PROV-SwSystem: A Web System to Capture and Analyze Provenance Data from Software Development Processes

Laiza Mota de Souza
laizasouza07@hotmail.com
Centro Federal de Educação Tecnológica de Minas Gerais
Leopoldina, Minas Gerais, Brasil

Marcela Gomes Pinheiro
marcela.gomes01@outlook.com
Centro Federal de Educação Tecnológica de Minas Gerais
Leopoldina, Minas Gerais, Brasil

Rafael de Oliveira Lacerda
rafaellacerda.ambiental@gmail.com
Centro Federal de Educação Tecnológica de Minas Gerais
Leopoldina, Minas Gerais, Brasil

Gabriel Medeiros Macedo
gabriellmacedo25@hotmail.com
Centro Federal de Educação Tecnológica de Minas Gerais
Leopoldina, Minas Gerais, Brasil

Gabriella Castro Barbosa Costa Dalpra
gabriella@cefetmg.br
Centro Federal de Educação Tecnológica de Minas Gerais
Leopoldina, Minas Gerais, Brasil

ABSTRACT

Data provenance encompasses the description of the origin and historical events of data processing. Although models such as W3C PROV allow capturing and analyzing provenance data, additional methods are required to address software development processes' specific uses and complexities. In this context, the PROV-SwProcess provenance model was developed to extend W3C PROV. However, the PROV-SwProcess provenance model does not provide a specific tool to enable its instantiation. In this vein, this paper aims to present the PROV-SwSystem - a web-based system to capture, store, and analyze software process data along with their provenance, adhering to the PROV-SwProcess model. To achieve this, the Design Science Research (DSR) methodology was adopted, supporting the system creation, construction, and validation. PROV-SwSystem was implemented using Javascript, Bootstrap, Node.js, and Jquery, encompassing all predefined use cases. Unit tests and system tests were performed, identifying six redundant variables and five redundant functions. In addition, an exploratory usability test was performed with an external user, revealing fifteen navigation issues, eight attempts to register data in inappropriate sections, six registration confirmation failures, and data integration problems. Actions for data entry validation and interface improvements were implemented, enabling the tool's usage and paving the way for future testing on a larger scale.

KEYWORDS

Software Development, Provenance Data, PROV, PROV-SwProcess Model, Web System

1 INTRODUCTION

A software process is defined as "a set of activities, methods, practices, and transformations that people use to develop and maintain software and its associated products" [1]. The appropriate selection of the software process to be used in an application is critical to ensuring the quality of the resulting products. Furthermore, data collected during software engineering processes serve as a valuable source of information on the evolution and history of a software project [2]. Currently, system development companies rely on data-driven practices in their operations [3]. However, adopting these data-oriented approaches in software development is accompanied by challenges, including data collection, storage, organization, sharing, and management, which can lead to limited feedback and uncertain decision-making [4].

Several provenance models are available, and the PROV-SwProcess model [5] serves as a reference for the system proposed in this study. This model, an extension of the W3C PROV standard [6], has been developed to capture software process provenance data and can also be used to share these data in heterogeneous environments [7]. However, for preexisting software process provenance data that were not generated in alignment with this model, it is necessary to develop a *wrapper* for each application or usage context of the PROV-SwProcess model [5] [8] (or for each specific software process data format). This fact ensures the alignment between the adopted data model and the PROV-SwProcess model [8]. In this vein, this work aims to prototype, develop, and validate the PROV-SwSystem, a web-based system to capture, store, and analyze software process data with its provenance and according to the PROV-SwProcess model [5] specifications. The proposed system is particularly relevant in scenarios where specialized tools for collecting software process data are unavailable, whether in academic or industrial contexts. It facilitates the manipulation and analysis of the collected data, thus contributing to the continuous improvement of software processes.

The structure of this paper is as follows: Section 2 presents related work. Section 3 details the methodology for the system development. Section 4 describes the design and implementation of the PROV-SwSystem. Section 5 presents the obtained results. Finally, Section 6 outlines the conclusions and final remarks, followed by the acknowledgments and references.

2 RELATED WORK

This section briefly discusses tools that provide some functionality to contribute to traceability, analysis, and continuous improvement in workflows or software development processes.

The YesWorkflow [9] enables the capture of data provenance in scientific workflows without requiring modifications to the source code. It enables the visualization of data flow and the traceability of the activities performed. However, it does not have a focus on software development processes.

The noWorkflow [10] tool was developed to capture metadata and provenance information from Python code executions, without the need for prior instrumentation. It allows you to track code changes and analyze the evolution of experiments. In addition to not focusing on software development processes, it only considers code executions in Python.

Finally, a tool-supported approach called iSPuP [5] allows PROV-SwProcess model instantiation, new information inferencing, and data visualization. However, this tool does not allow the capture of software process provenance data directly. It uses/requires specific wrappers to transform the data aligned to the model. To meet this gap, PROV-SwSystem fits in, allowing direct insertion and subsequent manipulation and analysis of these data.

3 METHODOLOGY

The focus of this paper is on the design, implementation, testing, and future project perspectives of PROV-SwSystem, a web-based system adherent to the PROV-SwProcess provenance model [5] [8]. Thus, grounded in the Design Science Research (DSR) methodology [11], which provides a set of guidelines and specific methods for the creation, construction, and validation of an artifact in the context of IT innovation, the following steps were considered:

- `Problem Identification`: Analysis of the needs for capturing and analyzing software process data; identification of the limitations of existing tools and the gap that the PROV-SwProcess model can address.
- `Definition of Expected Outcomes`: Specification of the project objectives, including the creation of a tool that allows direct insertion and subsequent manipulation and analysis of software process data according to the PROV-SwProcess model. Following the definition of the expected outcomes, internal meetings were held to detail the system definitions.
- `Design and Development`: Incremental development of the system, starting with prototypes and evolving into a functional version; This step also includes the development and validation of the use case diagram, along with continuous improvements to the overall system design; additionally, coding and testing of the code produced during the system's implementation were carried out. During the transition from this stage to the next, the deployment diagram was also modeled and validated.
- `Demonstration`: Application of the tool in real-world scenarios to validate its functionality and usability; collection of user feedback to identify areas for improvement. Weekly simulations of all stages of the system were conducted, along with the completion of each system page.
- `Evaluation`: Test planning and execution (unit tests, system tests, and usability tests) to evaluate the effectiveness of the tool; analysis of the collected data to verify whether the system objectives were achieved. It is important to highlight that a consent form was used during the testing phase, ensuring that the collected information was used only for academic and scientific purposes and following all the guidelines of Law No. 13,709/2018 [12] (the Brazilian General Law on the Protection of Personal Data - LGPD).
- `Communication`: Documentation of the obtained results considering the development of the system; dissemination of the results through scientific publications and conference presentations.

4 DESIGN AND IMPLEMENTATION OF PROV-SWSYSTEM

The requirements elicitation for PROV-SwSystem resulted in the creation of medium-fidelity prototypes. An example is shown in Figure 1. Medium-fidelity prototypes, when compared to low-fidelity prototypes, become more realistic, however, both types of prototypes do not represent the final product, but are ideal for validating the ideas, testing features, and facilitating communication between the team, customers, and users, before full development [13].

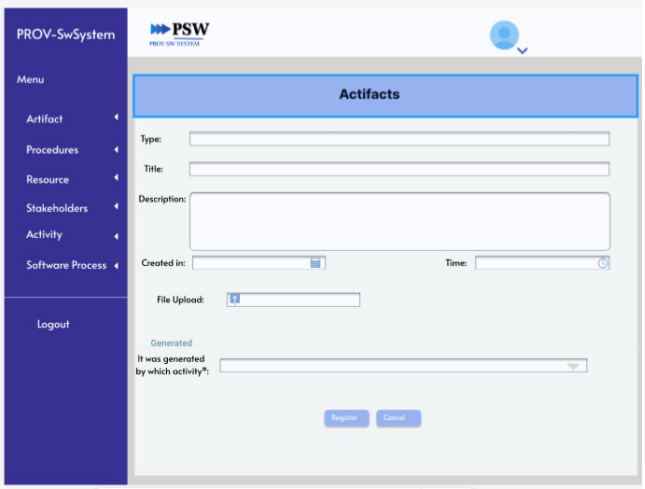

Figure 1: Prototype of the Artifact Registration Page, created in Figma[1]

For PROV-SwSystem prototyping, interfaces were established considering the requirement for the registration of all the PROV-SwProcess classes [5], such as Artifacts, Procedures, Software Processes, Resources, Stakeholders, and Activities. In addition, interfaces for user registration and login were produced. The Figma[2] application was used for creating these prototypes, enabling the subsequent validation of the pages designs.

For modeling the requirements to be addressed in the software, a use case diagram was used, according to Figure 2. The developed use case diagram includes a single actor, the "Stakeholder". The actors represent the interested parties in the system, and considering the PROV-SwSystem, they are the Stakeholders. These Stakeholders may be personal (e.g., individual developers), teams (e.g., development teams), or organizations (e.g., developer companies).

[2] https://help.figma.com/hc/pt-br/articles/14563969806359-O-que-C3%A9-o-Figma

Figure 2: Use Case Diagram of the PROV-SwSystem.

It is important to note that PROV-SwProcess stakeholders are "agents involved, interested, or affected by the activities of the software process" [5], and, in Figure 2, the stakeholder actors are users who interact with the system and wish to perform any actions described in the use cases.

The use case diagram includes 41 use cases, as shown in Figure 2. The use cases illustrate each interaction between the system and users or other systems. In PROV-SwSystem, the use cases can be grouped into five major "actions" as follows: I) Registration and login to the system, II) Registration of entities, agents, and activities, III) Search for entities, agents, and activities, IV) Data modification (edit, delete, and invalidate) and V) Activity finalization. The arrangement in the system functionalities model is focused on maintaining the entities, agents, and activities, with the possibility of extending to registrations or searches for these items. When performing a search, the actor (user) can edit or remove the retrieved registers. Editing the data of Artifacts and Procedures automatically invalidates the previous record that was edited, replacing it with the new register.

In addition to the use case diagram model, still in the stage of modeling the resources to be used for software development, the deployment diagram was produced, as illustrated in Figure 3. This diagram demonstrates the execution architecture of a system, focusing on the physical allocation of a software system in a hardware environment [14]. Furthermore, the creation of this diagram elucidates how the components interact with each other and the location where each software artifact should be installed, recorded, and documented throughout the structure.

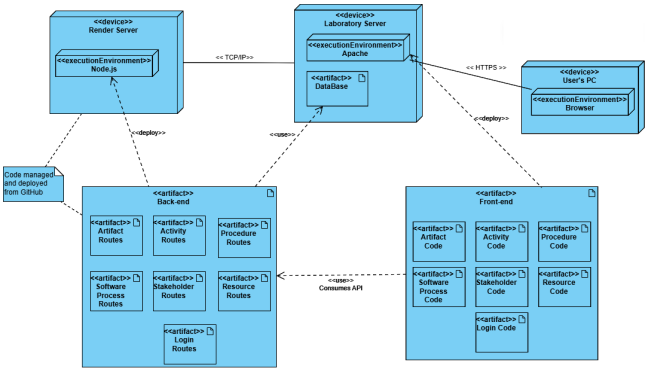

Figure 3: Deployment Diagram of the PROV-SwSystem [3].

According to Figure 3, the "Laboratory Server" contains the Apache execution environment and hosts the PROV-SwSystem database. The server is configured to function as both a file server and a server for providing the front-end for the project's online domain. It includes two artifacts: the "Database" and the "Front-end", which contains the code artifacts for each system page and is connected to the server through a deployment relationship, meaning the front-end is deployed on the Laboratory Server. On the "Render Server", which is a remote hosting server where the system's back-end is deployed, the Node.js runtime environment is located. In other words, the back-end was developed and is being executed in a Node.js environment. On this server, the back-end code is managed and deployed directly from the GitHub repository to the Render[4] server, which serves as the hosting location. This relationship was noted in the diagram. The Render Server contains one artifact, "Back-end", which encompasses all the artifacts of the back-end APIs and is connected to the server through a deployment relationship, meaning the back-end is deployed on the Render Server. The front-end is connected to the back-end via a "use" connection, meaning the front-end consumes the back-end APIs. Additionally, the back-end "uses" the database hosted on the "Laboratory Server". In addition to these two nodes, there is the "User's PC" node, which represents the end user's computer and includes the "Browser" runtime environment to access the system's front-end and interact with the software. The "User's PC" is connected to the "Laboratory Server" via the HTTPS protocol, providing a secure connection between the user's browser and the server. Communication between the "Render Server" and the "Laboratory Server" is carried out via the TCP/IP protocol.

During the development of PROV-SwSystem, each entity, agent, and activity was implemented and organized into separate folders, with the backend placed in a shared directory. The main menu was managed by the "index" file located in the root directory. The project structure followed the principles of the MVC (Model-View-Controller) architecture to ensure task separation and code organization [16].

4.1 Front-End Development

In terms of front-end design, the files for the registration screen and the information listing screen are stored in a shared folder. HTML5[5] was chosen to structure the content of the page, CSS3[6] for styling and layout, and the Bootstrap 5 framework [7] to facilitate the organization and styling of pages, taking responsiveness criteria into account. To enhance user interaction on the page and make it more dynamic, in addition to collecting registered data, JavaScript[8] was incorporated alongside the JQuery library[9] to manage events, display pop-up windows and communicate with servers.

The default template for PROV-SwSystem pages consists of a fixed header section, a dynamic and responsive navigation bar, and a form area for input data. The header includes the system's logo and a connection to the side navigation. The navigation bar contains options for users to start a project, view their profile, manage/logout of the session, and switch languages, with English, Portuguese, and Spanish as options. The main area displays fields where users can interact with the platform and input data, and at the end, the user can choose, via buttons, whether to register or cancel the operation, returning to the main menu.

To store artifact and procedure files, the Github[10] API was chosen as a version control manager [17]. Figure 4 presents the layout of the Artifact page, which includes a field for adding files in all permissible formats. Upon clicking the "register" button, the system invokes the "commitFile" function, which sends an AJAX request

[4] https://render.com/
[5] https://html.spec.whatwg.org/multipage/
[6] https://www.w3.org/Style/CSS/
[7] https://getbootstrap.com/
[8] https://developer.mozilla.org/en-US/docs/Web/JavaScript
[9] https://jquery.com/
[10] https://docs.github.com/en/rest?apiVersion=2022-11-28

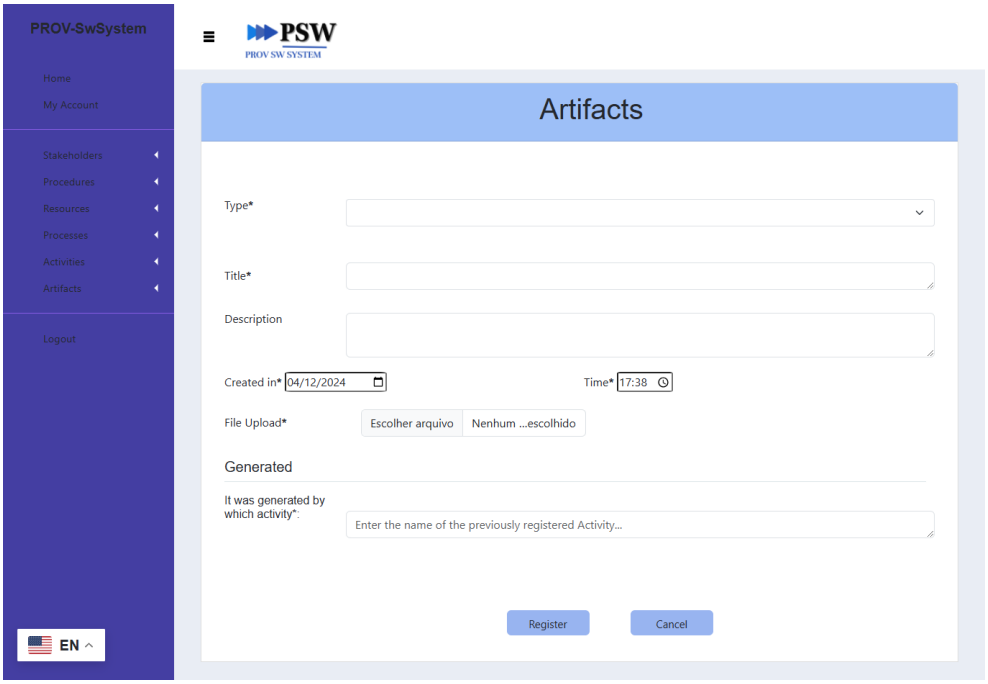

Figure 4: Implementation of the Artifact Registration Page

to the GitHub Rest API, transmitting the file and the "SHA" key collected via the "addEventListener" instantiation method [18]. This method adds a function or object to the event listener list. When an event of the specified type occurs, if the file has the same name, the commit is not modified; if the files are different, the system replaces or inserts a new entry.

The layout for listing database information is configured to display details of registered events, allowing users to view them on-screen and make modifications or deletions as needed. When performing edit or delete actions, confirmation is requested to ensure the user's intent before proceeding. After each edit or delete action, the table is automatically updated using the "loaddata" function to fetch updated data from the back-end for display on the screen.

4.2 Back-End Development

Considering the PROV-SwSystem back-end, Node.js was used in combination with the Express framework to manage web servers. Node.js was chosen for its ability to create server-side JavaScript applications and access a variety of libraries and tools required to keep the website operational online [19]. Express was selected for its capability to simplify route definitions and organize HTTP requests [20]. Additionally, some features are enabled through middleware, such as "Cors" and "body-parser," which manage access and data parsing within requests.

Furthermore, the "mysql2" library [11] was used to interact with the MySQL database, as it supports SQL queries and asynchronous interactions. API routes and HTTP requests are directed to basic CRUD (Create, Read, Update, Delete) operations using the "POST", "GET", "PUT" and "DELETE" methods, respectively, and are executed asynchronously. On the front-end, AJAX methods were utilized to "call" these CRUD functions.

By applying the principles of the MVC architecture[16] in the project construction, the back-end files followed the structure specified by the pattern: the "View" files request some functionality from the back-end API. This request is sent to the "Controller", which filters the authorized requests to access the database and forwards them to the "Model," where it interacts with the database records according to the request. Upon completing the query task and returning the value, it sends the result back to the "Controller" where the data collected from the database is organized and only the necessary information is sent to the user. When sent to the user, this data is received by the Views, which finalizes the requested action. This is a cyclical process, returning to the starting point and restarting the process with each iteration of the API.

To ensure the consistency of the analysis of the data collected by the system, data deletions are handled as an "edit" in the MySQL database, where a boolean field named "deleted" changes from 0 (active data) to 1 (inactive data). Additionally, an edit is treated as a new "insertion" or registration in the database, which deactivates the previous entry.

4.3 Software Configuration Management

During the life cycle of a piece of software, the system will require maintenance, modifying the previously defined life cycle [14]. Therefore, to develop secure systems, it is necessary to use software configuration management (SCM) techniques, seeking to maintain an organized history of all modifications and old versions [21]. In this vein, SCM can be defined as an area of Software Engineering

[11] https://www.npmjs.com/package/mysql2

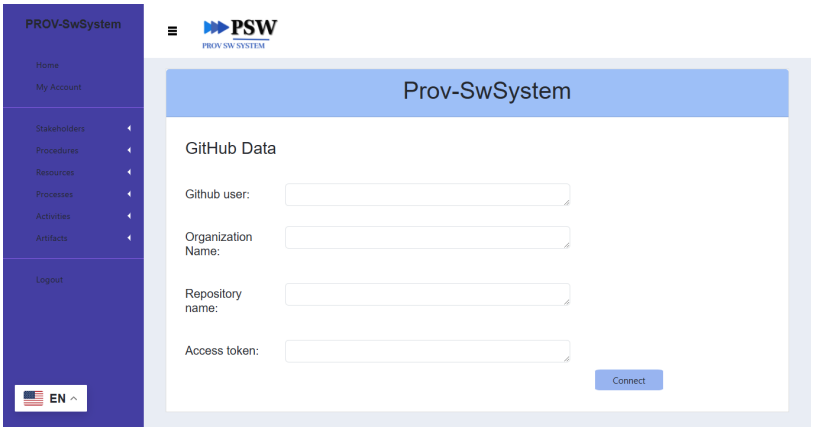

Figure 5: Software Configuration Management of the PROV-SwSystem

capable of establishing consistent attributes of software products throughout their life, as well as maintaining the system's traceability and quality during development [21, 22].

In order to maintain this traceability, SCM techniques were implemented in the PROV-SwSystem. GitHub [12] software with a version control system was chosen to carry out this action, and the "GitHub REST API"[13] was used to connect to PROV-SwSystem, interacting with the resources provided by GitHub, such as commits, and the Octokit library, to carry out API requests and action calls.

When inserting the SCM into the PROV-SwSystem, the user must first register on the GitHub platform, create an organization, a repository within that organization, and an access key (token) in order to insert the information into the PROV-SwSystem, as shown in the Figure 5. This access key is unique to each profile, with a single token being valid for any repository and any organization the user has access to. The purpose of using Git organizations to manage file versioning lies in enabling user teams to collaborate on software development and allowing other users to access, use, and modify the files that members of this organization upload to the organization's repository. Once this function has been created in the system, users can automatically insert new artifacts into the registered GitHub repository, keeping track of development and the stakeholders involved in the project simultaneously.

4.3.1 SCM Database Model. For the development of the PROV-SwSystem, PROV-SwProcess database model was adapted, adding some tables to enable software configuration management to the system. As shown in Figure 6, the relationship added by the SCM involves the "User" and "Github" tables and the relational table "hasOrganization", which organizes the relationships between users and GitHub organizations/repositories. The "Github" table was created in the system for the SCM and includes information about organizations and repositories, such as "idOrganization" (primary key), "nameOrganization" and "nameRepository." The "User" table stores user data, including "idUser" (primary key), "name", "password", "login", "github_userName" and "token_access". This table already existed in the PROV-SwProcess database model but did not include the attributes "github_userName" and "token_access", which are fully linked to GitHub.

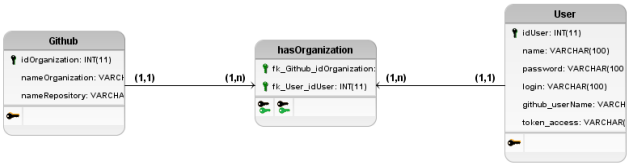

Figure 6: Software Configuration Management Database Model

The relationship between the tables Github and User is defined by the "hasOrganization" table, which uses two foreign keys: "fk_Github_idOrganization" and "fk_User_idUser". These two foreign keys form a composite primary key in the relational table. The cardinality of this relationship is expressed in the diagram: between the "Github" and "hasOrganization" tables, the cardinality is (1,n), meaning that each organization/repository must be associated with at least one user at the time of its creation and can later be associated with multiple users. Between the "User" and "hasOrganization" tables, the cardinality is (1,n), where a user must be linked to at least one organization to access the system and its functionalities and can later associate with multiple organizations. In summary, an organization is created only after a user logs in, being immediately associated with them, while allowing other users to associate with the same organization later.

5 RESULTS AND DISCUSSION

The results obtained, in addition to the implemented tool, are some code reviews and initial tests. Considering the initial phase of the development, the execution of the code was limited. Then, code reviews were performed by the project members, making it possible to identify six unused variables: one variable for controlling outdated activities, two variables related to the GitHub API for accessing API return data, and three environment variables for tokens.

[12] https://github.com/
[13] https://docs.github.com/en/rest?apiVersion=2022-11-28

These variables became redundant due to code changes but were not removed. Furthermore, five redundant functions were identified, primarily related to data control functions across multiple entities, activities, or agents, including: 1) **"inserirArchive"**, which was present on the procedure and artifact pages and registered the name of the attached file in the database, but the same function was implemented in both cases; 2) **"buscarActivity"**, which searched for an activity in the database based on user input but was implemented redundantly on the Artifact, Activity, and Stakeholder pages. It was retained only on the Activity page and reused in the others; 3) **"buscarActivityCorrect"**, which searched for the full name of an activity after selecting options, present on the Artifact and Stakeholder pages; 4) **"buscarUmStakeholder"**, used on the software process page to search for a stakeholder by name, even though the Stakeholder functions already included "buscarUm", which performed the same operation; 5) **"excluir"**, present on all pages, even though users do not delete data in the database but only deactivate it. Since database records are reused for new entries, deleting data could lead to users removing information used by other entities or activities. To resolve all identified issues, existing functions were reused by linking the files and removing redundant or unused variables.

With a significant portion of the system developed, functional tests were conducted. These tests were aimed at analyzing the functional behavior of the system, ensuring that all stages, buttons, and fields worked correctly and that the outputs matched the expected results [23]. Usability testing and system testing were adopted for this purpose, all performed manually or with the assistance of web tools.

The purpose of system testing was to identify failures and weaknesses in the system as a whole, considering software, hardware, and database [23]. The test results revealed some issues related to delayed responses on certain screens. This delay was caused by the high volume of code and functions, requiring the application of algorithm analysis and complexity concepts to optimize the code. Furthermore, six routing errors were identified, all returning a 404 “Not Found” response. These errors occurred because of the system’s failure to properly process all incoming requests.

For usability testing [24], the system was handed over to an external student, allowing the identification of fifteen navigation flow issues, eight attempts to register data in inappropriate sections, and six failures in confirming completed registrations. The test was carried out exploratoryly, which means it was a manual test that allowed the tester the freedom to explore the system, as test cases were not predefined in a test plan [25]. Among the navigation issues, there was confusion caused by the failure to list records in the "list" field, and six attempts to click incorrect buttons. These actions misled the user into believing that the entered data was being registered, whereas, in reality, the data was not registered, and the system redirected to another page. Of the eight attempts to register data in inappropriate sections, most involved attempts to input information into fields intended for searches, highlighting that these fields were not easily understandable. Regarding the six failures in confirming completed registrations, they occurred when the user selected the registration option, but no feedback was provided, which left the user uncertain about whether the data had been successfully registered.

Together with those involved in this project, adjustments and improvements were made, such as fixing some poorly performing functionalities, search buttons that were not working, and difficulties in system handling. Data entry validations were implemented on both the front-end and back-end, along with interface improvements by grouping information into a single page and minimizing content on others. The results were satisfactory, contributing to the continuous improvement of the tool and paving the way for its use in high-demand environments soon.

Based on the test results, it became clear that improving the security of PROV-SwSystem was necessary. Consequently, several security improvements were implemented, including improved database access control and mitigation measures against XSS attacks.

Identified security vulnerability was the direct execution of SQL queries using user-provided values, which could potentially allow malicious SQL injection attacks [26]. To mitigate this issue, all database queries were refactored to utilize parameterized queries, where placeholders ’?’ are employed to ensure that user inputs are treated strictly as data rather than executable SQL statements.

Furthermore, the risk of Cross-Site Scripting (XSS) [27], an attack method that injects malicious scripts into web pages—unlike SQL Injection, which targets database queries—was identified as a potential vulnerability in the system.

In the PROV-SwSystem, data retrieved from the API was inserted directly into the HTML without proper sanitization, rendering the system vulnerable to such attacks. An attacker could inject malicious scripts into input fields, which would then be stored and executed when displayed in the user interface. To address this issue, user input sanitization was applied before storing data in the database. Additionally, on the frontend, the DOMPurify library [28] was implemented to cleanse any potentially malicious input, thereby mitigating XSS risks and enhancing the system’s overall security posture.

6 FINAL REMARKS

This paper describes the prototyping, developing, and validating of a web system for collecting software process provenance data, considering the PROV-SwProcess model. For this purpose, requirement elicitation and modeling diagrams were employed, in addition to the deployment diagram and database models, along with essential concepts related to software design and testing. The system was implemented using technologies such as JavaScript, Bootstrap, Node.js, and jQuery, structures based on the principles of MVC architecture and specifications of the Express framework.

After the development stage, the tool was deployed to production environments using two servers for hosting: one for the back-end (Render server) and another for the front-end (Laboratory server), enabling testing of the tool and validation. The primary focus of testing was to identify any issues related to design, implementation, integration between servers and user experience in web browsers. Code reviews, system and usability tests were conducted to achieve this.

Integration with concepts associated with configuration management was implemented with the aim of maintaining a history of previous system versions and all changes that may still occur.

For this version control, the GitHub platform was selected due to prior knowledge of the tool's functionalities, and the ease of using the API.

The identified issues were resolved, and the initial version of the PROV-SwSystem is now online and accessible via the project's domain.

To increase awareness and usage, students at the institution will be encouraged to incorporate the platform into their classes to manage software process data.

Future perspectives include expanding the tool for use in real-world environments. Another expectation is the development of support features to better integrate users into the system, enabling them to maximize its potential.

ACKNOWLEDGMENTS

We want to thank the National Council for Scientific and Technological Development (CNPq), the Federal Center for Technological Education of Minas Gerais (CEFET-MG), the Undergraduate Directorate of CEFET-MG, and the Laboratory for Scientific Initiation and Extension of Computing (LINCE) for their support in developing this work.

REFERENCES

[1] Mark C. Paulk. A history of the capability maturity model for software. *ASQ Software Quality Professional*, 12(1):5–19, December 2009. Available at: https://personal.utdallas.edu/~mcp130030/papers/p2009c.pdf. Accessed on: March 2025.

[2] Marco Tulio de Oliveira Valente. *Modern software engineering: Principles and Practices for Productive Software Development.* Universidade Federal de Minas Gerais, Minas Gerais, Brasil, 2020. Available at: http://hdl.handle.net/1843/37905. Accessed on: March 2025.

[3] Larissa Pereira Moura and Cejana Marques Borges. Information systems: Management toolsat a logistics company in palmas-to. *Revista Ibero-Americana de Humanidades, Ciências e Educação*, 9(6):684–704, June 2023. Available at: https://periodicorease.pro.br/rease/article/view/10252. Accessed on: March 2025.

[4] Victor R. Basili, Dieter Rombach, and Kurt Schneider. *Empirical Software Engineering Issues. Critical Assessment and Future Directions.* Springer Berlin, Heidelberg, 1 edition, 2007. Accessed on: March 2025.

[5] Gabriella Castro Barbosa Costa, Claudia Werner, Regina Braga, Eldânae Nogueira Teixeira, Victor Ströele, Marco Antônio Pereira Araújo, and Marcos Alexandre Miguel. Design, application and evaluation of prov-swprocess: A prov extension data model for software development processes. *Journal of Web Semantics*, 71:100676, 2021. doi: https://doi.org/10.1016/j.websem.2021.100676. Available at: https://www.sciencedirect.com/science/article/pii/S1570826821000512. Accessed on: March 2025.

[6] W3C. Prov-overview. W3C Recommendation, April 2013. Available at: https://www.w3.org/TR/prov-overview/. Accessed on: March 2025.

[7] Gabriella Costa, Eldânae Teixeira, Cláudia Werner, and Regina Braga. A proposal for sharing software process provenance data in heterogeneous environments. In *Anais do I Workshop de Práticas de Ciência Aberta para Engenharia de Software*, pages 37–42, Porto Alegre, RS, Brasil, 2021. SBC. doi : 10.5753/opensciense.2021.17144. Available at: https://sol.sbc.org.br/index.php/opensciense/article/view/17144. Accessed on: March 2025.

[8] Gabriella Castro Barbosa Costa Dalpra and Lince Online. Prov-swprocess: A prov extension data model for software development processes, 2021. Available at: https://prov.linceonline.com.br/modelo/index.html. Accessed on: March 2025.

[9] Timothy McPhillips, Tianhong Song, Tyler Kolisnik, Steve Aulenbach, Khalid Belhajjame, R Kyle Bocinsky, Yang Cao, James Cheney, Fernando Chirigati, Saumen Dey, Juliana Freire, Christopher Jones, James Hanken, Keith W Kintigh, Timothy A Kohler, David Koop, James A Macklin, Paolo Missier, Mark Schildhauer, Christopher Schwalm, Yaxing Wei, Mark Bieda, and Bertram Ludäscher. YesWorkflow: A user-oriented, language-independent tool for recovering workflow information from scripts. *Int. J. Digit. Curation*, 10(1):298–313, May 2015. Available at: https://arxiv.org/abs/1502.02403. Accessed on: March 2025.

[10] Leonardo Murta, Vanessa Braganholo, Fernando Chirigati, David Koop, and Juliana Freire. NoWorkflow: Capturing and analyzing provenance of scripts. In *Lecture Notes in Computer Science*, Lecture notes in computer science, pages 71–83. Springer International Publishing, Cham, 2015. Available at: https://link.springer.com/chapter/10.1007/978-3-319-16462-5_6. Accessed on: March 2025.

[11] Ken Peffers, Tuure Tuunanen, Marcus A. Rothenberger, and Samir Chatterjee. A design science research methodology for information systems research. *MIS Quarterly*, pages 45–77, December 2014. doi: 10.2753/MIS0742-1222240302. Available at: https://www.tandfonline.com/doi/abs/10.2753/MIS0742-1222240302. Accessed on: March 2025.

[12] Brasil. Lei nº 13.709, de 14 de agosto de 2018: Lei geral de proteção de dados pessoais (lgpd), 2018. Available at: https://www.planalto.gov.br/ccivil_03/_Ato2015-2018/2018/Lei/L13709.htm. Accessed on: March 2025.

[13] Jennifer Preece, Yvonne Rogers, and Helen Sharp. *Interaction Design: Beyond Human-Computer Interaction.* John Wiley and Sons, 3rd edition, 2011. Accessed on: March 202.

[14] Roger S. Pressman and Bruce R. Maxim. *Software Engineering: A Practitioner's Approach.* AMGH, Porto Alegre, Brazil, 9 edition, 2021. Technical review by Reginaldo Arakaki, Julio Arakaki, and Renato Manzan.

[15] Inc. Object Management Group. Omg unified modeling language (omg uml) version 2.5.1., 2017. Available at: https://www.omg.org/spec/UML/2.5.1/About-UML. Accessed on: March 2025.

[16] Francisco Moreira Calado Souza, Edilson Carlos Silva Lima, and Elda Regina de Sena Caridade. Creating a scalable scheduling system using typescript with nestjs on the backend and nextjs on the frontend. *Revista Ibero-Americana de Humanidades, Ciências e Educação*, 8(12):43–57, 2022. Available at: https://periodicorease.pro.br/rease/article/view/7986. Accessed on: March 2025.

[17] GITHUB. About the rest api, June 2024. Available at: https://docs.github.com/en/rest/about-the-rest-api?apiVersion=2022-11-28. Accessed on: March 2025.

[18] MDN WEB DOCS. Eventtarget: addeventlistener() method, April 2024. Available at: https://developer.mozilla.org/en-US/docs/Web/API/EventTarget/addEventListener. Accessed on: March 2025.

[19] OPENJS FOUNDATION. Introduction to node.js, March 2025. Available at: https://nodejs.org/en/learn/getting-started/introduction-to-nodejs. Accessed on: March 2025.

[20] EXPRESS. Using express middleware, March 2025. Available at: https://expressjs.com/en/guide/using-middleware.html. Accessed on: March 2025.

[21] Mariana Magalhães Malhone and Mônica Frigeri. A importância do gerenciamento de configuração para o ciclo de vida do software: um estudo de caso baseado nas diretrizes da engenharia de software. *RBTI-Revista Brasileira em Tecnologia da Informação*, 3:1–60, 2021. Available at: https://www.fateccampinas.com.br/rbti/index.php/fatec/article/view/29. Accessed on: June 2024.

[22] Ian Buchanan. Gerenciamento de configuração: definição e benefícios. Available at: https://www.atlassian.com/br/microservices/microservices-architecture/configuration-management. Accessed on: March 2025.

[23] Ian Sommerville. *Engenharia de Software.* Pearson Prentice Hall, São Paulo, 10th. edition, 2011. Page 209. Available at: https://www.bvirtual.com.br/NossoAcervo/Publicacao/168127. Accessed on: March 2025.

[24] Francisco Dione de Souza Amâncio. *Um mapeamento sistemático da literatura de testes de usabilidade em aplicações móveis.* PhD thesis, Universidade Federal do Ceará, Campus Quixadá, Quixadá, CE, BRA, 2012. TCC (graduação em Sistemas de Informação). Available at: http://www.repositorio.ufc.br/handle/riufc/25259. Accessed on: March 2025.

[25] Igor Ernesto Ferreira Costa, Sandro Ronaldo Bezerra Oliveira, Lucas Felipe Ferraro Cardoso, Ana Isabela Manito Ramos, and Rafael Nascimento de Sousa. Uma gamificação para ensino e aprendizagem de teste exploratório de software: Aplicação em um estudo experimental. In *Proceedings of SBGames 2019 — Education Track – Short Papers*, Belém, PA, Brasil, 2019. Universidade Federal do Pará – UFPA, SBC. Programa de Pós-graduação em Ciência da Computação (PPGCC) – Faculdade de Computação (FACOMP) – Instituto de Ciências Exatas e Naturais (ICEN). Available at: https://www.sbgames.org/sbgames2019/files/papers/EducacaoShort/196894.pdf. Accessed on: March 2025.

[26] Gustavo P. Lages and Rafael T. Pereira. Estudo comparativo entre técnicas de detecção e prevenção de ataques de injeção sql. *Anais da XVII Escola Regional de Banco de Dados*, August 2022. doi: https://doi.org/10.5753/erbd.2022.223544. Available at: https://sol.sbc.org.br/index.php/erbd/article/view/21410/21234. Accessed on: March 2025.

[27] Lucas Matheus Gonçalves Faculdades and Leticia Maria de Oliveira Camenar. Exploração de vulnerabilidades cross-site scripting: Uma análise das principais técnicas de ataques xss. In *Anais do Congresso Latino-Americano de Software Livre e Tecnologias Abertas*, May 2021. Available at: https://sol.sbc.org.br/index.php/latinoware/article/view/19909/19737. Accessed on: March 2025.

[28] Mario Heiderich, Christopher Späth, and Jörg Schwenk. DOMPurify: Client-Side Protection Against XSS and Markup Injection. In *Proceedings of the European Symposium on Research in Computer Security (ESORICS 2017)*, volume 10493 of *Lecture Notes in Computer Science (LNCS)*, pages 116–134. Springer, Cham, 2017. Available at: https://link.springer.com/chapter/10.1007/978-3-319-66399-9_7. Accessed on: March 2025.