Empirical analysis of gherkin quality and common mistakes by novice testers: Implications for sustainable software testing

Luthfi Satrio Wicaksono[1], *Joe Lian* Min[1*], *Asri* Maspupah[1], *Rio* Agasta[1], *Jelang Anugrah* Raharjo[1], *Reqi Jumantara* Hapid[1], *Muhammad Rafif* Genadratama[1], and *Yahya Alfon* Sinaga[1]

[1]Computer Engineering and Informatics, Politeknik Negeri Bandung, 40559 Bandung Barat, Indonesia

Abstract. Behavior-Driven Development (BDD) relies on quality Gherkin scenarios to bridge communication between technical and non-technical stakeholders. However, novice software testers often struggle to transition from procedural writing to declarative scenarios, resulting in low-quality scenarios and wasted automated testing resources. This research proposes a systematic rubric-based methodology with 12 assessment aspects to measure the quality of Gherkin scenarios created by novices software tester. The focus of the study is an empirical comparative analysis of Gherkin file artifacts from two representative systems, JTK-Learn (novices) and WebsiteOne (professionals), to identify quality gaps and common mistakes. The research stages include data collection, rubric development, Gherkin quality measurement, and comparative descriptive quantitative analysis. The results show significant gaps in conceptual and collaborative aspects, while novices are strong in structural aspects, such as scenario focus and step structure. Five common mistakes made by novices were identified, namely the use of technical vocabulary, procedural writing, inconsistency in domain terminology, mixing of abstraction levels, and step duplication that reduces maintainability. These mistakes cause a waste of computational resources and energy. However, with the validated rubric as a reliable evaluation tool, there is a clear potential for more effective BDD training development and sustainable software testing practices, which can significantly improve the efficiency of automated test execution.

1 Introduction

The increasing complexity of modern software systems demands intensive collaboration among various roles. In this context, traditional test cases often fail to ensure alignment between developers, testers, and non-technical stakeholders [1]. Behavior-Driven Development (BDD) has emerged as an essential practice that promotes effective communication through structured natural language. BDD is now a widely adopted industry practice and has been shown to improve software quality, customer satisfaction, and overall

* Corresponding author: joelianmin@polban.ac.id

project success [2]. In its implementation, the Gherkin scenario, which functions as the primary artifact of BDD, plays a role in documenting system requirements in a narrative format that is easy to understand, thereby bridging technical and business needs. The quality of a good Gherkin scenario is essential for readability, test automation, and its role as living documentation [3]. Therefore, the success of BDD fundamentally depends on the quality of the written Gherkin scenarios.

Effective Gherkin scenario writing remains a significant challenge, especially for novice software testers. They often have difficulty switching their mindset from procedural testing steps to behavior-focused declarative principles in BDD [4]. Common mistakes include the use of technical terms, inconsistent levels of abstraction, and writing non-reusable steps [2]. This problem is exacerbated by the lack of standardized evaluation methods, making it challenging to measure competency and the effectiveness of BDD training [5].

Thus, there is a research gap in the form of a lack of validated quantitative instruments to objectively assess the quality of Gherkin scenarios. This is important because poor-quality scenarios not only hinder cross-role communication but also lead to wasted computational resources and energy consumption during repeated automated testing executions. From a green technology perspective, this inefficiency contradicts sustainable software practices [6]. By improving the quality of Gherkin scenarios through validated quantitative evaluation instruments, testing can become more efficient and environmentally friendly, while also improving cross-role communication.

To address the gap between the quality of Gherkin scenarios written by novice and professional software testers, this study proposes an empirical comparative approach to quantitatively and objectively evaluate the quality of Gherkin scenarios. The primary focus of the study aims to answer the following research question (RQ):

1. RQ-1: How can the ability of novice software testers to transform test cases into Gherkin scenarios be quantitatively measured using a standardized rubric?
2. RQ-2: What are the main common mistakes made by novice software testers?
3. RQ-3: How can a comparison of Gherkin file artifacts from professionals validate the existence of common mistakes made by novice software testers?

The comparison of Gherkin scenario quality against professional software testers in this study is intended to establish a benchmark that represents industry practice. The artifacts produced by professional software testers are considered a representation of the quality of Gherkin scenarios in real project environments, thus serving as a gold standard [7]. Therefore, the comparative analysis not only reveals differences in skill levels but also serves as a means of validation for the rubric used, particularly in distinguishing the quality of Gherkin scenario writing between novice and professionals software tester.

The main contribution of this research is a validated rubric-based methodology to evaluate Gherkin scenarios. This methodology has two objectives: (1) to provide a tool to objectively identify and quantify common mistakes made by novice software testers, and (2) to offer a foundation for the development of a targeted BDD training curriculum. To realize these two objectives, this research approach focuses on an empirical comparative analysis by comparing Gherkin scenarios written by novices and professionals, so that the research results can provide quantitative evidence of the quality gap as well as validate the rubric as a reliable evaluative instrument.

2 Literature review

BDD is a software development method that involves both technical and non-technical stakeholders, particularly in facilitating test automation. In BDD, a set of input data is written in natural language phrases, Gherkin scenarios. These scenarios play a crucial role in bridging the gap between business requirements and the technical implementation of application behavior [8]. Gherkin scenarios use the narrative structure Given–When–Then that describes the context (Given), user or system actions (When), and expected outcomes (Then). An example of writing a Gherkin scenario is shown in Fig. 1.

```
Given my account is in credit by $100
When I request withdrawal of $20
Then $20 is dispensed
And my balance is $80
```

Fig. 1. Sample *scenario from* an ATM *feature* [7].

The criteria for writing good Gherkin scenarios include that scenarios are declarative, easy to understand by non-technical stakeholders, and can be utilized directly in automated testing while serving as living documentation, which means they are continuously updated to reflect the current state of the system [9]. Relevant issues in writing Gherkin scenarios in empirical studies are efforts to maintain scenario quality and measurement methods [2,7].

Various practice studies show a consistent pattern of errors in Gherkin scenarios created by less experienced writers. These errors include writing technical steps that non-technical stakeholders do not easily understand, inconsistencies in the level of abstraction, which can lead to confusion, the use of terminology that non-technical stakeholders do not understand, and low reusability of steps. Poor-quality Gherkin scenarios have the potential to weaken communication across stakeholders, cause automated test suites to become fragile, and increase maintenance costs [2,7].

On the other hand, sustainable development in software engineering does not only focus on the efficient use of hardware resources, but also includes energy-friendly software development practices [10]. One important issue raised is efficiency in the software testing process, particularly in automated testing, which often consumes a lot of computing power. In the context of BDD, the quality of Gherkin scenarios is a crucial aspect because scenario quality directly affects the duration and number of test executions. Empirical studies show that high-quality scenarios, i.e., those that are clear, concise, and consistent, can reduce redundancy and duplication of steps, thereby speeding up the automated testing process and reducing energy consumption [11].

Although best practices and some metrics have been proposed, there is still a lack of standardized and validated assessment tools to measure the quality of Gherkin scenarios quantitatively. An initial set of evaluation criteria has been suggested, but only a few comprehensive rubrics have been tested for inter-rater reliability and validated through empirical evidence. The absence of such instruments makes it difficult to measure progress in the competency of writing quality Gherkin scenarios [2,9].

Based on these issues, this research aims to contribute to two areas: (1) presenting a structured rubric for quantitatively assessing Gherkin scenario quality, and (2) validating the rubric through comparative analysis between scenarios written by novices and professionals. Thus, this study provides an empirical basis for more effective BDD training for designers. Focusing on these areas, this research concentrates on evidence and instruments directly related to practical problems in Gherkin writing.

3 Method

This research applies an empirical comparative study to analyse the differences in Gherkin scenario quality between novice and professional writers, ensuring a robust research process. Data were collected from two representative systems. The data were then evaluated using a defined rubric in order to derive evidence-based findings and map common error patterns frequently encountered in novice Gherkin screenwriters.

3.1 Evaluation framework (assessment rubric)

The quality of Gherkin scenarios is assessed using a rubric adapted from Binamungu's research in 2020, titled "Characterising the Quality of Behaviour Driven Development Specifications". In his research, Binamungu revealed that the Gherkin scenario assessment rubric contains recommendations for quality aspects, stakeholder best practices, and perspectives from developers or Software Quality Assurance [7]. The rubric developed by Binamungu has proven to enhance the reliability of qualitative assessments involving subjective interpretation [12].

Based on the adopted rubric, the Gherkin scenario assessment encompasses 12 quality aspects detailed in Table 1. Several aspects of the rubric, such as Scenario Maintainability, Reusability of Steps, Abstraction Level Consistency, and Step Independence, have direct implications for reducing redundancy [11]. By optimizing these aspects, the quality of scenarios can be improved while supporting the sustainability of automated testing.

Each quality aspect is scored on an ordinal scale with a range of 1- 4 points. The ordinal scale is used because it is suitable for measuring performance quality that is hierarchical without an absolute distance between levels [13]. The representation of each quality score is explained as follows:

1) 1 indicates Poor, the quality aspect is not applied;
2) value 2 indicates Fair, the application of quality is minimal;
3) 3 indicates Good, the implementation of quality is pretty good but can be improved; and
4) a score of 4 indicates Excellent quality, the application of each quality criterion of the Gherkin scenario is consistent and optimal.

Table 1. Gherkin quality evaluation rubric.

No.	Quality Aspect	Description
1	Readability	How easy is the scenario to understand as business documentation for all stakeholders?
2	Clarity and Unambiguity	Does each step and scenario convey an unambiguous meaning?
3	Domain Vocabulary Consistency	To what extent is the step phrase reused across scenarios and features?
4	Elimination of Technical Vocabulary	Have Gherkin scenarios avoided technical or implementation-specific terms in the Gherkin step?
5	Reusability of Steps	How far is the phrase "step" reused between scenarios and features?
6	Abstraction Level Consistency	Consistency in step abstraction level across test suites.
7	Alignment with Stakeholder Understanding	To what extent does the scenario reflect a shared understanding with stakeholders?
8	Scenario Focus	Each scenario tests a single behavior or goal.
9	Step Structure (Given-When-Then Discipline)	Sequence and the semantic function according to the Given, When, and Then format.
10	Scenario Maintainability	Ease of updating and expanding the scenario without causing duplication.
11	Shared Glossary and Concept Inventory	The existence of a glossary for consistency.
12	Step Independence and Statelessness	To what extent do the scenario steps avoid implicit dependence?

3.2 Data participation

This research uses an artifact-based empirical comparative study approach by analysing Gherkin artifacts in two different groups. The selection of Gherkin artifacts in both groups is based on the similarity of the functionality domain.

3.2.1 Novice Gherkin scenario

In this group, the Gherkin file artifacts consist of nine feature files (.feature) written by novice software testers in the web-based educational application development project, JTK-Learn [14]. The educational application was chosen as the application domain for beginners because it is relatively familiar to them, as they are generally accustomed to using e-learning during the learning process [15]. This choice significantly reduces the cognitive load associated with understanding business processes, allowing the focus to be directed towards writing Gherkin scenarios rather than learning a new domain. The selection of nine JTK-Learn feature files refers to the main user interactions (students). As the main users of educational applications, students most often access courses, multimedia content, interactive assignments, motivational elements, and progress tracking as core functionality areas [15].

Before working on the project, these novice software testers had participated in a practical one-week intensive BDD training using materials from the online course, School of Cucumber [16]. The training covered the basic concepts and philosophy of BDD, discovery and example mapping techniques, declarative scenario writing, and basic Cucumber and Selenium integration, all of which are directly applicable to their work.

3.2.2 Professional Gherkin scenario

In this group, the Gherkin artifact files include nine feature files (.feature) from the open-source web-based project, WebsiteOne [17]. WebsiteOne is an open-source web application that has been developed by the AgileVentures community since 2014–2024 with extensive contributions (97 contributors). The project has measurable professional indicators through GitHub metrics, namely: 144 stars (popularity), 246 forks (technical interest), 97 contributors (broad collaboration), 5282 commits (high activity), and 13 open issues out of a total of 1546 handled, indicating an active cycle of improvements. The number of stars indicates popularity, which can lead to an increase in users and contributors [18]. These indicators confirm the quality and credibility of WebsiteOne as a representative system in the use of Gherkin specifications in real-world BDD practices. WebsiteOne's Gherkin artifacts also reflect more complex, collaborative, and measurable scenario writing practices, making them relevant as a gold standard for validating novice errors. Not all Gherkin files in the WebsiteOne project were used as assessment references. Nine WebsiteOne feature files were selected based on their similarity of theme with the JTK-Learn application domain, namely: login, content access, dashboard views, project list, information pages, and activity progress, to ensure functional domains were comparable in the analysis.

3.3 Assessment process

All Gherkin scenarios of data participation are evaluated manually using the proposed rubric. The assessments are carried out independently by several evaluators referencing explicit definitions for each aspect, to ensure consistency and reduce potential interpretive bias. Before the evaluation process, all evaluators have meticulously calibrated their understanding of the rubric to harmonize perceptions of the evaluation criteria. This rigorous calibration process instills confidence in the fairness of the evaluation process. It aims to improve the reliability and integrity of the evaluation results, as well as to achieve adequate inter-rater agreement.

3.4 Data analysis

The assessment of the data was conducted using a descriptive-quantitative comparative approach [19]. The data analysis process involved calculating the average scores of 12 quality aspects for each novice and professional group. The difference in average scores between novice and professional Gherkin file artifacts can indicate the gap in writing quality. This quality gap plays a pivotal role in identifying the most significant aspects that differentiate the quality of Gherkin produced by the two groups. The aspect with the most significant gap is identified as an indicator of common novice mistakes. In addition, the quality aspect rubric was designed to highlight repetitive and inefficient writing patterns, thus serving as a tool to identify potential waste in Gherkin scenario writing practices.

4 Result and discussion

4.1 Assessment results

The quality measurement of Gherkin file artifacts used in automated testing involves two different software under test (SUT), namely JTK-Learn and WebsiteOne. JTK-Learn is a web-based Learning Management System (LMS) application [14]. Meanwhile, WebsiteOne is a web application based on the Ruby on Rails framework as an online collaboration portal for project management and community communication [17]. Nine Gherkin artifact files in both groups are measured for quality based on the rubric, Table 1. All Gherkin files are assigned a valuation weight ranging from 1 to 4, resulting in an average score on each quality aspect. The results of the Gherkin scenario quality measurement are shown in Table 2.

Table 2. Results of the empirical comparative study: Analysis of Gherkin scenario quality scores.

Evaluation of Quality Aspects	JTK Learn (novice)	WebsiteOne (Professional)	Gap (Web01 − JTK)
Overall Average (1–4)	2,66	3.34	0,68
Elimination of Technical Vocabulary	1,58	2,72	1,44
Alignment with Stakeholder Understanding	1,50	2,61	1.11
Shared Glossary and Concept Inventory	1,33	2,44	1.11
Abstraction Level Consistency	2,11	2,89	0,78
Scenario Maintainability	2,22	3,00	0,78
Medium Improvement Areas (0.30–0.69 gap)			
• Readability	2,75	3,20	0,45
• Clarity and Unambiguity	2,80	3,30	0,50
• Reusability of Steps	2,60	3,00	0,40
Novice Strengths (scores > 3)			
• Scenario Focus	3,10	3,40	0,30
• Step Structure (G W T)	3,05	3,50	0,45
• Step Independence and Statelessness	3,20	3,60	0,40

Table 2 presents the average scores for each quality aspect and the overall average in both groups of Gherkin file artifacts, including the quality gap between novices and professionals. The gap analysis highlights the areas within Gherkin scenario quality that novices find most difficult compared to professionals. Based on Table 2, RQ1 can be answered by comparing the quality of Novice Gherkin scenarios (2.66) with the professional project standard (3.34), revealing a gap of 0.68. This shows that novices' mastery has only reached the basic structure, but not the practical level.

On the other hand, novices show strengths in syntax aspects oriented toward executing test scenarios, namely Scenario Focus (3.10), Step Structure (3.05), and Step Independence and Statelessness (3.20). This suggests consistency in form and the ability to ensure each test step can be executed independently. In contrast, aspects of collaboration and domain understanding weakened significantly, namely aspects of Elimination of Technical Vocabulary (1.28), Shared Glossary and Concept Inventory (1.33), and Alignment with Stakeholder Understanding (1.50). This indicates that scenarios are still too technical, the terminology is inconsistent, and does not reflect the user or business perspective. Some other aspects show medium gaps (0.30–0.69).

This comparison highlights that the main gap does not lie in syntax, but in aspects that determine comprehensibility for non-technical parties. For technical groups, Gherkin scenarios serve as executable instructions, while for non-technical parties, scenarios represent business requirements [7,20]. Thus, the quality of Gherkin scenarios must be understood in a collaborative framework, namely, the extent to which scenarios can be used simultaneously by both groups. These findings underscore the urgency and importance of a transition from procedural to descriptive writing. This shift is necessary for scenarios to truly function as a ubiquitous language that connects technical and non-technical perspectives.

4.2 Identification of common mistakes

The process of identifying common errors is conducted through quantitative analysis, highlighting the quality score gap between novices and professionals in Table 2, which shows five quality aspects with a gap ≥ 0.78. This analysis is continued with the process of harmonizing the understanding of error characteristics among evaluators. This process involves a detailed comparison of concrete examples of Gherkin file artifacts, where evaluators discuss and agree on the presence and severity of each error. Based on this analysis, five common mistakes in Gherkin scenario writing have been defined (RQ2), namely the use of technical vocabulary, procedural writing, inconsistent domain terminology, mixing levels of abstraction, and duplicate steps that reduce maintainability. Validation through comparison with professional artifacts (RQ3) confirms that these five common mistakes only consistently appear in novice scenarios, thus serving as a basis for diagnosing early competency weaknesses.

In addition, analysis of the evaluation results highlights the potential for waste in Gherkin scenarios written by beginners. Weaknesses in the areas of Scenario Maintainability and Abstraction Level Consistency indicate a tendency toward duplication of steps and mixing of abstraction levels, resulting in redundancy. This condition reduces the maintainability of artifacts and increases the execution load of the test suite, because repetitive steps increase processing time and consumption of computational resources. This waste has implications for increased energy consumption used in automated testing, which is contrary to the principles of green technology and sustainable development.

4.3 Positive implications of early training on basic Gherkin competency

The evaluation results demonstrate the success of the short training in developing the structural ability to write Gherkin scenario syntax. This indicates a mastery of syntax patterns and the ability to compose consistent and separate scenario steps. However, conceptual and communication weaknesses indicated a limited understanding of domain meaning and stakeholder perspectives. As a result, although the scenario form was syntactically correct, interpretation and readability by non-technical stakeholders were often hindered.

The fundamental difference between procedural and descriptive approaches further clarifies the essence of this transition, which is important for novices. Procedural writing focuses on how to test, while Gherkin descriptive scenarios emphasize what behavior matters. The descriptive orientation allows scenarios to be understood, validated, and utilized as living documentation [7]. Therefore, the training curriculum could expand the focus from syntactic techniques to conceptual aspects. Suggested training materials may include: (1) exercises to simplify technical language; (2) construction of a shared domain glossary; (3) exercises writing from stakeholder perspectives, and (4) practices to maintain consistency in level of abstraction and scenario clarity. This integrated approach will improve the connection between technical skills and cross-role communication skills, thereby improving the readiness of novices to meet industrial practice needs.

5 Conclusion

The results of the study show that the assessment rubric for 12 aspects of quality is effective as a quantitative instrument for evaluating the quality of Gherkin scenarios. Empirical comparative analysis reveals a significant gap between novice and professional Gherkin scenario artifacts (2.66 vs. 3.34; gap = 0.78). Novices demonstrate strengths in structural aspects but show weaknesses in conceptual and collaborative aspects. The identification of this gap and the potential for improvement is a significant finding. Common mistakes made by novices can reduce maintainability and cause waste of computational resources and energy in automated testing, including ineffective cross-role communication. These findings confirm that improving the quality of Gherkin scenarios contributes to automated testing efficiency and sustainable software testing practices, in line with green technology principles. BDD training needs to urgently emphasize conceptual aspects and cross-role communication to bridge this quality gap. Validated rubrics can be a reliable quantitative evaluation tool and have the potential to be integrated into automated tools for continuous assessment. It is recommended that future research expand the domain and level of participant experience and develop the integration of rubrics into automated tools to support more efficient and sustainable software development.

References

1. E. Bjarnason, H. Sharp, B. Regnell, Improving Requirements-Test Alignment by Prescribing Practices that Mitigate Communication Gaps. *Empirical Software Engineering*. **24**, 2364–2409 (2019). https://doi.org/10.1007/s10664-019-09698-6
2. L. P. Binamungu, S. Maro, Behaviour driven development: a systematic mapping study. *Journal of System and Software.* **203**, (Sep. 2023). https://doi.org/10.1016/j.jss.2023.111749
3. P. Rai, Extending automated testing to high-level software requirements. Technical Report Thesis, Mid Sweden University, Sundsvall (2016). https://miun.diva-portal.org/smash/record.jsf?pid=diva2%3A941164
4. S. Santos, T. Pimentel, F. G. Rocha, M. S. Soares, Using Behavior-Driven Development (BDD) for Non-Functional Requirements Software. *Software*. **3**, 271–83 (2024). https://doi.org/10.3390/software3030014
5. M. Irshad, R. Britto, K. Petersen, Adapting Behavior Driven Development (BDD) for Large-Scale Software Systems. *Journal of Systems and Software*. **177**, 1-20 (2021). https://doi.org/10.1016/j.jss.2021.110944
6. P. Gómez, B. Bermejo, C. Juiz, P. Freire, T. Gualotuña, D. Cortés, A Systematic Review of the State-of-the-Art of Green IT and Green Software from 2020. *Journal of*

Information Systems Engineering and Management. **10**, 811–33 (2025). https://doi.org/10.52783/jisem.v10i42s.8195

7. L. P. Binamungu, S. M. Embury, N. Konstantinou, Characterising the quality of behaviour driven development specifications, in Proceedings of the 21st International Conference on Agile Software Development, Lecture Notes in Business Information Processing, Copenhagen, Denmark, June 8–12 (2020), pp. 87–102. https://doi.org/10.1007/978-3-030-49392-9_6
8. M. S. Farooq, U. Omer, A. Ramzan, M. A. Rasheed, Z. Atal, Behavior driven development: a systematic literature review. *IEEE Access*. **11**, 88008–88024 (Aug. 2023). https://doi.org/10.1109/ACCESS.2023.3302356
9. L. P. Binamungu, S. M. Embury, N. Konstantinou, Maintaining behaviour driven development specifications: challenges and opportunities, in Proceedings of the 25th IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER), Campobasso, Italy, March 20-23 (2018), pp. 175–184. https://doi.org/10.1109/SANER.2018.8330207
10. S. Kapoor, Green Software Quality: A Comprehensive Framework for Sustainable Metrics in Software Development. *International Journal of Computer Trends and Technology*. **72**, 113–20 (2024). https://doi.org/10.14445/22312803/IJCTT-V72I10P118
11. M. R. H. Misu, J. Li, A. Bhattiprolu, Y. Liu, E. S. de Almeida, I. Ahmed Test smell: A Parasitic Energy Consumer in Software Testing. *Information and Software Technology*. **181**, 1–13 (2025). https://doi.org/10.1016/j.infsof.2025.107671
12. E. E. Hansson, P. J. Svensson, E. L. Strandberg, M. Troein, A. Beckman, Inter-rater reliability and agreement of rubrics for assessment of scientific writing. *Education.* **4**, 12–17 (2014). http://article.sapub.org/10.5923.j.edu.20140401.03.html
13. N. R. M. Rokeman, Likert measurement scale in education and social sciences: explored and explained. *EDUCATUM Journal of Social Science.* **10**, 77–88 (Apr. 2024). https://doi.org/10.37134/ejoss.vol10.1.7.2024
14. R. Rafifgen, R. J. Hapid, L. Satrio, Y. S. Sinaga, R. Agasta, JTKLearn-Testing. *GitHub* (accessed Jul. 26, 2025). https://github.com/rafifgen/JTKLearn-Testing/
15. M. Wienand, T. Wulfert, H. Hoang, Design principles for e-learning platforms featuring higher-education students' enterprise systems end-user training. *Discover Educcation.* **3**, 1–31 (Dec. 2024). https://doi.org/10.1007/s44217-024-00165-z
16. M. Wynne, A. Hellesøy, J. Biezemans, Learn BDD and Cucumber. *cucumber.io* (accessed Jul. 26, 2025). https://school.cucumber.io/courses/bdd-with-cucumber-java
17. AgileVentures, WebsiteOne. *GitHub* (accessed Jul. 26, 2025). https://github.com/AgileVentures/WebsiteOne
18. H. Borges, A. Hora, M. T. Valente, Understanding the factors that impact the popularity of GitHub repositories, in Proceedings of the. *IEEE International Conference on Software Maintenance and Evolution (ICSME)*, Raleigh, NC, USA October 02-07 2016 (published Jan. 2017), pp. 334–344. https://doi.org/10.1109/ICSME.2016.31
19. M. Leotta, F. Ricca, A. Marchetto, D. Olianas, An Empirical Study to Compare Three Web Test Automation Approaches: NLP-based, Programmable, and Capture & Replay *Journal of Software: Evolution and Process*. **36,** 1–24 (2023). https://doi.org/10.1002/smr.2606
20. D. Mateus, D. S. da Silveira, J. Araujo, Pattern-Based Automation of User Stories and Gherkin Scenarios from BPMN Models for Agile Requirement. *Applied Sciences*. **15**, 1–26 (2025). https://doi.org/10.3390/app15179434