturada com participantes de uma equipe de software em um projeto de telecomunicações. Em suas conclusões, o autor relatou que não foi possível detectar como planejar, monitorar e deixar a DT de automação em um valor aceitável, destacando problemas existente que a automação apresenta, sem visualizar soluções.

Snipes et al. [14] apresentam um estudo que identificou os fatores na tomada de decisão para corrigir defeitos e os custos que envolvem isto, sob a perspectiva de DTs. Os autores identificaram vários custos que são encadeados com a manutenção de defeitos. Foi ressaltado que na tomada de decisão para corrigir os defeitos, para incluir DT é necessário incluir custo-benefício, acrescentando-os junto com os fatores identificados.

Deve-se levar em consideração que os trabalhos relacionados à DT em teste de software, além de serem poucos, apresentam DTs em tipos específicos de teste (ex: testes exploratórios ou automação de testes). Assim, realizar um levantamento de DTs no processo de testes de software se mostra uma importante contribuição para comunidade de engenharia de software, com objetivo de cada vez mais ser cumprido o que for planejado dentro de um processo de testes de software, sem deixar em débito o que possa prejudicar a saúde de um projeto.

III. METODOLOGIA

A. DÍVIDAS TÉCNICAS EM TESTE DE SOFTWARE

Para identificar DTs na literatura técnica, foram realizadas pesquisas manuais em diversas bibliotecas digitais, tais como: ACM Digital Library, IEEExplore, Scopus, ScienceDirect e Google Scholar.

Os trabalhos que foram identificados abordam DT em teste de software, além de trabalhos em teste de software que não lidam com termo DT, porém, apresentam menção de problemas (dificuldades) na área.

Depois da identificação das DT, foram apontados causas e indicadores de cada DT. Neste contexto, Causa é o acontecimento que é feito ou deixa de ser feito para gerar a DT; e os Indicadores são indícios para perceber a DT em um projeto de software. Nas Tabelas 1 e 2 são apresentadas as DTs que foram encontradas na literatura técnica, separadas por DTs associadas ao Gerenciamento dos Testes (Tabela 1) e Projeto e Execução dos Testes (Tabela 2). Ao lado do título de cada DT, estão apresentadas as referências de onde elas foram extraídas. Também são apresentados as causas e indicadores associados a cada DT em análise.

Em seguida foi realizado o survey para a validação do conjunto de DT junto a especialistas em teste de software, a ser descrito na próxima seção.

B. SURVEY SOBRE DTS, CAUSAS E INDICADORES EM TESTE DE SOFTWARE

Nesta seção serão apresentados o planejamento e projeto de um survey que visou investigar um conjunto de dívidas técnicas (DTs) relacionadas ao processo de testes de software, suas causas e indicadores, sob o ponto de vista de profissionais da área.

O survey busca caracterizar como testadores avaliam as DTs identificadas na literatura técnica como relacionadas à área de teste de software, e ainda se há variações de percepção dentro da população analisada quanto ao nível de relevância.

1) Questão de Pesquisa

A questão de pesquisa que norteou este estudo foi a seguinte: Qual o nível de concordância de 22 DTs (suas causas e indicadores) presentes na literatura técnica sob o ponto de vista de profissionais de Teste de Software? A Variável Independente é o Conjunto Inicial de DTs, causas e indicadores (Tabelas 1 e 2). As Variáveis Dependentes são: O nível de concordância para cada DT, causa e indicador, pertencentes ao conjunto inicial e mantidos no conjunto final de acordo com a opinião dos profissionais.

2) Definição do Instrumento

O instrumento adotado neste survey é formado por quatro etapas sequenciais, descritas a seguir:

Convite. A partir do link presente no e-mail do convite, o participante era redirecionado à página de apresentação do survey, que dá acesso ao questionário. Apresentação. Foi desenvolvida uma página web contendo informações sobre o survey, e ao final desta página a solicitação do consentimento para participarem do survey.

Caracterização do Participante. Após a tela de apresentação o participante era direcionado para um formulário de caracterização, que apresenta o objetivo da pesquisa e perguntas que visam caracterizar o grau de conhecimento/experiência dos participantes a respeito do tópico do estudo.

Avaliação de DTs, Causas e Indicadores. Devido ao grande número de DTs e suas classificações dentro do processo de testes, optou-se pela separação da avaliação das DTs (suas causas e indicadores) em dois questionários, que deveriam ser respondidos por grupos diferentes de participantes, de acordo com o perfil de cada participante, definido no passo anterior. Os questionários são:

- O questionário "Planejar e Concluir": apresenta 9 Dívidas Técnicas, causas e indicadores nos processos gerenciais de Planejamento, Monitoramento e Controle de Teste de Software e Análise/Conclusão de Resultados (conforme a ISO 29119).
- O questionário ¨Projetar e Executar¨: apresenta 13 Dívidas Técnicas, causas e indicadores dos processos de Projetar e Executar Teste de Software (conforme ISO 29119).

Por fim, ao final de cada um dos dois questionários foi oferecido um campo para comentários gerais, que poderia ser preenchido como campo opcional. Em ambos os questionários, para cada DT (suas causas e indicadores), foram apresentadas seis opções de respostas (obrigatório), além de um campo para comentários (opcional). As opções de respostas são apresentadas na Tabela 3.

Através da combinação entre as opções de resposta, é possível identificar a concordância dos participantes quanto aos itens de uma DT (descrição da DT, causas e indicadores). Por exemplo, as respostas R1, R2 e R4 (Tabela 3) são opções que representam a concordância dos participantes quanto as causas da DT analisada. Desta forma, para cada questão, a partir do somatório dos participantes que concordaram com determinado item (dívidas técnicas, causas ou indicadores) é realizada a divisão pelo total de participantes que responderam ao questionário, resultando assim em uma porcentagem que representa o nível de concordância dos participantes quanto o item analisado.

Na próxima seção são apresentados os resultados obtidos a partir da análise das respostas coletadas

Tab. 1: Dívidas Técnicas, Causas e Indicadores associados ao Gerenciamento dos Testes.

ID	Dívidas Técnicas	Causas	Indicadores
DT01	Cronograma de teste não definido/ inadequado [15] [16] [17]	- Falta de experiência da equipe de teste	- Cronograma não foi criado
		- Tempo limitado para os testes como premissa do projeto	- Cronograma sem considerar atividades nos processos de testes
		- Não saber quando iniciar os testes	
		- Não saber quando terminar os testes	
DT02	Ferramentas de teste não utilizadas/ aplicáveis no projeto [4] [15] [16] [18]	- Falta de experiência da equipe de teste	- Quantidade de recurso disponível menor que o valor das ferramentas
		- Falta de recurso	- Diversidade de ferramentas disponíveis menor do que o esperado para a cobertura dos testes
DT03	Riscos de teste não definidos/inadequados [15] [19]	- Ausência de critérios para seleção dos testes	- Não ter os riscos mapeados
			- Premissas do projeto que impactam nos testes não consideradas nos levantamentos de riscos
DT04	Planejamento de testes não adequados para o projeto [11] [16] [18] [20]	- Não entendimento dos requisitos do projeto	- Tipos de testes não cobrem os requisitos
		- Não buscar entender melhor o projeto	
DT05	Equipe de teste não definida/inadequada para projetar e executar os testes [11] [15] [18]	- Falta de recurso	- Planejar uma quantidade X de pessoas e ter disponível menos pessoas
		- Falta de comunicação dos contratantes com o gerente de testes	- Não ter todos os perfis mapeados
		- Falta de planejamento e estratégia do projeto	
		- Ausência de critérios para seleção de equipe	
DT06	Fornecer um entregável com bugs [20] [21]	- Falta de regressão	- % bugs sendo encontrados em funcionalidades já testadas
		- Falta de tempo para corrigir	- % bugs conhecidos
DT07	Bugs sem correções [21]	- Falta de relatório dos bugs	- % Bugs não corrigidos
		- Falta de tempo para correção	
DT08	Falta de controle de quais tipos de testes estão sendo executados [13] [15]	- Falta de relatório de execução	- A quantidade de tipos de testes executados pela equipe não está presente em relatórios
		- Falta de tempo de fazer relatórios	
DT09	Problemas no software não aceito pelo cliente [16] [22]	- Cliente não ter conhecimento da estratégia (critérios de aceitação e saída)	- Quantidade de criticidades dos bugs para aquela entrega não aceita pelo cliente
		- Equipe de teste não checando a documentação do projeto (ex: caso de uso, ux-guide)	- % de cobertura dos testes realizados, não aceitos pelo cliente
			- Cliente esperando que sejam feitos testes que não estão sendo aplicados
DT10	Scripts de teste não implementados [3]	- Tempo limitado para os testes	- % de scripts de teste implementados (esperado / realizado)
		- Equipe de teste inexperiente	
		- Entrega do desenvolvedor atrasada	
		- Cronograma de teste não definido/inadequado	
DT11	Criação de cenários de testes não finalizados [3][10] [16]	- Tempo limitado para os testes	- % de cenários de testes implementados (esperado / realizado)
		- Equipe de teste inexperiente	
		- Falta de definição dos requisitos	
		- Cronograma de teste não definido/inadequado	
DT12	Falta de documentação dos testes que serão executados [11] [13] [15] [16]	- Equipe de teste inexperiente	- Quantidade de documentos
		- Tempo limitado para os testes	
		- Cronograma de teste não definido/inadequado	

Tab. 2: Dívidas Técnicas, Causas e Indicadores associados ao Projeto e Execução dos Testes.

ID	Dívidas Técnicas	Causas	Indicadores
DT13	Necessidade de refatorar os scripts de testes [15] [16]	- Equipe de teste inexperiente	- Falta de atributos no código para scripts de testes
		- Falta de conhecimento técnico	
		- Falta de organização na hora de iniciar os scritps	
		- Alteração na funcionalidade	
DT14	Cenários/scripts de testes que não representam o estado atual da funcionalidade [16] [23] [24]	- Não realizar manutenção nos casos de testes e scripts de testes	- % de cenários/scripts de testes não coerente com a atual funcionalidade
		- Equipe de teste inexperiente	- Quantidade de tempo disponível para manutenção menor que o planejado
DT15	Casos/scripts de testes desatualizados na ferramenta de gestão [15] [16] [17]	- Tempo limite para os testes	- % de testes sendo projetados sem incluir/atualizar os casos de testes / scripts das ferramentas de gestão
		- Cronograma de teste não definido/inadequado	- Quantidade de tempo disponível para manutenção menor que o planejado
		- Falta de manutenção	
DT16	Casos de testes não executados [3] [25]	- Atraso na liberação da versão para teste	- Quantidade de casos de testes não executados (ou %)
		- Tempo curto para execução dos testes	- Quantidade de tempo disponível para execução menor que o planejado
		- Execução apenas de testes automatizados	
		- Cronograma de teste não definido/inadequado	
		- Regressão não executada	
DT17	Scripts de Teste não executados [3]	- Atraso na liberação da versão de teste	- Quantidade de scripts de testes não executados (ou %)
		- Tempo curto para execução dos testes	- % Scripts falhando
		- Cronograma de teste não definido/inadequado	- Quantidade de tempo disponível para execução menor que o planejado
		- Regressão não executada	
DT18	Sobrecarga de atividades em determinados períodos do projeto [16] [27]	- Cronograma de teste não definido/inadequado	- Quantidade de tempo para trabalho menor que o tempo planejado para as atividades no período
		- Falta de divisão de atividades	- % de versões lançadas em determinado período do projeto
		- Atraso na liberação da versão para teste	
DT19	Bugs encontrados tardiamente [15] [20]	- Falta de Cronograma	- % de tipos de testes não sedo executados há tempo ou não sendo executados
		- Falta de integração contínua	- % de bugs encontrados em momentos errados (sem término de desenvolvimento)
		- Desenvolvimento atrasado	
		- Testes iniciados tardiamente	
DT20	Acúmulo de bugs não corrigidos [21]	- Não corrigir os bugs por criticidade	- % funcionalidades importantes com erros graves
		- Falta de tempo para correções	- % de bugs com criticidades erradas
		- Equipe de desenvolvimento não disponibiliza tempo para corrigir bugs	
DT21	Falta de cobertura dos requisitos do sistema (funcionais e não funcionais) [3] [16] [20]	- Inexperiência dos testadores	- Quantidade de bugs sendo encontrados por falta de cobertura de certos tipos de testes
		- Execução de somente um tipo de teste	- % de requisitos não cobertos pelos tipos de testes aplicados
DT22	Funcionalidades já finalizadas presentando bugs [15] [20]	- Pouco tempo para executar os testes	- % bugs sendo encontrados em funcionalidades já testadas
		- Testes não fazem mais sentido (funcionalidade alterada)	- % de bugs encontrados nos testes de regressão
		- Não executar testes de regressão	

Tab. 3: Opções de Resposta

Opções	Respostas
R1	Concordo com a Dívida Técnica, causas e indicadores
R2	Concordo com a Dívida Técnica e com as causas, mas não concordo com os indicadores
R3	Concordo com a Dívida Técnica e com os indicadores, mas não concordo com as causas
R4	Concordo com as causas e os indicadores, mas não concordo com a Dívida Técnica
R5	Não concordo com a Dívida Técnica, causas e indicadores
R6	Concordo com a Dívida Técnica, mas não concordo com as causas e indicadores

$$n = \frac{N.\frac{1}{E_0^2}}{N + \frac{1}{E_0^2}} \rightarrow 64 = \frac{248.\frac{1}{E_0^2}}{248 + \frac{1}{E_0^2}} \rightarrow E_0 = \sqrt{\frac{248 - 64}{248 * 64}} \rightarrow E_0 = 0{,}107 \rightarrow Confiança = 89{,}3\%$$

Onde: N = tamanho da população | n = tamanho da amostra | E_0 = nível de confiança (ex: 0,05 → 95%)

Fig. 1: Cálculo do Nível de Confiança de uma amostra baseado em [28]

com o survey.

IV. RESULTADOS

A. PERÍODO E TOTAL DE PARTICIPANTES DO SURVEY

Quanto ao período de realização da coleta de dados, o survey deste artigo é de corte transversal, pois os dados foram coletados em apenas um momento, do dia 01/06/2015 a 22/06/2015, no endereço http://icomp.ufam.edu.br/experts/survey/. A população de estudo foi focada em profissionais da área de Teste de Software que trabalham em empresas brasileiras centradas em desenvolvimento de software.

Foram enviados um total de 248 e-mails, destes convites um total de 64 participantes finalizaram a pesquisa, em que 19 responderam o questionário Planejar e Concluir e 45 responderam o questionário Projetar e Executar. Para calcular o nível de confiança da amostra deste estudo, utilizou-se a fórmula descrita na Figura 1. Sendo o tamanho da população de 248 pessoas e tamanho da amostra de 64 participantes, isso resulta em 89,3% de nível confiança para a amostra obtida.

B. ANÁLISE DA CARACTERIZAÇÃO DOS PARTICIPANTES

Na Figura 2a, o número de participantes com experiência acima de 5 anos em teste de software foi de 51,6%. A pesquisa contou ainda com um número

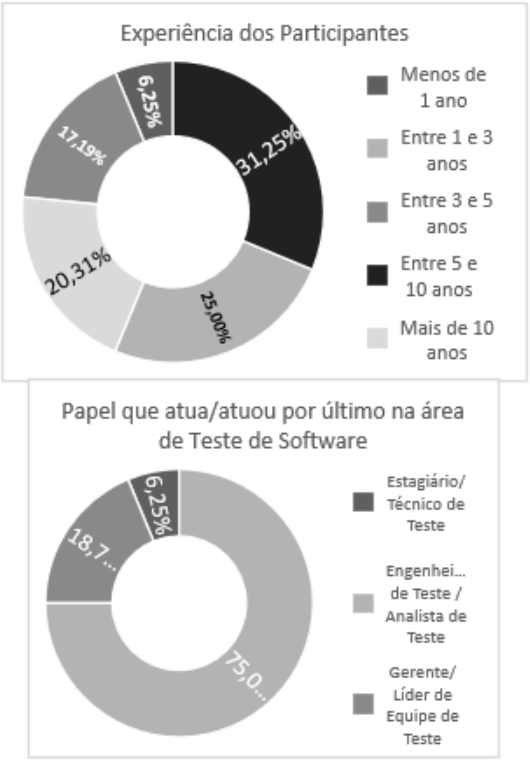


Fig. 2: (a) Distribuição de Experiência dos Participantes. (b) Papel da Área de Testes.

relevante de participantes com experiência entre 3 e 5 anos (17,2%). Por fim, o fato de 31,3% dos participantes terem menos 3 anos de experiência ajudou para que a pesquisa apresente uma variação de experiência nas respostas.

A Figura 2b apresenta a distribuição dos papéis que os participantes atuam/atuaram por último na área de teste de software. É possível observar que a maioria dos participantes atua/atuou como Engenheiros/Analistas de Testes (75%), seguido por Gerentes/Líderes de testes (18,8%) e, finalmente, estagiários/técnicos em teste (6,3%).

C. ANÁLISE DO RESULTADO DO SURVEY

Para análise dos dados gerados a partir da aplicação do survey, foi realizada uma análise quantitativa a partir de estatística descritiva.

A Tabela 4 apresenta os resultados das DTs presentes nos questionários, divididos por questionário: DT01 a DT09 refere-se ao questionário de Planejar e Concluir, no qual foram obtidas 19 respostas, enquanto que DT10 a DT22 são as respostas do questionário de Projetar e Executar, que contou com 45 participantes. Nessa tabela, cada linha representa um grupo de DT, causas e indicadores analisado

no estudo. As colunas de R1 à R6 representam a contagem das possíveis opções de resposta que os usuários escolheram para cada uma das DT. As três últimas colunas condizem com o número de participantes que concordaram com os seguintes itens:

- Dívidas: número participantes que concordaram com o conceito que define a DT analisada. Essa opção consiste na soma das colunas R1+R2+R3, pois todas estas opções estão associadas à concordância com a DT.
- Causas: número participantes que concordaram com causas relacionadas à DT que está sendo analisada. Essa opção consiste na soma das colunas R1+R2+R4, pois todas estas opções estão associadas à concordância com as causas.
- Indicadores: número participantes que concordaram com os indicadores relacionados à DT que está sendo analisada. Essa opção consiste na soma das colunas R1+R3+R4, pois todas estas opções estão associadas à concordância com os indicadores que relacionados à DT.

Por exemplo, para a DT1 (Cronograma de teste não definido/inadequado), a opção de resposta R1 (Concorda com a Dívida Técnica causas e indicadores) foi escolhida 8 vezes. Neste exemplo, 15 pessoas, do total de 19, concordaram com os indicadores da primeira Dívida Técnica.

Nas próximas seções serão apresentadas as análises referentes às DTs, causas e indicadores, separando-as por tipo de questionário aplicado ("Planejar e Concluir" e "Projetar e Executar").

D. DÍVIDA TÉCNICA, CAUSAS E INDICADORES DO QUESTIONÁRIO PLANEJAR E CONCLUIR

A Figura 3 exibe o percentual de concordância referente a DTs, causas e indicadores do questionário Planejar e Concluir, pela ordem decrescente de concordância que cada item obteve. A DT que obteve maior concordância foi "Riscos de teste não definidos/inadequados" (DT3) e a que teve menor grau de concordância foi a DT7 "Bugs sem correções", com uma diferença considerável para a DT com maior concordância. As demais DTs, apresentaram uma variação de concordância entre 70% e 90%. Entre todos os comentários realizados pelos participantes do questionário Planejar e Concluir, a maioria foi direcionada a opiniões referentes às causas e indicadores. Os poucos comentários referentes às DTs apresentam discordância da DT em algum contexto. Por exemplo, um participante comentou sua discordância quanto à DT1 (Cronograma de teste não definido/inadequado) no contexto de

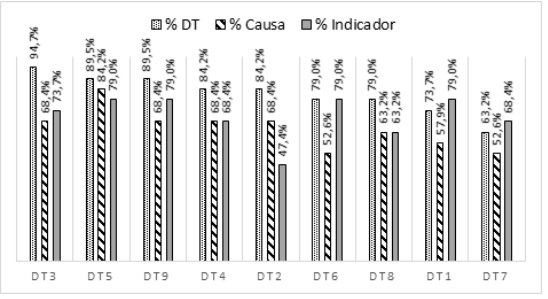


Fig. 3: Análise de DTs, Causas e Indicadores do Questionário "Planejar e Concluir Testes".

sua aplicação em projetos ágeis: "No processo que testes é considerado no final, posso concordar, mas em uma metodologia ágil não faz sentido um cronograma de testes e sim um cronograma do time como um todo". A Figura 3 ainda apresenta a análise de concordância das causas de cada DT. As causas, diferentemente das DTs, apresentaram porcentagens de grau de concordância menores, ficando a maioria, com exceção de uma, abaixo de 69%.

O fato de as causas analisadas terem apresentado um menor grau de concordância não significa que os participantes tenham discordado completamente delas. Pode-se observar por alguns comentários dos participantes que em determinados casos a discordância das causas ocorreu por acharem que mais de uma causa da mesma DT tinham o mesmo significado e deveriam ser unificadas. Em outros comentários, os participantes sugeriram que o conjunto das causas da DT estavam incompletas e era necessário acrescentar novas causas. Por exemplo: um participante sugeriu acrescentar uma causa: "Outra causa: Falta de automação, teste unitário na build, etc.", comentário referente à DT6 (Fornecer um entregável com bugs).

Por fim, pode observar a partir da Figura 3 que os indicadores, assim como as causas, tiveram graus de concordância menores que as DTs. O indicador que obteve o grau de concordância mais baixo (47,4%) apresenta o menor grau de concordância de toda a análise. Este indicador está relacionado à DT2 (Ferramentas de teste não utilizadas/aplicáveis no projeto), com os indicadores: quantidade de recurso disponível menor que os valores das ferramentas; diversidade de ferramentas disponíveis menor do que o esperado para a cobertura dos testes. Alguns dos comentários referentes à DT2 indicaram que existem muitas ferramentas sem custo e o fator financeiro não seria um indicador. Isto pode refletir a baixa concordância para os indicadores da DT2.

Como ocorreu com as causas, os participantes

Tab. 4: Resumos das Respostas Obtidas por Dívida Técnica Avaliada.

	Itens Analisados	R1	R2	R3	R4	R5	R6	Dívidas	Causas	Indicadores
Planejar e Concluir	DT01	8	1	5	2	3	0	14	11	15
	DT02	7	6	2	0	3	1	16	13	9
	DT03	11	2	3	0	1	2	18	13	14
	DT04	10	2	2	1	2	2	16	13	13
	DT05	14	2	1	0	2	0	17	16	15
	DT06	8	1	6	1	3	0	15	10	15
	DT07	7	1	4	2	5	0	12	10	13
	DT08	9	2	2	1	3	2	15	12	12
	DT09	11	2	4	0	2	0	17	13	15
Projetar e Executar	DT10	29	8	6	1	1	0	43	38	36
	DT11	27	6	9	1	0	2	44	34	37
	DT12	16	10	6	2	9	1	33	28	24
	DT13	29	9	4	1	1	1	43	39	34
	DT14	29	4	7	0	3	2	42	33	36
	DT15	33	5	1	1	3	1	40	39	35
	DT16	33	2	7	1	1	0	42	36	41
	DT17	32	2	7	2	2	0	41	36	41
	DT18	30	4	5	1	0	1	40	35	36
	DT19	35	2	6	0	0	1	44	37	41
	DT20	35	6	1	0	1	1	43	41	36
	DT21	32	1	9	0	1	1	43	33	41
	DT22	34	1	7	0	1	2	44	35	41

não necessariamente discordavam completamente dos indicadores. Eles concordavam em partes, pediam para acrescentar mais indicadores ou retirar algum deles. Dentre os comentários, um participante acrescentou na DT8 (Falta de controle de quais tipos de testes estão sendo executados): "Acredito que temos muitos indicadores aí: progressão de testes, planejados X executados, critérios de saída de testes, testes por tipo, entre outros para a DT".

E. DÍVIDA TÉCNICA, CAUSAS E INDICADORES DO QUESTIONÁRIO PROJETAR E EXECUTAR

A Figura 4 exibe o percentual de concordância referente a DTs, causas e indicadores do questionário Projetar e Executar, pela ordem de concordância que cada item obteve. A partir da Figura 4 pode-se observar que as DTs tiveram em sua maioria mais 90% de aprovação. Isto demostra a alto nível de concordância adquirida para a maioria das DTs deste questionário.

Assim como no questionário anterior, alguns participantes realizaram comentários quanto as DTs sugerindo, por exemplo, agrupar uma DT com outra (sugerindo que elas apresentam mesmo significado). Também houve sugestões para alterar a nomenclatura de determinada DT. Isto ocorreu, por exemplo, com a DT13 (Necessidade de refatorar os scripts de teste), em que um participante comentou: "Dívida Técnica: creio que pode ser alterado para Scripts de Teste Obsoletos".

A Figura 4 apresenta ainda a análise das Causas. As causas apresentaram uma maior variação de concordância, entre 62% e 91%, diferente das DTs que

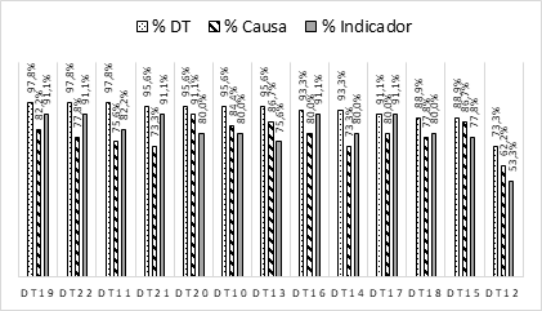


Fig. 4: Análise de DTs, Causas e Indicadores do Questionário "Projetar e Executar Testes".

apresentaram níveis de concordância com menor variação e maiores porcentagens. As causas deste questionário, assim como as do questionário anterior, apresentaram discordâncias que refletem à necessidade de acrescentar novas causas, ou ainda, retirar uma ou mais causas sugeridas. Por exemplo, referente as causas da DT18 (Sobrecarga de atividades em determinados períodos do projeto) um participante sugeriu a inclusão de uma nova causa: "Causas: Inexperiência deve ser incluído".

A Figura 4 apresenta também a concordância dos indicadores das DTs do questionário Projetar e Executar. Pode-se observar que os indicadores não apresentaram muita variação na porcentagem de concordância e mantiveram a maioria dos grupos de indicadores com aceite acima de 80%.

O grupo de indicadores que tiveram menor aprovação (53,33%, mais de 20% de distância do penúltimo grupo de indicadores) é associado à

DT12 (Falta de documentação dos testes que serão executados). O indicador da DT12 é: Quantidade de documentos. Alguns dos comentários referentes à DT12 indicaram que a quantidade de documentos pode não ser uma métrica adequada como indicador, vários participantes citam que a qualidade dos documentos reflete diretamente na eficiência dos mesmos, e seria um indicador mais adequado, isto pode refletir a baixa concordância para o indicador da DT12.

Os comentários sobre os indicadores foram em menor número, porém, manteve o mesmo padrão observado nos comentários referentes as causas, isto é, acrescentando novos indicadores, refutando alguns, ou ainda discordando de um ou outro indicador. Na próxima seção são apresentadas as conclusões e trabalhos futuros gerados a partir da execução desta pesquisa.

V. CONCLUSÕES

DT é um tema recente em Engenharia de Software, e apresenta a maioria dos trabalhos com ênfase na fase de codificação. Assim, são poucos os trabalhos relacionados diretamente a DT na área de Teste de Software. Este artigo buscou mapear possíveis DTs no processo de testes, com objetivo de servir como instrumento para facilitar a identificação das DTs e ajudar no entendimento de suas possíveis origens. Para isto, no mapeamento proposto foram relacionados indicadores e causas à cada DT. Os indicadores se apresentaram como recursos interessantes para que o gestor possa assumir medidas preditivas ao identificar um indicador eminente em uma DT. Já as causas ajudam no mapeamento da origem da DT facilitando a identificação de possíveis soluções. O artigo apresentou um survey no qual profissionais da área concordaram com as DTs, que foram retiradas de problemas que acontecem em teste de software. O trabalho apresentou 22 DTs com suas causas e indicadores. Como limitação, vale ressaltar que podem existir outros itens (DT, causa ou identificador) que não foram identificados para serem avaliados. Como trabalhos futuros pretende-se identificar possíveis soluções (a partir da literatura técnica) para cada uma das 22 DTs reportadas neste trabalho, avaliando-as com profissionais da área de teste de software, e, por fim, propor um mapa de apoio a gestão de DTs no processo de teste de software.

REFERÊNCIAS

[1] 1. R. S. Pressman. Software Engineering: a Practitioner's Approach, McGraw-Hill, 7 ed. 2011.

[2] W. Cunningham. The Wycash Portfolio Management System, In: ACM SIGPLAN OOPS Messenger (Vol. 4, No. 2). ACM. December 1992, p. 29-30.

[3] Z. Li, P. Avgeriou, P. Liang. A Systematic Mapping Stydy on Technical Debt and its Management. In: Journal Of Systems and Software. v. 101. Março de 2015. p. 1 – 272.

[4] A. Bertolino. Software Testing Research: Achievements, Challenges, Dreams. In: Future of Software Engineering, 2007. FOSE '07. 2007, Minneapolis, MN. 23 - 25 de mai. de 2007. p. 85–103.

[5] ISO/IEC/IEEE 29119-2. Software and Systems Engineering Software Testing - Part 2: Test Processes, International Organization of Standardization, 2013.

[6] T. Klinger. An Enterprise Perspective on Technical Debt. In: International Conference on Software Engineering. 2011, New York, NY, USA. ACM, 2011. p. 35–38.

[7] Y. Guo. Measuring and Monitoring Technical Debt. 4th International Doctoral Symposium on Empirical Software Engineering. [S.I.]:[s.n.].2009 p. 25-46.

[8] F. Shull. Perfectionists in a World of Finite Resources: IEEE Software, 2011, 28, p. 4 - 6

[9] N.S.R. Alves, L. F. Ribeiro, V. Caires, T. S. Mendes, R .O. Spínola. Towards an Ontology of Terms on Technical Debt. In: International Workshop on Managing Technical Debt, 6, 2014. p. 1-7

[10] A. Mettle, J. Hass. Testing Processes, In: Software Testing Verification and Validation Workshop, 2008. ICSTW '08. IEEE International Conference on, 2008 Lillehammer, Norway, Anais, Lillehammer, Norway, 9-11 de Abril de 2008, p. 321 - 327

[11] S. Shah; Torchiano, M.; Vetro, A.; Morisio, M.. Exploratory Testing as a Source of Testing Technical Debt. IT Professional, v. 16, ed. 3. p. 44 - 51. 7 de mar. de 2014

[12] R. Bavani. Distributed Agile: Agile Testing and Technical Debt. IEEE Software 29. Jun. de 2012. p. 28 – 33.

[13] K. Wiklund; Eldh, S.; Sundmark, Daniel; Lundqvist, K. et al. Tehnical Debt in Test Automation. In: Software Testing, Verification and Validation (ICST), 2012 IEEE Fifth International Conference on. 2012. p. 887 – 892.

[14] W. Snipes, B. Robinson, Y. Guo. C. Seaman. Defining the Decision Factors for Managing Defects: A Technical Debt Perspective, In: Proceedings of the 3rd International Workshop on Managing Technical Debt (MTD'12), IEEE, Zurich, Switzerland, 2012, p. 54 – 60.

[15] E. Dustin. Effective Software Testing: 50 Ways to Improve Your Software Testing. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc. 2003.

[16] K. Naik, P. Tripathy. Software Testing and Quality Assurance: Theory and Practice. Hoboken, New Jersey: John Wiley Sons, Inc. 2008. 616 p.

[17] W. E. Perry. Effective Methods for Software Testing: Includes Complete Guidelines, Checklists, and Templates. 3 ed. Indianapolis, Indiana: Wiley Publishing, Inc. 2006. 973 p.

[18] J. Kasurinen, O. Taipale, K. Smolander. Analysis of Problems in Testing Practices, In: Software Engineering Conference, 2009. APSEC '09. Asia-Pacific 2009, Batu Ferringhi, Penang, Malaysia. 1 - 3 de Dezembro de 2009, p. 309 – 315.

[19] S. Amland. Risk-based testing: risk analysis fundamentals and metrics for software testing including a financial application case study. New York, NY, USA: Elsevier Science Inc. ed. 3, 2000. vol. 53.

[20] S. M. Quadri, S. U. Farooq. Software Testing: Goals, Principles, and Limitations. International Journal of Computer Applications, v. 6, n. 9, p. 7 - 10, 2010.

[21] J. W. Rittinghouse. Managing Software Deliverabels: A Software Development Management Methodology. Digital Press, 2004. p. 111–133.

[22] D. Lorenzoli, L. Mariani, M. Pezze. Automatic generation of software behavioral models. In: International Conference on Software Engineering, 30, 2008. p. 501-510.

[23] G. J. Myers, T. Badgett, C. Sandler. The Art of Software Testing. ed. 3. Canadá: John Wiley Sons, Inc, 2012. 240 p.
[24] A. I. BaarsT. E. J. Vos, D. M. Dimitrov. Using Evolutionary Testing to Find Test Scenarios for Hard to Reproduce Faults, In: Third International Conference on Software Testing, Verification, and Validation Workshops (ICSTW), 2010, Paris, França, p. 173 – 181.
[25] B. Jiang; Zhenyu Zhang; Tse, T.H.; Chen, T.Y. et al. How well does test case prioritization integrate with statistical fault localization?. Information and
[26] Software Technology, v 54, ed. 7, p. 739 - 758, 2012.
[27] E. Aranha, P. Borba. An Estimation Model for Test Execution Effort, In: Empirical Software Engineering and Measurement, 2007. ESEM 2007. First International Symposium on, 2007, Madri, 2007, 20 - 21 de Setembro de 2007, p. 107 – 116.
[28] M. Hamburg. Basic Statistics: A Modern Approach, Journal of the Royal Statistical Society. Series A (General), v. 146, no. 1, ed. 2.

CLEYDIANE LIMA DE SOUSA Mestrado pela Universidade Federal de Pernambuco (2016), com o tema de dissertação: Dívida Técnica no Processo de Teste de Software. Graduação em Sistemas de Informação pelo Centro Universitário Luterano de Palmas (2012) Trabalha como Engenheira de Testes do Centro de Estudos e Sistemas Avançados do Recife (CESAR). Casada com automação de testes de contrato, API, componentes e e2e. Brinca com testes de carga no Artillery. Ama tartarugas. Maranhense morando em Recife. Estudante de fotografia nas horas vagas.

ARILO CLAUDIO DIAS NETO Possui graduação em Ciência da Computação pela Universidade Federal do Amazonas (2004), mestrado em Engenharia de Sistemas e Computação pela Universidade Federal do Rio de Janeiro (2006), doutorado em Engenharia de Sistemas e Computação pela Universidade Federal do Rio de Janeiro (2009). É Professor Adjunto da Universidade Federal do Amazonas, coordenador do Grupo de Pesquisa de Experimentação e Teste de Software. Atua desde 2017 como CPO (Chief Product Officer) do Méliuz.

Publicação: Outubro de 2019

Digital Object Identifier https://doi.org/10.33911/singular-etg.v1i2.63

Plataforma de visualização dos dados minerados do ENADE dos cursos de Computação nos anos de 2008 a 2014

RENATO MARINHO ALVES[1], ALEXANDRE MORAES MATOS[2], EDNA DIAS CANEDO[3].

1- Departamento de Computação - Unicamp - Av. Albert Einstein, 1251 – Campinas – São Paulo, Brasil 2 - Urpay Brasil – Palmas – Tocantins, Brasil 3- Departamento de Ciência da Computação - Universidade de Brasília, Caixa Postal 4466, 70910-900, Brasília–DF, Brasil..
(e-mail:renato.mar.alves@gmail.com, alexandremt03@gmail.com, edna.canedo@gmail.com)
Autor Correspondente: Renato Marinho Alves (e-mail: renato.mar.alves@gmail.com).

RESUMO - O conceito de mineração e análise de dados se baseia na utilização de algoritmos computacionais e técnicas de exploração de conteúdo em um conjunto de dados de modo a se obter maiores informações sobre estes que possibilitem ao usuário ter um maior entendimento sobre os dados e a fonte das informações. Desta forma, foi criada área de ciência dos dados, do inglês *data science*, que é atualmente empregada a utilização de técnicas científicas da computação visando a obtenção de informações implícitas em conjuntos de dados. Este trabalho apresenta a aplicação de técnicas de mineração e análise de dados seguindo o modelo CRISP-DM sobre os dados de respostas dos estudantes da área de Computação às provas do ENADE. Diante disso é apresentada uma plataforma para apresentação dos resultados obtidos da aplicação das técnicas.

PALAVRAS-CHAVE - Mineração de dados, Dados Abertos, ENADE

I. INTRODUÇÃO

Atualmente no Brasil tem-se o Exame Nacional de Desempenho de Estudantes (ENADE) como forma de analisar a qualidade do ensino superior das universidades do Brasil. Este exame avalia os cursos superiores com base nas notas obtidas pelos estudantes na prova correspondente a sua área de atuação. Segundo o Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira (Inep) [7] o "ENADE avalia o rendimento de concluintes dos cursos de graduação, em relação aos conteúdos programáticos, habilidade e competências adquiridas em sua formação".

Tal exame busca avaliar o desempenho e conhecimento dos estudantes em função dos conteúdos programáticos previstos nas diretrizes curriculares estabelecidas em seus cursos, o desenvolvimento de competências e habilidades necessárias ao aprofundamento da formação acadêmica, geral e profissional, além de constatar o nível de atualização dos estudantes com relação a realidade brasileira e mundial (Inep, 2008) [7].

Diante disso, um grupo seleto de cursos de graduação realizam o Enade, de forma que a aplicação do exame é efetuada em ciclos de três anos, sendo os participantes os acadêmicos de cada curso. Parte dos acadêmicos são selecionados para o processo e tem presença obrigatória, sendo uma condição imprescindível para a emissão do histórico escolar de conclusão de curso.

A prova do ENADE é dividida em três etapas, sendo estas: o Questionário do Estudante, a Prova e o Questionário da Coordenação do Cursos. Os dados referentes à prova são disponibilizados de forma pública e gratuita no portal do Inep. Este trabalho apresenta parte dos resultados da mineração dos dados coletados, organizados e disponibilizados pelo Inep acerca das provas realizadas pelos acadêmicos

dos cursos da área de Computação, devido ao fato que a análise se aprofunda nas demais áreas da computação para um dos ano já citados. Assim, os resultados completos podem ser vistos no endereço online da plataforma de visualização dos resultados coletados.

A mineração dos dados tem por o objetivo geral de avaliar e analisar o desempenho destes estudantes nos anos de 2008, 2011 e 2014, que são os últimos 2 anos com dados divulgados para as provas da área de Computação. A escolha da mineração de dados se deve ao fato que esta surge como uma solução para extração de informações de bases de dados numerosas de forma rápida e eficaz por meio de suas técnicas, tendo em vista que o processo manual demanda muito tempo [10]. De modo a direcionar os esforços foi utilizado o modelo de referência CRISP-DM [3] para orientação das etapas que aplicarão os algoritmos de classificação e clusterização.

O presente trabalho está estruturado da seguinte forma: na Seção 2, é descrito o referencial teórico necessário para embasar a realização da pesquisa. Na Seção 3 é apresentada a sequência das etapas realizada no desenvolvimento deste trabalho. Na Seção 5 são descritos alguns dos resultados obtidos neste projeto. Por fim, Na Seção 6 é apresentado as considerações finais sobre o trabalho desenvolvido até o momento e a proposta de trabalhos futuros.

II. REFERENCIAL TEÓRICO

A Mineração de Dados é uma etapa de extrema importância para que haja o descoberta de informações, tendo em vista que há uma grande quantidade de informações potenciais que podem ser obtidas a partir de análises mais profundas dos dados. Da Silva [5] conceitua a mineração como "o esforço para descoberta de padrões em bases de dados".

De acordo com Cortês [4] a mineração de dados é classificada com uma combinação entre pesquisas em estatística, inteligência artificial e bancos de dados que vem emergindo como uma área de grande importância que destaca-se em diversos congressos científicos e produtos comerciais.

A Mineração de Dados se fundamenta na utilização de meios automáticos para a busca e descoberta de padrões em grandes bases de dados. Diante disso são propostos algoritmos computacionais que implementam lógicas de extração e análise de informações [13].

Os algoritmos implementados para Mineração de Dados geralmente utilizam conceitos de Inteligência Artificial (IA) e Aprendizado de Máquina. Os métodos de DM são divididos em aprendizado supervisionado (preditivo) e não-supervisionado (descritivo) [2]. De modo que o aprendizado supervisionado exige o acompanhamento constante do analista de dados no processo de mineração, enquanto o processo não-supervisionado propõe que são necessários apenas os dados de entrada para se gerar uma saída.

Dentre os diversos algoritmos de Mineração de Dados e seus usos, os mais relevantes são os que utilizam técnicas de Associação, Classificação e Clusterização de dados. Cada algoritmo possui uma lógica de análise e resultados distintos que podem ser aplicados em diversos contextos, conforme descrito a seguir:

- Associação: uma das maiores técnicas de data mining que é comumente utilizada para a descoberta de padrões de forma não supervisionada [8]. Este método busca em grandes bases de dados aspectos que ajudem a compreender os padrões. Os algoritmos executados buscam relações entre os dados, de modo que são verificados os eventos que ocorrem de forma concorrente a fim de se alcançar melhores resultados [11].
- Classificação: consiste na definição de classes a partir de informações recorrentes entre os dados analisados. Conforme Bartolomeu [1] é uma das tarefas mais comuns da Mineração de Dados, consistindo da localização de propriedades comuns entre um conjunto de dados em uma base e classificação desses dados em classes pré-definidas, seguindo o modelo estipulado.
- Agrupamento (Clusterização): no agrupamento os dados analisados são separados em subgrupos ou clusters. O objetivo desse método é formar grupos baseados no princípio de que esses grupos devem ser o mais homogêneo em si e mais heterogêneo entre si [4]. A diferença entre a aplicação do agrupamento e da classificação está no fato de que o algoritmo de agrupamento não utiliza classes predefinidas para agrupar os dados analisados, sendo agrupados com base em similaridades.

De maneira a otimizar o processo mineração de dados, existem modelos de referência que propõem estruturar uma série de passos a serem seguidos para a obtenção de um melhor e mais rápido resultado, como o CRISP-DM (Cross Industry Standard Process for Data Mining) [3]. Este modelo de referência consiste em um conjunto de etapas que buscam aumentar a taxa de sucesso de processos de Mineração de Dados.

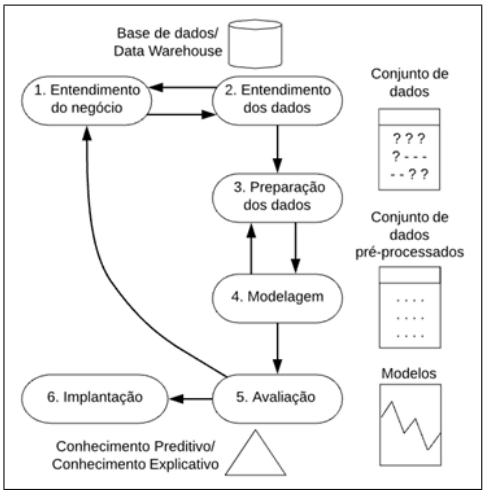


Fig. 1: Modelo de processos do CRISP-DM, adaptado de [9].

A Figura 1 demonstra como é a estruturação das fases do CRISP-DM, sendo que o uso das fases em conjunto é considerado essencial para o correto funcionamento do modelo. A breve descrição dessas fases é apresentada a seguir.

- Entendimento do negócio: Etapa focada no entendimento do objetivo a ser atingido ao se usar a Mineração de Dados. Sendo assim uma etapa fundamental para o desenvolvimento das demais etapas.
- Entendimento dos dados: Consiste em compreender os dados que estão sendo analisados de forma a identificar o conjunto de dados relevante à proposta.As fontes fornecedoras dos dados podem vir de diversos locais e possuírem diversos formatos [2].
- Preparação dos dados: Esta etapa consiste na formatação e transformação dos dados de modo a padronizá-los. O propósito dessa fase é limpar os dados selecionados de modo a obter-se melhor qualidade, tendo em vista que alguns dos dados selecionados podem seguir diferentes padrões por conta de serem coletados de diferentes fontes [10].
- Modelagem: Na etapa de modelagem são aplicados os algoritmos de mineração de dados de modo que gerar os resultados esperados.
- Avaliação: "Considerada uma fase crítica do processo de mineração, nesta etapa é necessária a participação de especialistas nos dados, conhecedores do negócio e tomadores de decisão" [2]. Para apoiar nessa etapa são utilizados gráficos para analisar e visualizar os resultados obtidos nas etapas anteriores. Para garantir a confiabilidade dos modelos é indicada a realização de testes e validações nos modelos construídos.
- Implementação: Nesta etapa os resultados do projeto de DM são apresentados aos envolvidos. "O estudo de data mining possui novos conhecimentos descobertos, que necessitam de estar bem atados aos objetivos originais do projeto de data mining" [10].

A. CLUSTERIZAÇÃO

No agrupamento os dados analisados são separados em subgrupos ou clusters. "Seu objetivo é formar grupos baseados no princípio de que esses grupos devem ser o mais homogêneo em si e mais heterogêneo entre si" [4]. A diferença entre a aplicação do agrupamento e da classificação está no fato de que o algoritmo de agrupamento não utiliza classes predefinidas para agrupar os dados analisados, sendo agrupados com base em similaridades. "Na segmentação não há classes nem exemplos predefinidos. Os registros são agrupados de acordo com a semelhança, e a partir daí o significado será determinado" [1].

A análise clusterizada propõe a execução do algoritmo sobre dados não agrupados e utiliza técnicas automatizadas para colocar os dados em grupos [10]. Por este fato e por não requerer conjuntos de treinamentos para o seu funcionamento a Clusterização é considerada uma técnica de mineração de dados não supervisionada. Kantardzic [8] define a análise clusterizada a clusterização propõe o estudo formal de métodos e algoritmos para agrupamentos naturais de objetos de acordo com métricas, características intrínsecas ou similaridades percebidas nos dados.

Ainda segundo Kantardzic [8], a clusterização é observada como um conjunto de metodologias para classificação automática de amostras em grupos utilizando de métricas de associação de forma que estas amostras agrupadas em um grupo sejam similares, enquanto amostras pertencentes a grupos distintos são diferentes. Olson e Delen [10] afirmam que a clusterização compartilha uma área metodologia comum a classificação, em que certa parte dos modelos matemáticos recomendados para a análise classificativa podem ser utilizados para a análise clusterizada.

A partir dos dados de entrada a aplicação do algoritmo de clusterização gera amostras de clusters com base nas similaridades encontradas.

A análise clusterizada pode ser aplicada em di-

versas áreas de estudo com a finalidade de agrupar dados com base em similaridades destes, como área empresarial, educação, saúde, engenharias, entre outras.

B. TRABALHOS RELACIONADOS

Técnicas de mineração de dados aplicadas aos microdados do ENADE para avaliar o desempenho dos acadêmicos do curso de Ciência da Computação no Rio Grande do Sul utilizando o software.

Com o uso de *Hierarchical Clustering* (Agrupamento Hierárquico) através da linguagem de programação R, [14] realizaram a mineração dos microdados do ENADE para avaliação do desempenho dos acadêmicos dos cursos de Ciência da Computação no estado do Rio Grande do Sul. Os resultados alcançados possibilitaram a identificação das instituições com bons e ruins desempenhos acadêmicos, promovendo o apoio na tomada de decisão no que tange a melhoria do ensino superior brasileiro.

Os primeiros resultados obtidos, ainda parciais, foi um levantamento dos dados das variáveis do ENADE 2014, tal levantamento apresentava informações como número mínimo e máximo de inscritos para a realização da prova, número de participantes e o Conceito ENADE das universidades. A partir disto, foi realizada a tarefa de agrupamento em cima dos dados, atividade na qual resultou na geração de quatro grupos de IES.

- Grupo 1: Composto por nove IES, onde seis obtiveram Conceito ENADE 3, e outras 3 obtiveram Conceito ENADE 4. O Conceito ENADE Contínuo apresentou 2,86 de média.
- Grupo 2: Composto por dez instituições, das quais sete obtiveram Conceito ENADE 2 e somente duas obtiveram Conceito 3. A média do Conceito ENADE Contínuo foi de 1,74.
- Grupo 3: Este grupo é constituído por apenas uma instituição, a mesma obteve Conceito ENADE 1 e o Conceito ENADE Contínuo foi de 0,43.
- Grupo 4: Composto por três IES, 2 obtiveram Conceito ENADE 4 e apenas uma obteve Conceito ENADE 5. A média do Conceito ENADE Contínuo foi de 5,53.

Com os grupos acima pode-se observar que as instituições contidas no grupo quatro foram as que obtiveram melhores resultados no ENADE no ano de 2014, pois seus Conceitos ENADE Contínuo foi o mais alto na escala de 0 a 5, as IES do grupo 1 ficam em segundo lugar, também apresentando bons resultados. A IES presente no grupo 1 ficou isolada em seu *cluster*, pois foi a que apresentou pior resultado.

Prática de Mineração de Dados no Exame Nacional do Ensino Médio

Utilizando a tarefa de associação, [12] através da aplicação do algoritmo de Apriori na tentativa de encontrar padrões nos resultados de provas e questionários socioeconômicos do Exame Nacional do Ensino Médio (ENEM). Os dados do trabalho foram coletados diretamente no site do Instituto Nacional de Estudos e Pesquisas Educacionais (INEP), que os dispões de forma aberta.

A primeira etapa do trabalho foi definir o foco regional da pesquisa, na qual foram selecionados dados somente das capitais da região Sudeste do país, com um total de 452.710 alunos, fez-se necessário também a eliminação dos registros dos alunos que não compareceram nos dois dias de prova, um total aproximado de 310 mil pessoas.

As perguntas selecionadas para a pesquisa foram voltadas para a quantidade de membros na família, escolaridade da mãe, renda familiar e em que tipo de escola o aluno estudou durante o Ensino Médio. Tais perguntas foram escolhidas com o intuito de saber se alguma delas influenciam na nota e no desempenho do aluno na prova do ENEM.

Os resultados obtidos apontaram que, renda familiar baixa, escolaridade em nível primário dos pais e um alto número de pessoas morando com o estudante são possíveis influencias negativas no desempenho do aluno. Os concluem que o ensino público no Brasil precisa de melhorias, tanto politica quanto pedagogicamente, onde a classe social mais baixa é afetada, o que exerce influência direta no desempenho do estudante.

Aplicação de Técnicas de Mineração de Dados no Processo de Aprendizagem na Educação à Distância

[6] aplicaram técnicas de Mineração de Dados na busca de descobrir informações relevantes sobre o perfil do aluno com relação à utilização do modelo de ensino-aprendizagem à distância.

O trabalho do tipo exploratório foi realizado a partir da aplicação das técnicas de Árvore de Decisão e Redes Bayesianas. Foram analisados um total de 272 registros de diferentes alunos do Lab-SQL, um ambiente interativo que auxilia alunos no aprendizado da linguagem SQL, sendo útil para o mediador na realização automática de avaliações.

Os resultados obtidos através de Redes Bayesianas indicaram que quanto mais cedo o aluno inicia o estudo da disciplina através do ambiente, maior desempenho o mesmo tem na resolução de exercícios e

também possui maior conhecimento das funcionalidades da ferramenta do que alunos atrasados.

Os resultados da Árvore de Decisão apontaram que 95% dos alunos do curso de Sistemas de Informação que resolvem problemas com nível de dificuldade com média de 1,2 (em uma escala de 1 a 3) possuem média de pontos acima da média geral da turma que pertence. Também foi possível perceber nos resultados desta técnica que alunos com maior interesse e que buscam iniciar o uso da ferramenta mais cedo, mostraram ter um desempenho maior no que tange à pontuação dos exercícios aplicados.

III. MÉTODO PROPOSTO

Para o desenvolvimento deste trabalho foi utilizado como base o modelo de referência CRISP-DM para otimizar o processo de mineração de dados e alcançar resultados para análise.

Durante o projeto foram considerados os estudantes de Bacharelado dos cursos de Ciência da Computação, Sistemas de Informação e Engenharia da Computação das Instituições de Ensino Superior (IES) brasileiras. Essa amostra dos dados foi escolhida devido ao fato deste conjunto de discentes estar presente em todas as aplicações das provas dentre os anos mencionados e por se tratar do grupo completo de cursos da área de Tecnologia da Informação que participam do ENADE.

As adequações feitas ao modelo para o desenvolvimento deste trabalho são representadas pela Figura 2 e serão abordadas de forma mais detalhada nos tópicos a seguir.

Fig. 2: Fluxograma do desenvolvimento da proposta

- Definição da proposta: a primeira etapa deste trabalho consistiu na definição da proposta, englobando quais dados serão utilizados, quais algoritmos podem ser interessantes e estruturando a forma que estes dados podem ser apresentados. Para a análise do desempenho dos alunos, pretende-se classificar as perguntas da prova do Enade dentro das principais áreas da Computação. Para isso, a área foi dividida inicialmente de acordo com as grandes áreas existentes no modelo de mapa curricular dos cursos de Computação (CC e SI) disponibilizado pelo CEULP/ULBRA. Nesse modelo, a partir de 6 grandes áreas é possível encaixar as áreas de enquadramentos de cada questão da prova do ENADE nas grandes áreas da Computação. Para este trabalho, serão focadas em apenas 4 das 6 grandes áreas, isso devido a existirem áreas com poucas questões aplicadas nas provas, sendo assim estas agrupadas em áreas com uma similaridade. A Tabela 1 apresenta essa classificação.

Tab. 1: Áreas da computação.

Grandes áreas	**Áreas para enquadramento das questões**
Análise e Desenvolvimento	1. Engenharia de Software
	2. Banco de Dados
	3. Qualidade de Software
	4. Arquitetura de Software
Fundamentos das Ciências exatas	5. Lógica
Fundamentos da Computação	6. Arquitetura de Computadores
	7. Estrutura de Dados
	8. Sistemas Operacionais
Tecnologias da Computação	9. Sistemas de Informação
	10. Redes de Computadores
	11. Inteligência Artificial
	12. Sistemas Operacionais
	13. Segurança de Sistemas
	14. Sistemas Distribuídos

- Extração dos dados: os dados a serem utilizados no desenvolvimento deste trabalho são dispostos para livre acesso pelo Inep em formato de microdados em planilhas .csv para cada ano de aplicação da prova do ENADE. Segundo o definido na proposta, deverão ser extraídas as respostas dadas nas provas e as informações dos discentes da área de computação dos dados do ENADE para os anos pré-definidos.
- Entendimento dos dados: para compreensão dos dados extraídos, estes deverão ser observados de forma a entender sua estrutura e seus relacionamentos. Para isso, será necessário utilizar os dicionários de variáveis que são disponibilizados junto às planilhas de cada ano, identificando as propriedades dos atribu-

tos contidos nas planilhas de respostas. Também deverão ser identificados com base em qual(is) campo(s) de estudo cada questão foi construída, de maneira a auxiliar nos resultados da análise.

- Adequações da proposta: as adequações à proposta serão feitas de acordo com a fase de entendimento dos dados, de maneira que quaisquer informações que possam induzir a alterações na proposta sejam discutidas e ponderadas para verificação da aplicação ou não.
- Transformação e adequação dos dados para análise: logo após as adequações serem ponderadas, os dados extraídos serão transformados e adequados de modo que torne possível a aplicação dos algoritmos de classificação e clusterização sobre estes.
- Aplicação dos algoritmos de mineração de dados: com os dados padronizados e prontos para a mineração, serão aplicados os algoritmos estipulados de modo a se obterem resultados contendo novos padrões de análise. Para o algoritmo de classificação serão determinadas classes com base nos campos de estudos da computação, de forma que as respostas possam ser agrupadas de acordo com estes campos.
- Análise e validação dos resultados: os resultados obtidos da utilização dos algoritmos serão analisados, de forma que estes sejam avaliados e validados com base nos objetivos da proposta. Deste modo estes serão analisados a fim de se avaliar o desempenho dos discentes dos cursos de computação de acordo com seu curso e campo de estudo com maior acerto.
- Elaboração da plataforma para apresentação dos resultados: utilizando-se de técnicas computacionais e frameworks de desenvolvimento web, será desenvolvida uma plataforma em que sejam apresentados os resultados obtidos pela aplicação dos algoritmos e a análise dos resultados de forma visual.
- Disponibilização da plataforma: a plataforma será hospedada em um servidor e disponibilizada publicamente para que possa ser acessada por pessoas que tenham interesse nos resultados da análise.

IV. DESENVOLVIMENTO

A partir do estudo de trabalhos relacionados e da observação dos microdados do ENADE, foi escolhido o algoritmos de clusterização para aplicação neste trabalho devido a possibilidade de se agrupar as respostas dos discentes por curso e por campo de estudo da computação, fornecendo assim condições de analisar o desempenho os estudantes de cada um dos curso citados dentro dos campos de estudo.

Para este projeto foram consideradas as respostas dos discentes, de modo a avaliar a taxa de acerto destes nas questões propostas na prova para cada curso. Desta forma, com a identificação dos campos de estudos da computação em que as questões são baseadas, foi possível efetuar o relacionamento entre as questões da prova, as respostas dos discentes às questões e os campos de estudo. Com base neste relacionamento, o desempenho dos estudantes foi avaliado de modo a identificar os desempenhos dos discentes nos vários campos da computação.

É apresentado na Tabela 2 o número de questões existente nas provas de cada curso nos anos analisados. Percebe-se que nos anos de 2008 e 2011 existiam questões denominadas pelos organizadores do ENADE como **"componente específico"** que eram questões aplicadas para todos os discentes da área de computação. Esse "componente específico" foi excluído das provas no ano de 2014, ou seja, cada curso passou a realizar uma prova totalmente exclusiva.

Tab. 2: Quantidade de questões por curso.

Ano	Curso	Número
2008	Comp.Específico	9
	Ciência da Computação	18
	Engenharia da Computação	18
	Sistemas de Informação	18
2011	Comp.Específico	23
	Ciência da Computação	5
	Engenharia da Computação	5
	Sistemas de Informação	5
2014	Ciência da Computação	27
	Engenharia da Computação	27
	Sistemas de Informação	27
	Total Geral	182

Outro fato interessante observado na análise das questões do "componente específico" é que em 2008 essas questões correspondiam a um pequeno percentual das questões totais da prova (33,3%), já em 2011 esse percentual passou a ser de 85,2% da prova. Tal fato não é explicado nas documentações fornecidas pelo INEP.

Para a análise do desempenho dos alunos, as perguntas da prova do Enade foram separadas dentro das principais áreas da computação. Para isso, a área foi dividida inicialmente de acordo com as grandes áreas existente no modelo de mapa curricular dos cursos de computação (CC e SI) de uma IES brasileira. Nesse modelo, a partir de 6 grandes áreas é possível encaixar as disciplinas dos cursos e na sequência identificar as áreas de enquadramentos

de cada questão da prova do ENADE. São apresentados na Tabela 2 os resultados da classificação das questões utilizada para os anos analisados neste trabalho.

Tab. 3: Áreas das Questões

Área da questão	**Contagem de Numero Dados**
Arquitetura de Computadores	17
Banco de dados	11
Computação Gráfica	6
Engenharia de Software	23
Estruturas de dados	14
Inteligência Artificial	7
Lógica	27
Redes de Computadores	16
Sistemas de Informação	9
Sistemas Operacionais	12
Teorias da Computação	27
Tópicos avançados de Engenharia	13

A Tabela 2 apresenta as 12 áreas de enquadramento das questões escolhidas, cada qual seguida do número de questões das provas do ENADE classificadas. Durante a separação das questões, foram observadas áreas com um número de questões irrelevante, que para otimizar o resultado foram agrupadas em uma área próxima.

É possível notar a falta da área de desenvolvimento de software na Tabela 2, isso ocorreu devido à esta estar diretamente relacionada a Engenharia de Software e pela falta de questões sobre esta área nas provas destes anos. Pode-se perceber que devido ao teor da prova do ENADE estar voltada a análise do conhecimento dos estudantes quanto às partes teóricas da graduação as questões de conteúdos práticos não são frequentemente abordadas no contexto da prova.

Com base no modelo estabelecido foi necessária a extração dos dados referentes às respostas dos acadêmicos da área de Computação às provas do ENADE do website do Inep, para os anos de 2008, 2011 e 2014. Com os dados no formato adequado para a mineração, estes foram organizados de forma a separar os dados de cada um dos 3 cursos (Ciência da Computação, Sistemas de Informação e Engenharia da Computação).

Como guia para a separação correta dos dados, foram utilizados os dicionários de variáveis dispostos pelo Inep para as tabelas de respostas. Ao fim deste processo foi realizado o processo de análise manual das questões das provas selecionadas, de modo a enquadrá-las nas áreas abordadas na tabela 2

Para transformar, padronizar os dados e adequar o formato dos dados, foram elaborados algoritmos em Python que permitissem a comparação das respostas dos estudantes e o gabarito oficial da prova devido ao formato de armazenamento do gabarito e das questões apresentar divergências nos anos das provas. Como exemplo tem-se a prova de 2011 em que os gabaritos das provas são armazenados em um único vetor no formato string, o qual utiliza apenas do caractere 'N' para como separador dos gabaritos e o caractere 'X' para indicar uma questão anulada. A figura 3 ilustra o gabarito e das questões da prova de 2011 sem a transformação dos dados.

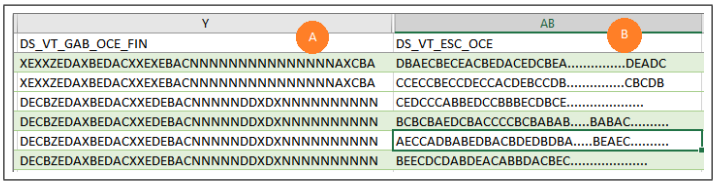
Com base no apresentado na figura 3 o algoritmo desenvolvido cria novas colunas de acordo com o total de questões do gabarito para cada curso (totalizando 27), estas colunas foram usadas para armazenar os resultados das comparações das respostas do acadêmico e do gabarito.

Para a comparação os vetores de respostas e do gabarito foram percorridos, comparando os caracteres a fim de identificar se houve o acerto ou erro do respondente quanto à questão, armazenando "certa" para o primeiro caso e "errada" para o segundo caso. Ainda na comparação, também foi verificado se houve anulação da questão ou se a questão fora deixada em branco (indicado pelos caracteres '*' e '.'), caso houvesse, foi armazenado o caractere 'X' na coluna correspondente. A figura 4 exibe um fragmento do algoritmo responsável pela comparação dos campos.

Ao observar a figura 4 é possível notar a primeira comparação sendo feita a fim de verificar que a questão não foi anulada. Em seguida verifica-se se a resposta não foi deixada em branco ou anulada. Por fim é comparado o caractere do vetor do gabrito com o caractere do vetor de resposta correspondente. Ao fim da análise foi gerado um novo arquivo de dados contendo os resultados da comparação.

Com os dados organizados por curso, a etapa seguinte envolveu a divisão das regiões em que foram aplicadas as provas. A tabela 4 apresenta a porcentagem de respostas referentes à área de computação por região do Brasil.

Esta divisão por regiões brasileiras foi necessária devido à grande concentração de respostas nas regiões Sudeste e Sul nos anos analisados, correspondendo a aproximadamente 70% do total de respostas nos anos, sendo destes 52% respostas oriundas da região Sudeste. Então, para permitir a realização da análise para cada região individualmente e a fim de evitar possíveis problemas na análise dos resultados e na aplicação dos algoritmos de mineração de dados, a divisão foi realizada.



Y	AB
DS_VT_GAB_OCE_FIN (A)	DS_VT_ESC_OCE (B)
XEXXZEDAXBEDACXXEXEBACNNNNNNNNNNNNNNNAXCBA	DBAECBECEACBEDACEDCBEA.....DEADC
XEXXZEDAXBEDACXXEXEBACNNNNNNNNNNNNNNNAXCBA	CCECCBECCDECCACDEBCCDB.....CBCDB
DECBZEDAXBEDACXXEDEBACNNNNNDDXDXNNNNNNNNNN	CEDCCCABBEDCCBBBECDBCE.....
DECBZEDAXBEDACXXEDEBACNNNNNDDXDXNNNNNNNNNN	BCBCBAEDCBACCCCBCBABAB.....BABAC.....
DECBZEDAXBEDACXXEDEBACNNNNNDDXDXNNNNNNNNNN	AECCADBABEDBACBDEDBDBA.....BEAEC.....
DECBZEDAXBEDACXXEDEBACNNNNNDDXDXNNNNNNNNNN	BEECDCDABDEACABBDACBEC.....

Fig. 3: Gabarito e respostas dos alunos no ano de 2011.

```
for j in range(len(students_answers[i])):
    if list_of_answers[i][j] == 'X' or list_of_answers[i][j] == 'Z' or list_of_answers[i][j] == 'N':
        dataset['questao_'+str(j+1)][i] = 'X'
    else:
        if students_answers[i][j] == '*' or students_answers[i][j] == '.':
            dataset['questao_'+str(j+1)][i] = 'X'
        else:
            if students_answers[i][j] == list_of_answers[i][j]:
                dataset['questao_'+str(j+1)][i] = 'Certa'
            else:
                dataset['questao_'+str(j+1)][i] = 'Errada'
```

Fig. 4: Algoritmo de comparação para os dados de 2011.

area_enquadramento	curso	ano	cluster	regiao	questao	questao	questao	questao	questao	questao	volume_i	volume_i	qtd_que	qtd_certa	qtd_erra	qtd_brani
Arquitetura de Computadores	Ciência da Computação	2008	Cluster 0	Norte	Certa	Errada	Errada	Certa	Errada	Errada	74	13	6	2	4	0
Arquitetura de Computadores	Ciência da Computação	2008	Cluster 1	Norte	Errada	Errada	Certa	Errada	Certa	Errada	19	3	6	2	4	0
Arquitetura de Computadores	Ciência da Computação	2008	Cluster 2	Norte	Errada	Errada	Errada	Errada	Errada	Errada	101	18	6	0	6	0
Arquitetura de Computadores	Ciência da Computação	2008	Cluster 3	Norte	Certa	Errada	Errada	Errada	Certa	Errada	24	4	6	2	4	0

Fig. 5: Resultados da clusterização das questões da área de Arquitetura de Computadores do ano de 2008 da região Norte.

Tab. 4: Número e porcentagem de respostas por regiões.

Ano	Total	Região	Nº respostas	(%) total
2008	37.152	Norte	1764	4,75
		Nordeste	4694	12,63
		Centro-Oeste	3565	9,60
		Sudeste	19896	53,55
		Sul	7235	19,47
2011	21.913	Norte	1017	4,64
		Nordeste	3379	15,42
		Centro-Oeste	1860	8,49
		Sudeste	11814	53,91
		Sul	3843	17,54
2014	24.076	Norte	1426	5,92
		Nordeste	4073	16,92
		Centro-Oeste	1975	8,20
		Sudeste	12270	50,96
		Sul	4332	17,99

V. RESULTADOS

A fim de se atingir os objetivos do trabalho foram analisados e testados diversos algoritmos de clusterização, associação e classificação. O algoritmo que apresentou um melhor resultado para atender o objetivo do trabalho foi o algoritmo de clustrização KMeans, que gerou agrupamentos de acordo com a característica das respostas das questões (certa ou errada) junto à região de origem da resposta.

Devido a necessidade de definição do numero k de grupos necessários foram feitos testes sobre os dados a fim de se encontrar um número em que o algoritmo pudesse apresentar um resultado válido. Para isso foram escolhidos um numero de k=20 de modo que o algoritmo viesse a gerar k ou menos grupos sobre os dados. Deste modo, o algoritmo foi aplicado sobre as questões enquadradas em cada área de maneira que os resultados da aplicação do algoritmo foram armazenadas em tabelas para cada ano e curso, separadas por páginas para cada área de enquadramento das questões. A figura 5 apresenta a tabela com os resultados para as questões de Arquitetura de Computadores da região Norte.

A partir destes resultados foram gerados gráficos que representassem a taxa de questões certas em cada cluster em função do numero de incidências ocorridas naquele grupo. Deste modo é possível observar a taxa de questões em que foram obtidas respostas corretas e erradas por região. A figura 6 apresenta um gráfico do volume de incidências (quantidade de dados) por região resultantes da aplicação do algoritmo sobre os dados de 2011.

A fim de validar estes resultados, foram excluídos grupos que continham um volume de incidências que correspondessem a menos de 3% do volume de dados originais. A figura 7 apresenta o gráfico re-

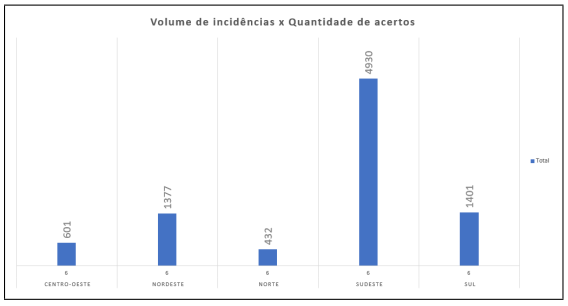


Fig. 6: Volume de incidências de respostas por região na área de Lógica de Ciência da Computação no ano de 2011.

sultante da aplicação do algoritmo às cinco regiões sobre as respostas às questões da área de Lógica para o curso de Ciência da Computação no ano de 2011.

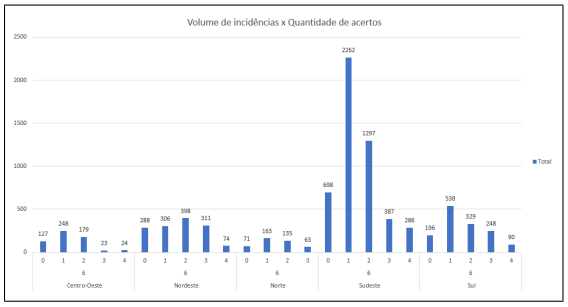


Fig. 7: Taxas de acerto por regiões na área de Lógica de estudantes de Ciência da Computação no ano de 2011.

O gráfico disposto na figura 9 apresenta o volume de incidências(eixo Y) em função da quantidade de respostas corretas dos discentes de Ciência da Computação (eixo X) nas questões da área de Lógica para cada região em 2011. Ao realizar uma analise estatística sobre o desempenho geral dos discentes nas questões da área, é possível considerar que os discentes da região Sul obtiveram a maior taxa de acerto nas questões da área de Lógica, tendo um total 1205(aproximadamente 86%) incidências, das 1401 incidências dos clusters, considerando que o discente acertou ao menos uma questão. Seguida da região Sudeste, que apesar de maior volume de respostas teve um total de 4232(85,8%) com base nos mesmos dados. Com a menor taxa de acertos tem-se a região Centro-Oeste com um total de 474(78,8%) sob 601 respostas.

Ao analisar o desempenho de modo a considerar o acerto total das 6 questões, a região Sul também encontra-se a frente, com um total de 90 respostas que correspondem a 6% do volume total de incidências para a região. Seguida pela região Sudeste com 5.8% do total de respostas para a região e pela região Nordeste com 5,3%. A figura 8 gráfico do volume de incidências resultantes da aplicação do algoritmo sobre os dados de 2014.

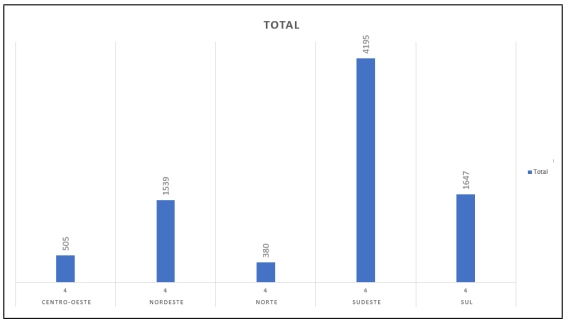


Fig. 8: Volume de incidências de respostas por região na área de Lógica de Ciência da Computação no ano de 2014.

O gráfico da figura 8 é apresenta uma ligeira mudança no volume de incidências quando colocado em contraste com gráfico da figura 6. Para fins de comparação, observa-se que houve um aumento no volume de incidências das regiões Centro-Oeste, Nordeste e Sul e em contra partida houve uma diminuição do volume de respostas das regiões Norte e Sudeste para o curso de Ciência da Computação. De modo a analisar estas informações a figura 9 traz o gráfico de desempenho dos discentes de Ciência da Computação nas questões da área de Lógica para cada região em 2014.

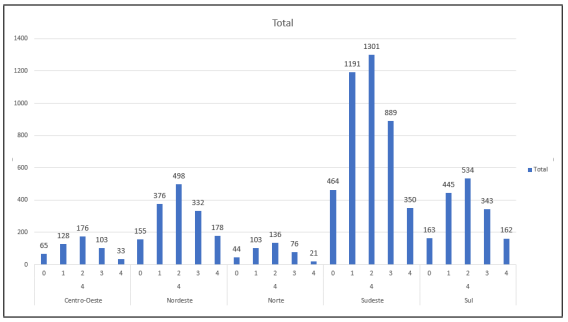


Fig. 9: Taxas de acerto por regiões na área de Lógica de estudantes de Ciência da Computação no ano de 2014.

Ao realizar a análise de desempenho nos dados minerados de 2014, seguindo os mesmos critério utilizados para 2011, pode-se notar que a região Sul também se encontra estatisticamente a frente das demais regiões com 1484(90,1% do total da região)

incidências com ao menos uma questão certa nas questões de Lógica. Logo após a região Sul vem a região Nordeste com 1384(89,9%) incidências. Com a menor taxa encontra-se a região Centro-Oeste com 440(87,1%) respostas com ao menos uma certa das 505 totais da região. Diante disso pode-se notar que a região Sul do país continua com o maior volume de acerto em comparação ao ano anterior, contudo a região Nordeste ultrapassou a região Sudeste em 2014 ao alcançar uma incidência maior de respostas com pelo menos um acerto.

VI. CONCLUSÃO

O presente trabalho descreveu alguns dos resultados da mineração dos microdados do ENADE das respostas dos estudantes da área da Computação que foi realizado com o objetivo de avaliar o desempenho desses estudantes e acompanhar e a análise do desempenho destes com base nos campos de estudo da computação e seus cursos. Para isso foi apresentado um referencial teórico acerca do tema, assim como conceitos e características do modelo de referência CRISP-DM, que foi o modelo de referência escolhido para apoiar no processo de alcance dos objetivos propostos.

Os resultados obtidos podem ser analisados e utilizados por Instituições de Ensino Superior (IES) brasileiras que tenham cursos na área de computação, assim como para pesquisadores da área de Mineração de Dados, ou outros interessados.

A partir do endereço https://enadedm.netlify.com/ é possível visualizar os padrões identificados e demais resultados obtidos com a mineração. A plataforma de visualização criada apresenta de forma gráfica as informações obtidas da aplicação dos algoritmos e as informações da análise feita. A aplicação está disponibilizada publicamente na internet de modo que facilite seu acesso. A princípio esta será utilizada como forma de visualização das informações, contudo, seus resultados poderão ser utilizados em conjunto a demais pesquisas com foco mais delimitado a regiões ou cursos específicos, de maneira a embasar e auxiliar na obtenção e observação de novas informações.

Espera-se com as informações obtidas fornecer aos estudantes da área de computação um conjunto de dados relevantes, coerentes e úteis para implementação de melhorias nos cursos de graduação da área de computação das IES brasileiras. Diante disso, a utilização destas informações poderá beneficiar o entendimento de como os focos dos cursos da área de computação estão sendo aplicados aos alunos e auxiliar no entendimento das diferenças entre os focos destes.

Como trabalhos futuros, pretende-se ampliar o escopo da mineração de dados para outras áreas de aplicação do ENADE, como também ampliar a análise nos resultados da mineração ao utilizar diferentes algoritmos e técnicas de mineração dos dados.

REFERÊNCIAS

[1] T. A. Bartolomeu et al. Modelo de investigação de acidentes do trabalho baseado na aplicação de tecnologias de extração de conhecimento. Florianópolis, SC, 2002.

[2] C. O. Camilo and J. C. d. Silva. Mineração de dados: Conceitos, tarefas, métodos e ferramentas. Universidade Federal de Goiás (UFC), pages 1–29, 2009.

[3] I. K. Center. Visão geral da ajuda do crisp-dm. ibm knowledge center, 2017. Accessed: 2018-04-17.

[4] S. da Costa Côrtes, R. M. Porcaro, and S. Lifschitz. Mineração de dados-funcionalidades, técnicas e abordagens. PUC, 2002.

[5] L. A. da Silva, S. M. Peres, and C. Boscarioli. Introdução à mineração de dados: com aplicações em R. Elsevier Brasil, 2017.

[6] M. M. Dias, L. A. da Silva Filho, A. D. P. Lino, E. L. Favero, and E. M. L. S. Ramos. Aplicação de técnicas de mineração de dados no processo de aprendizagem na educação a distância. In Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE), 2008.

[7] Instituto de Pesquisas Anísio Teixeira - INEP. Microdados do enade, 2018. Accessed: 2018-04-17.

[8] M. Kantardzic. Data mining: concepts, models, methods, and algorithms. John Wiley & Sons, 2011.

[9] S. Moro, R. Laureano, and P. Cortez. Using data mining for bank direct marketing: An application of the crisp-dm methodology. In Proceedings of European Simulation and Modelling Conference-ESM-2011, pages 117–121. EUROSIS-ETI, 2011.

[10] D. L. Olson and D. Delen. Advanced data mining techniques. Springer Science & Business Media, 2008.

[11] D. C. Pelegrin, D. P. Casagrande, D. P. Martins, M. C. de Mattos, P. W. T. de Azevedo Simões, and R. Charnovscki. A shell de data mining orion: Classificação, clusterização e associação. Anais SULCOMP, 1, 2012.

[12] L. A. Silva, A. H. Morino, and T. M. C. Sato. Prática de mineração de dados no exame nacional do ensino médio. In Anais dos Workshops do Congresso Brasileiro de Informática na Educação, page 651, 2014.

[13] P.-N. Tan, M. Steinbach, and V. Kumar. Introdução ao datamining: mineração de dados. Ciência Moderna, 2009.

[14] N. P. B. Vista, M. F. Figueiró, and P. M. M. Chicon. Técnicas de mineração de dados aplicadas aos microdados do enade para avaliar o desempenho dos acadêmicos do curso de ciência da computação no rio grande do sul utilizando o software r. I Seminário de Pesquisa Científica e Tecnológica, 1(1), 2017.

RENATO MARINHO ALVES Bacharel em Sistemas de Informação pelo Centro Universitário Luterano de Palmas (CEULP/ULBRA). Atua como Desenvolvedor Web na URPAY TECNOLOGIA EM PAGAMENTOS LTDA. Tem experiência na área de Ciência da Computação, com ênfase em Sistemas de Informação. Suas principais áreas de interesse são: Data Science, Inteligência Artificial, Machine Learning e Deep Learning.

ALEXANDRE MORAES MATOS Bacharel em Sistemas de Informação pelo Centro Universitário Luterano de Palmas (CEULP/ULBRA). Atua como Desenvolvedor Web na URPAY TECNOLOGIA EM PAGAMENTOS LTDA. Tem experiência na área de Ciência da Computação, com ênfase em Sistemas de Informação.

EDNA DIAS CANEDO Doutora em Engenharia Elétrica pela Universidade de Brasília (UNB). Área de concentração: Telecomunicações - Modelo de Confiança para a Troca de Arquivos em uma Nuvem Privada. Título concedido em 15.08.2012. Mestre pela Universidade Federal da Paraíba UFPB. Área de concentração: Sistemas de Software. Área Especifica: Engenharia de Software, título concedido em 29.08.2002. Graduada em Análise de Sistemas pela Universidade Salgado de Oliveira; Goiás (1999). Professora do curso de Engenharia de Software da Faculdade FGA Gama, da Universidade de Brasília - UNB. Atua na área de desenvolvimento de sistemas desde 1998, tendo trabalhado como Analista de Sistemas na Empresa de Processamento de Dados do Estado de Goiás Prodago, até o ano de 2000. Em Brasília atuou na Poliedro como Consultora de Desenvolvimento e Gerência de Projetos e como Analista de Sistemas Pleno em Desenvolvimento na ECT - Empresa de Correios e Telégrafos até abril de 2010, desenvolvendo atividades na área de Governança em Tecnologia da Informação na implantação do framework COBIT. Desde 2000 dedica-se a atividades de docência universitária nos cursos de graduação e pós-graduação, atuando na área de Segurança de Software, Sistemas de Software, Engenharia de Software, Orientação a Objetos, Gerência de Projetos, Teste de Software, Arquitetura Orientada a Serviços, Programação e Governança em Tecnologia da Informação.

Singular Engenharia, Tecnologia e Gestão

Vol. 1, N. 2, Outubro, 2019
eISSN: 2596-2604
https://doi.org/10.33911/singular-etg.v1i2