2023 Brazilian Technology Symposium

A Multiplataform Business Intelligence Solution for Software Test Analysis

André Luiz Printes
Embedded Systems Laboratory
Universidade do Estado do Amazonas
Manaus, Brazil
aprintes@gmail.com

Daniel de Sena Fernandes
Embedded Systems Laboratory
Universidade do Estado do Amazonas
Manaus, Brazil
dsfernandes@hub-uea.org

Rubens de Andrade Fernandes
Embedded Systems Laboratory
Universidade do Estado do Amazonas
Manaus, Brazil
rfernandes@uea.edu.br

Israel Gondres Torné
Embedded Systems Laboratory
Universidade do Estado do Amazonas
Manaus, Brazil
itorne@uea.edu.br

Fabricio Ribeiro Seppe
Embedded Systems Laboratory
Universidade do Estado do Amazonas
Manaus, Brazil
fab.seppe@gmail.com

Yago Costa de Oliveira
Embedded Systems Laboratory
Universidade do Estado do Amazonas
Manaus, Brazil
yago.cst.oliveira@gmail.com

***Abstract*— This article discusses the development of a Business Intelligence solution aimed at optimizing software testing analysis. The proposal's main objective is to provide a more efficient analysis of test results conducted during the development cycle of Frontend, Backend, Mobile, and Web applications. To achieve this, the solution integrates tools such as Google Sheets, Clickup, and Power BI desktop. This approach not only facilitates data management and visualization but also promotes more effective collaboration between Quality Assurance and development teams, enabling a more informed decision-making process based on concrete and reliable data. Furthermore, the work emphasizes the importance of the proposed resources in the continuous improvement of the software development process and suggests their replication in other contexts by adding new functionalities.**

Keywords— Business Intelligence, Quality Assurance, test management, PowerBI, ClickUp.

INTRODUCTION

In the realm of software product development, the testing process plays a critical role, ensuring that the software operates as expected, with a reduced number of failures and in compliance with established requirements. In this context, the application of Business Intelligence (BI) in the field of software testing emerges as a solution to enhance test analysis, enabling managers and Quality Assurance (QA) and development teams to make more precise and informed decisions based on the data resulting from test results analysis.

This understanding is supported by the contributions of the authors in [1], who emphasize the importance of evaluating software through rigorous testing. Through this, it becomes possible to assess the software's quality by revealing defects, system failures, and unexpected behaviors. These insights are essential to ensure that the software meets both functional and non-functional requirements, thus promoting a reliable and satisfactory experience for users. Similarly, the authors in [2] highlight the importance of Quality Assurance (QA), embraced by the software testing community, as a methodical and planned approach to ensure quality in all stages of the development cycle. Through the application of disciplined methodologies, transparent processes, and thorough testing, QA acts as a crucial pillar in creating robust, reliable software that meets the needs and expectations of users.

The literature presents various researches focusing on applications within this theme, utilizing different approaches. In [3], the authors developed a system for visualization and analysis of purchase and sales data of products, based on the Laravel framework and Python language. On the other hand, the work in [4] proposed an application for software testing management, highlighting failures and successes. In this context, the authors of [5] developed a tool for analyzing automated tests for web applications, using the Cypress framework integrated with Pentaho software. Furthermore, in the research [6], the authors presented the 4Testers tool, an application that assists in the documentation and management of software testing scenarios and scripts. The work in [7] proposed a visualization approach to represent test case results and their relation to object-oriented software systems. In [8], the authors developed a visualization technique to represent software testing results on object-oriented code elements, including method, class, package, UML (Unified Modeling Language), and system.

While the literature demonstrates significant advancements in enhancing software testing, gaps related to the analysis of results with accessible implementation, automation, and evaluation tools persist. In this regard, this article focuses on developing a BI solution to optimize the analysis and monitoring of test results throughout the software development cycle. The tool also aims to provide resources for analyzing the performance of the development team and visualizing the status of software product-related failures. The solution offers insights into test coverage, result quality, and the efficiency of testing activities, with the purpose of improving the development quality of Frontend, Backend, Mobile, and Web applications.

One of the distinctive elements of this work is the integration of easily accessible tools in the development of the solution, including Google Sheets, Clickup, and Power BI desktop. This approach facilitates team collaboration and simplifies usability, maintenance, and data updates. With the implementation of this system in a QA team, the goal is to enable managers, testers, and developers to monitor the progress of tests more efficiently and accurately. Additionally, they can identify areas of concern and make informed decisions to enhance the overall quality of the developed software.

To present the proposal of this work, we have divided the prepared content into the following sections: Section II provides a brief overview of theoretical concepts for the readers' understanding. Section III displays the resources used and the methodology proposed in this work. Finally, Sections VII and IX present the results and conclusions obtained, respectively..

Theoretical Reference

Quality Assurance

According to [9], Quality Assurance (QA) is defined as a systematic and planned set of activities that ensure that the processes and products related to software development meet established requirements, standards and procedures. QA involves the implementation of methods, techniques, and tools aimed at ensuring that the software is produced consistently, reliably, and of high quality.

The Quality Assurance (QA) approach permeates all stages of the software development lifecycle, from the initial planning stage to product delivery. As emphasized in [10], the essence of QA spans from simple defect identification to proactive prevention through the adoption of good engineering practices, methodical processes, and continuous reviews.

Business Inteligence

Business Intelligence (BI) involves transforming raw data into meaningful and actionable insights to guide effective decisions. As outlined in [11], BI encompasses a systematic and strategic approach to collecting, processing, analyzing, and providing resources for data visualization from a variety of sources. Its presence is central in modern business environments, where data-driven decision-making is crucial for success.

Scrum Methodology

The Scrum methodology, an agile project management approach, has gained prominence in software development literature due to its effectiveness in promoting iterative and collaborative deliveries [12]. Scrum operates in cycles known as "sprints", each lasting two to four weeks. During a sprint, a multifunctional team works on selected tasks from a product backlog. Monitoring progress is facilitated through daily meetings, where team members share updates and identify obstacles. In [13], the authors emphasize the importance of the Scrum Master, Product Owner, and Development Team roles, as well as the need to maintain a well-managed product backlog.

Theoretical concepts

Development Environment

The solution proposed in this work was developed on a local server with a 12th generation Intel Core i7 processor, 16 GB DDR4 RAM, and a 512 GB SSD. However, the tool is made available for cloud access to enable remote access to desired information and reports from anywhere, providing geographical independence for the offered resources.

Tools and Technologies definitions

Initially, a detailed survey of the requirements for the proposed solution was conducted in collaboration with the QA and development teams. This process involved meetings, interviews, and analysis of previous information to understand the needs and expectations regarding the tool proposed in this work. Based on this, the technologies and tools to be used were defined.

Power BI was chosen as the main tool for building the dashboard of the proposed solution due to its data visualization capabilities and integration with other data sources. On the other hand, Google Sheets was selected to organize and store information on test cases and bug reports, facilitating collaboration and data updates. Additionally, the software Clickup was chosen as the tool for reporting bugs to the development team. Concepts and additional functionalities of the proposed software tools will be detailed below.

Power BI Desktop: A local application that allows the extraction, transformation, and visualization of data related to corporate process value chains [14]. In addition to the capability to extract data from various sources, this software enables the development of reports and visual collections that can be shared with other individuals in an organizational environment.

Google Sheets: A cloud-hosted web application for collaborative online editing and creation of spreadsheets [15]. Additionally, this application provides users with features to import and export spreadsheets in various formats, including files compatible with Microsoft Excel and CSV (comma-separated values) files. Google Sheets also offers the ability to create forms and surveys directly within the spreadsheets, allowing for the organized and automated collection of data and responses.

ClickUp: Software designed to enhance the productivity of development teams, developed with the purpose of improving company productivity [16]. It provides graphical features for intuitive report and backlog creation, optimizing management processes in the corporate environment.

Development Methodology

The development of the proposed tool followed an iterative and incremental approach, in alignment with best practices of agile development. Throughout this process, collaboration between the QA team and software developers played a central role in structuring the solution. Based on this, it was possible to define the visualizations for displaying software test results and implement the necessary features to meet the established objectives. Figure 1 illustrates the architecture of the proposed tool with the respective integrations of the software tools used within the desired functionality flow of the solution.

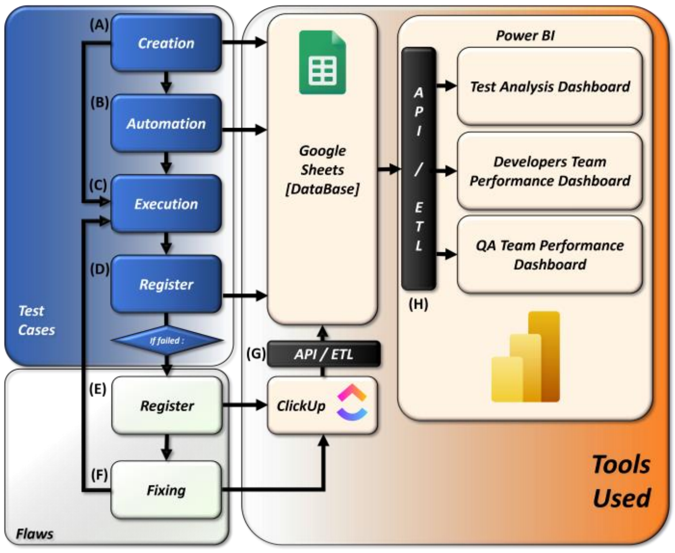


Figure 1: Architecture of the proposed solution.

Next, we will detail the key aspects of the proposed solution as depicted in Figure 1:

Test cases:

Test cases are conditions developed to implement tests in the software appropriately and ensure that the product aligns with the established requirements. They should specify what needs to be analyzed and the expected results, aiming to uncover potential defects in the software. In the architecture of the proposed solution, test case handling processes were included, interconnected to generalize the tool for other systems and applications. The test case processes incorporated into the proposal were:

Test cases creation (A):
In this work's solution, the test creation process followed a structured approach based on predefined software requirements and business rules. For each test case, a thorough analysis of the requirements was conducted to fully understand the software's functionalities and objectives under examination. The documentation of these cases is recorded in Google Sheets (Database).
Test cases automation (B):
The automation of test cases is based on the previously established test cases. We used the details as a starting point to develop automation scripts. This allowed us to reproduce the same test scenarios in an automated manner, ensuring consistency and saving time in this process. The progress and records of the automation process results are also logged in Google Sheets.
Test cases execution (C):
During the test execution, we provide the inputs specified in the automated test cases and register the outputs generated by the software. By comparing these outputs with the expected results, we identify discrepancies and potential issues. As a result, we can effectively detect and report failures, contributing to the ongoing maintenance of software quality.
Test cases registration (D):
The register of tests and execution results are done in the Google Sheets tool. The records include the date and time of execution, the provided inputs, the outputs generated by the software, and their respective results.

Flaws:

Software failures stem from errors, defects, or issues that lead to undesirable behaviors, vulnerabilities, and other types of malfunctions. The proposed solution includes processes for handling failures, described as follows:
Flaws registration (E):
In cases of failures in test results, the results are reported in the Clickup tool for issue tracking. The log includes identifications of the failed test cases, the current state of the problem, priority, and the individual responsible for resolution.
Through this process, we ensure that issues identified during the testing process are properly documented, tracked, and forwarded for resolution.
Flaws fixing (F):
In the process of correcting failures, once a failure is recorded and documented in the Clickup tool, the team responsible for fixing takes charge of corrective actions. After the necessary corrections, the test case is executed again to validate if the failure has been rectified.

Tools used:

Based on the previously mentioned testing processes, the flow of generated information moves between the tools used and towards the monitoring dashboards. To achieve this, it's essential to understand how the integration of these tools was carried out during the process. The integration of Clickup with Google Sheets was accomplished by configuring Application Programming Interfaces (APIs) and implementing Extract, Transform, Load (ETL) processes to handle data appropriately (7). This allowed the failure data recorded in the Clickup tool to be inserted into Google Sheets.

DEVELOPMENT OF SOLUTION'S DASHBOARDS

The dashboards were developed using the Power BI tool to analyze the test results in the software development cycle, aiming to present and facilitate the understanding of relevant test information. Initially, the team outlined the structure of the dashboards, identifying crucial information to be displayed based on the data in Google Sheets. Based on the collected requirements, charts, filters, and visualizations were selected to display and select data in a clear and synthesized manner. The filters include the selection of test status, severity of failures, and test execution date. Next, we will detail the objectives and functionalities of the dashboards indicated in Figure 1.
Test Analysis Dashboard:
In the overall test analysis dashboard (Figure 2), you can view the total values of different test categories, including manual tests, automated tests, frontend tests, backend tests, mobile

tests, and web tests. Additionally, there is a pie chart representing the overall status of manual tests. The panel also includes two clustered bar charts to display test sessions for both backend and frontend, along with a clustered column chart that shows the overall status of automated tests. To enhance data visualization, the panel is equipped with corresponding filters.

Developers Team Performance Dashboard:

In the performance dashboard (Figure 4), for developers, you can view the quantity of failures attributed and resolved by each developer in pie charts. Additionally, there is a clustered column chart showing the average resolution time of failures by severity for each developer. The panel also includes a Gantt chart for temporal analysis of failure resolution, providing a clear view of performance over time.

QA Team Performance Dashboard:

In the QA Team Performance Dashboard, you can view essential information about the performance of each QA member, including the total accumulated test creation by member, the total accumulated automated test creation by member, and the total accumulated test executions per member. Additionally, the panel includes a stacked area chart representing the temporal evolution of test cases, executions, and automations performed by each collaborator. The number of failures reported by each member is also visible in the dashboard.

I. Solution's Test and Validation

The proposed solution underwent a series of tests to ensure its functionalities. Initially, integration tests enabled the analysis of interactions between the different software elements used to ensure the proposed functionalities. Additionally, usability tests were conducted to confirm that the system meets not only the predefined requirements but also the high expectations of the QA team and end-users. Next, details of the procedures for system validation are presented:

- The integration tests among Google Sheets, ClickUp, and PowerBI were manually conducted, focusing on verifying functionalities related to data manipulation, synchronization, data transmission, interface updates, and database operations. This meticulous approach ensured that the integration of the platforms used in our proposal effectively contributes to providing accurate information to the dashboard.
- Conversely, during usability testing, both the QA and development teams critically assessed the dashboard's interface for navigability, efficiency, and its ability to provide information in a clear and comprehensible manner. Based on this assessment, we made enhancements to the proposal pertaining to usability, intuitiveness of filtering functionalities, readability of charts, and ease of data access. The usability tests significantly contributed to an efficient analysis of test results in the software development cycle.

RESULTS

Figure 2 showcases the main page of the test analysis dashboard. The pie and column charts depict the distribution of tests per category, encompassing both manual and automated tests for frontend, backend, mobile, and web applications. This visualization facilitates a quick and comprehensive analysis of test coverage across various areas of the system. The pie chart illustrates the overall test status, representing the proportion of tests that passed, failed, are blocked, or have not been executed. This visualization allows the team to promptly identify the overall software quality and critical areas that require immediate attention. The dashboard also incorporates a funnel chart, illustrating the progress concerning reported failures at different stages. This visual representation has proven pivotal in understanding and enhancing the efficiency of the failure resolution process, resulting in a substantial reduction in the time required to troubleshoot in software development.

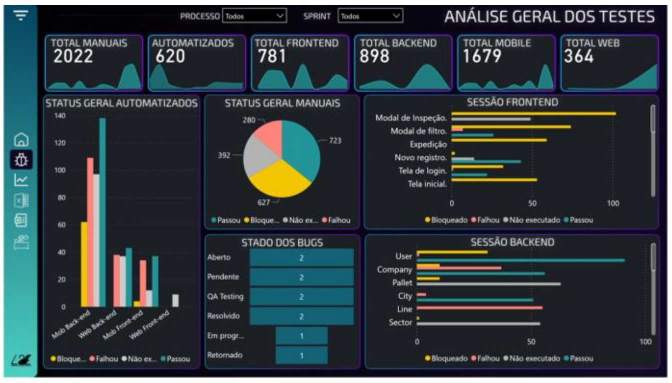


Figure 2: Test Analysis Dashboard.

Figure 3 illustrates essential metrics related to the individual performance of software testing team members. These critical pieces of information encompass the total accumulated tests created by each tester, the total accumulation of automated tests, and the total accumulation of test executions. Furthermore, this panel is enriched with stacked area charts that provide a visual representation of the evolution over time concerning test case-related activities. This chart not only tracks test creation but also their execution and automation implementation. Moreover, it displays the progression of the number of test cases, executions, and automations, enabling an in-depth analysis of trends and patterns throughout the software development cycle.

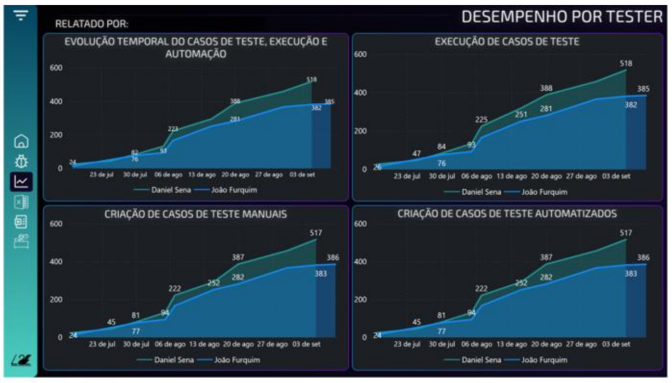
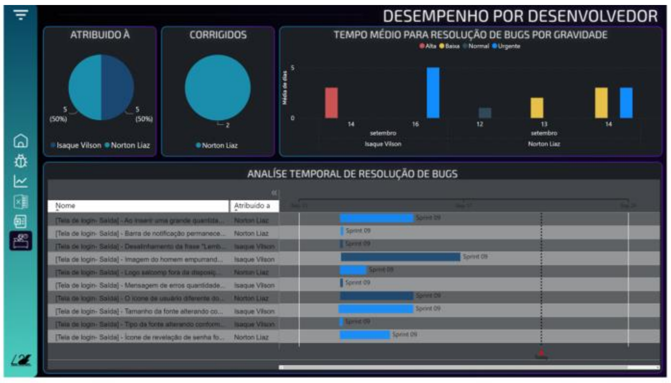

DESEMPENHO POR TESTER
RELATADO POR:
EVOLUÇÃO TEMPORAL DO CASOS DE TESTE, EXECUÇÃO E AUTOMAÇÃO
EXECUÇÃO DE CASOS DE TESTE
CRIAÇÃO DE CASOS DE TESTE MANUAIS
CRIAÇÃO DE CASOS DE TESTE AUTOMATIZADOS
Daniel Sena
João Furquim
Figure 3: QA Team Performance Dashboard.

Figure 4 provides insights presented in pie charts illustrating the quantity of failures attributed to and resolved by each developer. Additionally, there is a clustered column chart offering an analysis of the average resolution time of failures categorized by severity. This allows for a clear understanding of each team member's performance over time. Within the same panel, we have also included a Gantt chart, offering a temporal analysis of failure resolution. This visual representation enables an effective visualization of progression and efficiency in failure resolution, providing a clear view of individual and collective performance over time.



Nome	Atribuído a
[Tela de login- Saída] - Ao inserir uma grande quantida...	Norton Liaz
[Tela de login- Saída] - Barra de notificação permanece...	Norton Liaz
[Tela de login- Saída] - Desalinhamento da frase "Lemb...	Isaque Vilson
[Tela de login- Saída] - Imagem do homem empurrand...	Isaque Vilson
[Tela de login- Saída] - Logo salcomp fora da disposiç...	Norton Liaz
[Tela de login- Saída] - Mensagem de erros quantidade...	Isaque Vilson
[Tela de login- Saída] - O ícone de usuário diferente do...	Norton Liaz
[Tela de login- Saída] - Tamanho da fonte alterando co...	Isaque Vilson
[Tela de login- Saída] - Tipo da fonte alterando conform...	Isaque Vilson
[Tela de login- Saída] - Ícone de revelação de senha fo...	Norton Liaz

Figure 4: Developers Team Performance Dashboard.

In addition to providing a more comprehensive representation for evaluating test results, the implementation of the proposed solution has resulted in several notable achievements that have bolstered the effectiveness and quality of the software development cycle:

1. More informed and well-founded decisions through attractive visualization of the obtained results;
1. Continuous and dynamic monitoring of test results, keeping the software development team updated on test progress and failure resolution;
2. Optimization of resource allocation by directing testing efforts towards more critical areas;
3. Enhancement of software quality through the results provided by the solution.
4. Enhancement of engagement and collaboration among QA team members and software development.

CHALLENGES FACED AND STRATEGIC OVERCOME

During the development of the solution, the QA and development team encountered several significant challenges. The integration of data between platforms represented a critical milestone in the project, aiming to facilitate the development of a framework with minimal complexity, utilizing well-established software solutions. To achieve this, it was necessary to establish an efficient connection between Clickup and Power BI, ensuring data integrity and robust updating of information on the dashboard to guarantee precise and reliable analysis. In terms of customization and interactivity, the implementation of a filter modal and interactive features posed technical and design challenges. Maintaining the fluidity and usability of these functionalities became a priority, with a focus on delivering an optimized user experience. Finally, adherence to the team's needs emerged as a central challenge. The project required an iterative and collaborative process of collecting feedback from the management and development team to ensure that the solution was meticulously aligned with the team's needs, as well as the specificities of the organization's development cycle. This collaborative, user-centered approach proved to be fundamental for the successful implementation of the solution.

CONCLUSION

The proposed solution has emerged as a transformative tool in the context of the software development cycle, working towards continuous improvement and informed action. This resource provided the QA and development team with a precise and comprehensive view of test results, enabling data-driven and reliable decision-making. The seamless integration between Google Sheets, Power BI, and Clickup not only facilitated dynamic test tracking but also fostered a synchronized collaborative environment among team members, enhancing their collective efficiency. Furthermore, the high level of customization of the tool allowed users to engage more deeply with the data, streamlining the identification of problematic areas and enabling quick and effective interventions.
In summary, the dashboard has been established as an essential tool, allowing the development team to navigate the complex software creation process with greater clarity, confidence, and competence, closely aligning the product's quality with the high expectations set for the project. For future solutions, we propose replicating the proposed tool for other cases and systems, as well as specifying new functionalities for evaluating software test results.

REFERENCES

[1] Kane L, Liu V, McKague M, Walker G (2023) An experimental field comparison of wi-fi halow and lora for the smart grid. Sensors 23(17):7409
[2] Graham D, Black R, Van Veenendaal E (2021) Foundations of softwaretesting ISTQB Certification. Cengage Learning
[3] MOTA AFE (2021) Técnicas de bi aplicados em sistema de dashboard
[4] Dzidic E (2023) Data visualization of software test results: A financial technology case study
[5] Campos MM, da Silva EdO (2023) Implementação de uma ferramenta de análise de testes automatizados para aumento de produtividade em um ambiente de desenvolvimento. Caderno de Estudos em Engenharia de Software 5(1)
[6] GONC¸ ALVES J, SILVA SR (2020) 4 testers: documentacão e gerenciamento de testes de software
[7] Hammad M, Otoom AF, Hammad M, Al-Jawabreh N, Abu Seini R (2020) Multiview visualization of software testing results. International Journal of Computing and Digital Systems 9(1)
[8] Otoom AF, Hammad M, Al-Jawabreh N, Seini RA (2016) Visualizing testing results for software projects. In: Proc. of the 17th International Arab Conference on Information Technology (ACIT'16), Morocco

[9] Rosak-Szyrocka J, ywiolek J, Shahbaz M (2023) Quality Management, Value Creation, and the Digital Economy. Taylor & Francis
[10] Immonen M (2023) Java-sovelluksen kestavyystestausprosessin automati-sointi
[11] Basile LJ, Carbonara N, Pellegrino R, Panniello U (2023) Business intelli-gence in the healthcare industry: The utilization of a data-driven approachto support clinical decision making. Technovation 120:102,482
[12] Kadenic MD, Koumaditis K, Junker-Jensen L (2023) Mastering scrum with a focus on team maturity and key components of scrum. Information and Software Technology 153:107,079
[13] Verwijs C, Russo D (2023) A theory of scrum team effectiveness. ACM Transactions on Software Engineering and Methodology 32(3):1–51
[14] Power B, Excel U, Desktop P, Tiles P (2021) Microsoft power bi. Available here: https://powerbi microsoft com/en-us 130
[15] Dong E, Ratcliff J, Goyea TD, Katz A, Lau R, Ng TK, Garcia B, BoltE, Prata S, Zhang D, et al (2022) The johns hopkins university center for systems science and engineering covid-19 dashboard: data collection process, challenges faced, and lessons learned. The lancet infectious diseases 22(12):e370–e376
[16] Garcia EJ, Chadha S, Duan X, Alhmod M, Singh J, Ahmad MA (2023) Group project: Report-building an olympic stadium.