Towards a Web Portal for Teaching and Practicing of Software Engineering Controlled Experiments

Fernando S. Grande[1], **André F. R. Cordeiro**[1], **Edson OliveiraJr**[1]

[1]Departamento de Informática – Universidade Estadual de Maringá (UEM)
Maringá – PR – Brazil

fsg0314@gmail.com, cordeiroandrefelipe@gmail.com, edson@din.uem.br

Abstract. *Experimentation in Software Engineering is vital for improving methods and processes, but replication challenges and the lack of structured repositories limit progress. While models and ontologies organize knowledge, they do not provide integrated environments for teaching and practice. This paper presents a web portal for controlled experiments in Software Engineering, structured on a experimental process and supported by an experimentation ontology. It enables experiment registration, storage, and retrieval through a search mechanism. Tests on database connectivity, validation, and end-to-end functionality confirmed effectiveness. Future work includes enhancing usability, expanding the repository, and adding advanced analysis features.*

1. Introduction

Software Engineering (SE) encompasses the study and application of processes for software development and maintenance [Sommerville 2015]. Key subareas of SE include Distributed Software Development [Robinson 2019], Software Maintenance [Haldar and Capretz 2024], and Experimentation in Software Engineering (ESE) [Juristo and Vegas 2016]. The ESE subarea focuses on studying and implementing experimental practices that enhance the development and improvement of maintenance methods, tools, and processes [Kitchenham 2016]. By refining the quality of experiments, researchers can broaden the knowledge base related to specific research topics [Wohlin 2012].

Formalization is a characteristic that enhances the execution of experiments [Freire et al. 2014]. In addition to the initial execution, it is essential to consider the repetition, replication, and reproduction of the experiment [Gonzalez-Barahona and Robles 2023]. The literature presents different solutions to assist in these activities [Anchundia 2020], such as conceptual models and ontologies related to SE experiments [Scatalon et al. 2011, Furtado 2018, Luz et al. 2020, Vignando et al. 2020].

Considering formal structures for representing fundamental concepts and their relationships, solutions can utilize the established semantic relationships, such as Recommendation Systems [Vignando 2020]. In addition to these systems, one can also think of Software Portals. A portal can be defined as a structure that enables access to a certain type of information [Mead and McGraw 2005].

The adoption of portals can be viewed as a solution across various contexts [Correa 2018, Kipping et al. 2016, Brito 2014, Adeyinka and Bashorun 2011]. Given the unique characteristics of a portal, it is particularly suitable for use in the context of ESE. This paper discusses the development of a web portal designed to support the teaching, learning, and application of controlled experiments in SE, representing a novel contribution. One portal with these characteristics was not found in the literature.

This portal was developed based on an ontology built for the context of ESE, which provides a framework for planning and executing experiments in SE[Vignando 2020]. We anticipate that the web portal will allow researchers to input experiments, search for existing experiments, and gather data for purposes such as repetition, replication, and reproduction.

This paper outlines the process involved in constructing the portal. The subsequent sections detail the development stages. Section 2 discusses the essential research studies necessary for the portal's development. Section 3 covers the specifics of the portal project. Section 3.4 describes the testing process for the portal and the outcomes of these tests. Section 6 summarizes the conclusions drawn from the study and presents directions for future work.

2. Background

The literature on ESE presents various artifacts that can facilitate the execution of experiments. Possible artifacts include Experimental Templates [Juristo and Moreno 2013, Wohlin 2012, Jedlitschka et al. 2008, Kitchenham et al. 2008, Wohlin et al. 2000] and Experimental Process [Wohlin 2012, Wohlin et al. 2000]. Experimental Templates can be defined as documents that describe what information should be recorded regarding experiments. Possible information includes limitations and threats to validity [Wohlin 2012, Wohlin et al. 2000]. Regarding the Experimental Process, it is observed that the process established by Wohlin is widely applied, according to Furtado [Furtado 2018].

The process illustrated in Figure 1 is structured based on the following activities: Experiment Scoping, Experiment Planning, Experiment Operation, Analysis and Interpretation, Presentation, and Package.

The Experimental Process described by Wohlin et al. is composed by the following activities: **Experiment Scoping** describes the objective, problem, and hypothesis for the experiment. **Experiment Planning** presents the definition of the methods, techniques, resources, variables, and the elaboration of a guide for the experimental activities. The **Experiment Operation** describes the steps to conduct the execution of the experiment. Data validation is also conducted. **Analysis and Interpretation** consider descriptive statistics in the data collected. The data can be interpreted and reduced. After that, one hypothesis test is executed to determine whether the hypothesis will be accepted or rejected. Finally, conclusions will be drawn to define how the experiment results will be used. The **Presentation and Package** considers the documentation of the results. A package that contains the experimental data for future repetitions, replications, and reproductions of the experiment can be organized.

These activities can guide the researcher during the planning and execution of an experiment [Wohlin 2012]. The experimental process has been applied in different research contexts within ESE, such as quality. The study of Furtado presents guidelines related to quality assessment of quasi-experiments and controlled experiments in Software

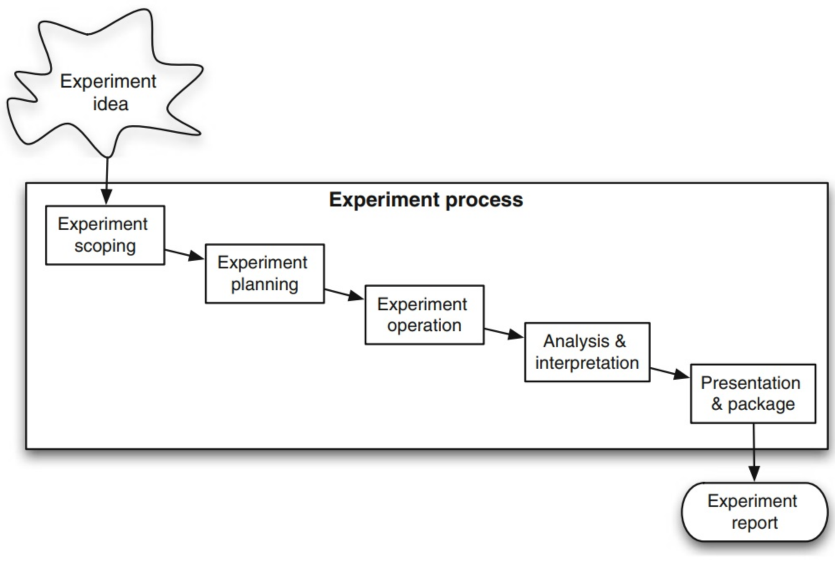

Figure 1. Experiment Process, designed by Wohlin et al. [Wohlin 2012]

Product Line (SPL) [Furtado 2018]. A conceptual model was developed to support the defined guidelines. Another study presents the creation of an ontology to formally support experiments in SPL [Vignando 2020].

3. Portal Project

The Web Portal started with a project. The information about the project was collected and organized in terms of Research Methodology, Development Methodology, Requirements, Technologies Used, and the Interface Prototype.

3.1. Research Methodology

The development of the portal started with a non-systematic literature review on topics relevant to the portal. Next, exploratory studies were carried out to evaluate the web development technologies. These studies were important for choosing technologies and developing algorithms for constructing the portal, as well as its interface and search mechanism. After this, the portal was implemented, including its interface and search mechanism. Subsequently, tests were conducted to identify potential defects and possible improvements. Finally, an empirical evaluation of the project was planned and executed. The activities carried out can be summarized as follows:

- Conducting a literature review on relevant topics for the portal.
- Performing exploratory studies on web development technologies.
- Selecting appropriate technologies for the project.
- Developing algorithms for the portal's construction.
- Designing and implementing the portal's interface.
- Developing and integrating the search mechanism.
- Implementing the portal with all selected features.
- Conducting tests to verify requirements and defects.
- Performing an empirical evaluation of the project.

3.2. Development Methodology

The requirements and the technologies were defined to develop the portal. Also, a prototype of the interface was designed. Finally, through the elaborated project, the portal was constructed.

The development process involved iterative activities in which the requirements guided the design decisions and the choice of technologies. Initially, the general and functional requirements oriented the definition of the portal layout, the organization of navigation elements, and the construction of the repository page. At the same time, non-functional requirements such as clarity, simplicity, and consistency were considered in the interface. Once the requirements were defined, the technologies were choosen. The development was carried out incrementally. Each portal functionality was implemented and integrated into the portal, ensuring alignment with the initial requirements.

3.2.1. Requirements

The requirements considered by the first version of the web portal can be classified as functional or non-functional.

3.2.1.1. Functional Requirements

1. **Repository Page**: the portal has a dedicated page to present all experiments available on the portal.
2. **Simple Search**: the portal is equipped with a search feature that enables users to find experiments using keywords or phrases, returning all the results related to that entry.
3. **Data Storage**: the portal's data, including all experiment details, is stored and managed in a database.

3.2.1.2. Non-Functional Requirements

1. **Portal Availability**: the portal must be available to all users;
2. **Layout**: all pages on the portal must have the same layout pattern, which must include a header;
3. **Header Layout**: the header must enable the search of registered experiments, as well as a navigation mechanism that helps the user navigate through all the pages;
4. **Clarity and Simplicity**: one simple interface with intuitively organized elements and the use of one simple language;
5. **Consistency**: same standard layout, colors, and typography to all pages. Pages with the same element organization.

3.3. Technologies Used

The following technologies were used to build the portal:

- **TypeScript**[1]: TypeScript is considered a superset of the JavaScript programming language with static typing. Based on this programming language, errors related

[1]**TypeScript** official website: https://www.typescriptlang.org

to data typing can be discovered and corrected at development time. TypeScript also supports most of the libraries and frameworks used for web development.

- **React**[2]: React is a JavaScript framework that contains a set of elements that help develop code using JavaScript in web systems. React works with the front-end of the application and uses components to create the interface, allowing flexibility and scalability.
- **Vite**[3]: Vite is a front-end development tool that speeds up development by using native ES modules, so only changed files are reloaded. For example, in a React project, updating a single component triggers instant hot module replacement instead of rebuilding the entire app. Vite also supports TypeScript.
- **MongoDB**[4]: MongoDB is a NoSQL database management system that uses a JSON(JavaScript Object Notation)-oriented document structure (BSON (Binary-encoded serialization of JSON data) in MongoDB) to store data. MongoDB is used in applications that work with unstructured data and require flexibility and scalability.
- **Axios**[5]: Axios is a library that uses http protocol, providing flexibility, scalability, and error treatments using promises.
- **TailwindCSS**[6]: TailwindCSS is a framework that aims to reduce the number of lines of CSS code, used for the web portal interface.

3.3.1. Interface Development

An initial version of the interface was developed to explore and validate the core functionalities of the portal. The design emphasizes three central features: experiment registration, experiment visualization, and the integrated search mechanism available in the portal's header. More information can be found at `https://doi.org/10.5281/zenodo.17230418`.

3.4. Portal Testing Methodology

Software testing is a critical activity in the software process, conducted to verify that a program performs its intended functions and to discover defects before the system is put into use [Sommerville 2015]. As noted by Pressman and Sommerville, testing consists of two distinct activities: verification and validation. Verification demonstrates that the software meets its requirements and specifications. Validation ensures that the system meets the user's needs [Pressman 2019], [Sommerville 2015].

To ensure that the requirements defined for the portal were correctly implemented, a testing strategy was devised, focusing on the portal's main functionalities. In this context, the following tests were performed, following the testing strategies considered by Pressman [Pressman 2019]: Database Testing and Content Testing.

On the other hand, a validation approach was also adopted to assess whether the system fulfills the users' real needs [Sommerville 2015, Pressman 2019]. In this context,

[2]**React** official website: https://react.dev
[3]**Vite** official website: https://vitejs.dev
[4]**MongoDB** official website: https://www.mongodb.com
[5]**Axios** official website: https://axios-http.com
[6]**TailwindCSS** official website: https://tailwindcss.com

End-To-End Testing was performed, considering what is proposed by Valente, simulating a user workflow in the portal [Valente 2020].

3.4.1. Database Testing

For the portal, the database is a central and critical element, containing the data from the registered experiments. It is therefore essential to test the database to ensure that data is stored and retrieved correctly and that performance is acceptable [Pressman 2019]. This process is particularly critical for the portal's experiment search functionalities, which depend on the database's reliability and efficiency. Our database testing focused on verifying the application's interaction with the database, ensuring both correct connectivity and the integrity of the data. For the database testing, the Jest JavaScript Testing Framework was used[7]. In the context of database testing, connectivity and validation were chosen:

- **Connectivity:** This test verifies that the application establishes a proper connection with the database. A failure to connect makes all other database-related functions unusable.
- **Validation:** This test seeks to ensure that the data stored in the database is valid and adheres to the defined schema and integrity constraints. This includes checking data types, relationships, and business rules implemented at the database level.

3.4.2. Content Testing

The content testing, particularly relevant for web applications, focuses on uncovering semantic and syntactic errors in the content presented to the user [Pressman 2019]. The objective is to find errors in the organization or structure of content and to verify the accuracy and completeness of the information presented as navigation occurs [Pressman 2019].

This test was planned in two steps. First, the functionalities responsible for retrieving data were tested to ensure that the interface components could correctly render the content stored in the database. These functionalities, tested using the Jest framework, are related to the following functions:

- **getAllExperiments:** Makes one request to the backend to fetch all experiments from the portal.
- **getSimpleSearchData:** Makes one request to the backend to search for experiments that match a given search string.
- **getExperimentById:** Makes one request to the backend to fetch data for a specific experiment using its ID.

Following the verification of these data-retrieval functions, the interface components themselves were checked manually to ensure that they rendered the fetched content correctly.

[7]Jest official website: https://jestjs.io/

3.4.3. End-to-End Testing

While unit and integration tests focus on specific parts of a system, system testing, or end-to-end testing, tests the system as a whole [Sommerville 2015]. The goal is to verify that all system components are compatible, interact correctly, and that the complete system meets its functional and non-functional requirements [Valente 2020]. This form of testing is essentially a black-box activity where the system's external behavior is evaluated against the requirements [Pressman 2019].

For the end-to-end test, Playwright [8], a framework for end-to-end tests in web applications. This test simulates a user's interaction with the portal by executing a complete workflow. The test navigates to the experiment registration page, enters valid data into the form, and submits it. Upon receiving a confirmation from the backend, the test stores the new experiment's ID from the JavaScript Object Notation (JSON) response.

Subsequently, the test proceeds to the portal's experiment repository page and searches for the title and authorship of the newly registered experiment. Upon finding it, the test clicks the "See Details" button to navigate to the experiment's full data page. On this final page, it verifies two key points:

1. That the title and authorship match the data originally submitted.
2. That the experiment ID in the page's URL corresponds to the ID received from the backend after the initial registration.

This test was conducted using three rendering engines: Chromium[9], WebKit[10], and Gecko[11]. That means the test was conducted using the following navigators: Google Chrome[12], Safari[13], and Firefox[14].

4. Results

The tests performed on the portal confirmed the correct implementation of the functionalities defined in the requirements. The results are summarized below, while the detailed logs and full reports of all tests are available in the public repository: https://doi.org/10.5281/zenodo.13788104

4.1. Database Testing

The database tests covered connectivity and validation aspects.

- **Connectivity:** In the connectivity test, the system successfully established a connection to the database when valid credentials were provided. Conversely, when invalid credentials were used, the system correctly rejected the connection attempt, returning the expected authentication error.

[8]Playwright official website: https://playwright.dev/
[9]https://www.chromium.org
[10]https://webkit.org
[11]https://developer.mozilla.org/en-US/docs/Glossary/Gecko
[12]https://www.google.com/chrome
[13]https://www.apple.com/safari
[14]https://www.mozilla.org/firefox

- **Validation:** In the validation test, the database accepted experimental data that adhered to the defined schema and constraints, while invalid data was rejected, generating the appropriate validation errors. These results confirm that the database layer enforces both access security and data integrity.

4.2. Content Testing

The content tests first verified the correctness of the functionalities responsible for retrieving data from the backend. The functions *getAllExperiments*, *getSimpleSearchData*, and *getExperimentById* all returned the expected data when valid requests were made and handled errors appropriately when failures were simulated.

Subsequently, the interface components that render this data were tested. The repository page of the portal correctly displayed the list of experiments, including essential metadata, and appropriately presented error messages when backend failures were forced. These results confirm that both data retrieval and presentation in the user interface operate as intended.

4.3. End-to-End Testing

The end-to-end test simulated the complete workflow of a user registering a new experiment. Across all tested browsers (Google Chrome, Firefox, and Safari), the experiment was successfully registered, stored in the database, and retrieved in the repository view. The details displayed matched the data initially submitted, and the identifiers used in the interface corresponded to those returned by the server. These results confirm that the portal supports the user workflow across multiple execution environments.

5. Lessons Learned and Future Directions

In this study, several key lessons emerged. The experimental process shaped the portal's structure and functionalities, ensuring alignment with best practices in the ESE field [Wohlin 2012]. Utilizing an experimentation ontology significantly improved the organization and clarity of experimental data, underscoring the importance of structured approaches in software engineering research [Vignando et al. 2020]. The requirements ensured that the portal's main functionalities were built, with a simple and intuitive interface.

From a technological perspective, the use of technologies such as TypeScript, React, and MongoDB improved development efficiency and scalability. The database testing, content testing, and end-to-end testing were essential for verifying the portal's reliability. Nevertheless, the tests were limited in scope, as they primarily assessed functional correctness. Aspects such as performance under high loads, usability across diverse user profiles, and security vulnerabilities remain open challenges.

Looking ahead, the project's future directions include improving the portal's usability, which will be assessed through a study validation with students, professors, and researchers experienced in ESE. Other plans include increasing the database of registered experiments and implementing advanced features such as more robust search mechanisms.

6. Final Remarks

This paper describes the design, implementation, and evaluation of an initial version of a web portal for teaching and practicing controlled experiments in SE, following an experimentation ontology as a structural basis [Vignando 2020]. This alignment with methodological foundations ensures that the portal is not merely a technological artifact, but an environment that supports the systematic practice of experimentation in SE.

The partial results obtained through the testing phase confirm that the portal meets the proposed requirements. These evaluations validate the feasibility of the portal as a reliable and scalable tool for researchers. Moreover, the portal demonstrated robustness in its core functionalities, such as experiment registration and retrieval, while maintaining simplicity of use.

Future studies will focus on enhancing usability through empirical evaluations with real users, ensuring that the system is accessible and intuitive. Also, the repository of experiments will be expanded, and an advanced search will be implemented. As a practical application, the portal can be used in undergraduate or graduate courses to facilitate in-class discussions or to help students plan the experimental activities of their research. Considering these new functionalities, the portal has the potential to evolve into an ecosystem for designing, executing, and disseminating controlled experiments in SE.

References

Adeyinka, T. and Bashorun, M. (2011). Impact of web portals on e-learning. *4th International Conference on the Applications of Digital Information and Web Technologies, ICADIWT 2011.*

Anchundia, C. E. (2020). Resources for reproducibility of experiments in empirical software engineering: Topics derived from a secondary study. *IEEE Access*, 8:8992–9004.

Brito, K. S. (2014). Brazilian government open data: Implementation, challenges, and potential opportunities.

Correa, A. S. (2018). Investigating open data portals automatically: a methodology and some illustrations.

Freire, M. A., Kulesza, U., Aranha, E., Jedlitschka, A., Neto, E. C., Acuña, S. T., and Gómez, M. (2014). An empirical study to evaluate a domain specific language for formalizing software engineering experiments. In *SEKE*, pages 250–255.

Furtado, V. R. (2018). Diretrizes para avaliação de qualidade de quasi-experimentos e experimentos controlados em linha de produto de software. Master's thesis, Universidade Estadual de Maringá.

Gonzalez-Barahona, J. M. and Robles, G. (2023). Revisiting the reproducibility of empirical software engineering studies based on data retrieved from development repositories. *Information and Software Technology*, 164:107318.

Haldar, S. and Capretz, L. F. (2024). Interpretable software maintenance and support effort prediction using machine learning. In *2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, pages 288–289, Lisbon, Portugal.

Jedlitschka, A., Ciolkowski, M., and Pfahl, D. (2008). Reporting experiments in software engineering. In *Guide to advanced empirical software engineering*, pages 201–228.

Juristo, N. and Moreno, A. M. (2013). *Basics of software engineering experimentation*. Springer Science and Business Media.

Juristo, N. and Vegas, S. (2016). Analyzing software engineering experiments: Everything you always wanted to know but were afraid to ask. In *2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C)*, pages 900–901, Austin, TX, USA.

Kipping, S., Stuckey, M. I., Hernandez, A., Nguyen, T., and Riahi, S. (2016). A web-based patient portal for mental health care: Benefits evaluation. *J Med Internet Res*, 18(11):e294.

Kitchenham, B., Al-Khilidar, H., Babar, M. A., Berry, M., Cox, K., Keung, J., ..., and Zhu, L. (2008). Evaluating guidelines for reporting empirical software engineering studies. *Empirical Software Engineering*, 13:97–121.

Kitchenham, B. A. (2016). *Evidence-Based Software Engineering and Systematic Reviews*. CRC Press.

Luz, C. D., OliveiraJr, E., and Steinmacher, I. (2020). Uma ontologia de apoio ao ensino de experimentaçao em engenharia de software. In *Anais Estendidos do XI Congresso Brasileiro de Software: Teoria e Prática*, pages 70–76. SBC.

Mead, N. R. and McGraw, G. (2005). A portal for software security. *IEEE Security and Privacy*, 3(4):75–79.

Pressman, R. S. (2019). *Software Engineering: A Practitioner's Approach*. MC GRAW HILL INDIA, 8th edition.

Robinson, P. T. (2019). Communication network in an agile distributed software development team. In *2019 ACM/IEEE 14th International Conference on Global Software Engineering (ICGSE)*, pages 100–104, Montreal, QC, Canada.

Scatalon, L. P., Garcia, R. E., and Correia, R. C. M. (2011). Packaging controlled experiments using an evolutionary approach based on ontology (s). In *SEKE*, pages 408–413.

Sommerville, I. (2015). *Software Engineering*. Pearson.

Valente, M. T. (2020). *Engenharia de Software Moderna: Princípios e Práticas para Desenvolvimento de Software com Produtividade*. Editora: Independente.

Vignando, H. (2020). Ontoexper-spl: uma ontologia de apoio a experimentos de linha de produto de software. Master's thesis, Universidade Estadual de Maringá.

Vignando, H., Furtado, V. R., Teixeira, L. O., and OliveiraJr, E. (2020). Ontoexper-spl: An ontology for software product line experiments. In *ICEIS (2)*, pages 401–408.

Wohlin, C. (2012). *Experimentation in Software Engineering*. Springer Science Business Media.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., and Wesslén, A. (2000). *Experimentation in software engineering: an introduction*. Kluwer Academic Publishers, Massachusetts.