Investigating the Alignment of Software Testing Education with Industry Practices

Carla Lizandra Silva[1], Victor Anthony Alves[2], Carla Bezerra [1], Lara Gabrielly Lima[2]

[1]Universidade Federal do Ceará (UFC),
[2] Universidade de São Paulo (USP)

carlalizandra92@gmail.com, victorpa@usp.br, carlailane@ufc.br, laralima@usp.br

Abstract. *This study aims to analyze how software testing is currently taught in undergraduate computing programs and to what extent it aligns with modern practices adopted by the industry. We conducted a three-phase investigation: (i) a review of course syllabi from Brazilian undergraduate programs to examine the coverage and depth of testing-related content; (ii) a survey with students to assess their experiences and perceptions regarding software testing education; and (iii) a survey with industry professionals to identify prevailing testing practices. The results reveal a significant discrepancy between academic instruction and industry practices in software testing.*

1. Introduction

Software testing involves a set of procedures aimed at ensuring that the code behaves according to its specifications while preventing unwanted behavior [Myers et al. 2011]. In this context, testing is the main verification and validation mechanism used throughout the software development process [Clarke et al. 2017]. In undergraduate courses in Information Technology (IT), the teaching of testing is generally covered in the subject of Software Engineering (SE), especially through the topics of Verification, Validation and Testing. Among the most common types explored in teaching are unit testing, integration testing, system testing and acceptance testing [Melo et al. 2020]. However, there is a lack of integration between the teaching of testing and other subjects in the curriculum, which can limit students' practical and interdisciplinary training [Valle et al. 2015, Tramontana et al. 2024, Venson e Alfayez 2024].

Various pedagogical approaches have been adopted for teaching software testing, including project-based learning, problem-based learning and peer review [McManus e Costello 2019, Cheiran et al. 2017, Smith et al. 2012]. These methodologies use resources such as slides, books, tutorials and quizzes to support the learning process [Melo et al. 2020, da Silva et al. 2025]. In addition, strategies based on gamification, such as the use of the CleanGame [dos Santos et al. 2019] and GATE [Chen 2011] tools, have been gaining ground as they promote motivation and engagement through mechanics such as scoring and solving puzzles [Fulcini et al. 2023]. The software industry has made use of modern SE practices [Valente 2024].

Therefore, the aim of this study was to investigate the use of modern software testing practices, which are more common in industry and those covered in academic teaching, to identify gaps and challenges students face in undergraduate IT courses during the learning process. To this end, an initial analysis of the syllabi for the software testing subjects offered in these courses was carried out to identify the content covered.

A survey was then administered to students to assess their level of knowledge and their perceptions of software testing teaching, with an emphasis on the modern approaches presented during their training. Subsequently, a second survey was conducted with industry professionals, with the aim of mapping the most widely adopted testing practices in the market. Finally, a comparative analysis was conducted of the data obtained from the two surveys to identify possible discrepancies between academic training and the skills required by the production sector.

2. Background

Modern practices refer to approaches and techniques recently adopted in the software quality assurance process. One of these practices is test automation, which offers the advantage of covering more areas of the system and carrying out tests more efficiently, reducing execution time. Automation allows test scenarios to be repeated whenever necessary, making it more advantageous in terms of time, cost and usability [Gamido e Gamido 2019]. Test automation can be divided into two main types: code-based tests, which use existing interfaces, libraries, classes and modules to test different inputs and validate the results, and tests based on the Graphical User Interface (GUI), which generate events such as clicks and keystrokes. These events are then used by the test framework to evaluate the program's performance [Sneha e Malle 2017]. In their study, [Muneer 2014] identified 22 types of automated tests described in the literature, which are detailed in Table 1.

Table 1. Automated test types

1	Code-Coverage Testing	**12**	Acceptance Testing
2	Module Testing	**13**	Model-based Testing
3	Functional Testing	**14**	Specification-based Testing
4	Security Testing	**15**	Concurrency Testing
5	GUI Testing	**16**	Fault-based Testing
6	Regression Testing	**17**	Interoperability Testing
7	Random Testing	**18**	Load Testing
8	Usability Testing	**19**	Performance Testing
9	Unit Testing	**20**	Stress Testing
10	Integration Testing	**21**	Memory Leakage Testing
11	System Testing	**22**	Environment Testing

3. Related Work

The study by [Paschoal e de Souza 2018] focuses on understanding how software testing is taught in Brazilian universities, relying exclusively on the perspective of faculty members from higher education institutions in Brazil. Meanwhile, [Martins et al. 2021] shifts the lens to the industry, targeting software testing professionals, specifically those with a bachelor's degree in computing-related fields. Their work investigates professional experiences and practices, but does not explore how these professionals were trained or how their academic background influenced their current skillset. On the other hand, [Barrett et al. 2023] analyzes university syllabi and curriculum design to assess how well current educational programs prepare students for software testing in practice. However, their study remains limited to documentary analysis and does not incorporate first-hand feedback from those directly involved in the teaching or learning process. Likewise, [Cammaerts et al. 2024] provides valuable insights into students' opinions on specific aspects of a software testing course, but their investigation is restricted to perceptions within a single course context and lacks broader generalizability.

This study diverges from the aforementioned works in several important ways. First, it targets a dual audience: (i) undergraduate students currently enrolled in IT-related programs, who offer insights into the academic experience from the learner's perspective; and (ii) professionals working in the software testing industry who have completed a degree in IT, thus representing the practical application of academic training. This dual approach enables a deeper, more nuanced understanding of the alignment—or misalignment—between educational practices and industry needs. Second, this research investigates the perceived effectiveness, relevance, and gaps in these practices from both groups' perspectives. This allows for the triangulation of data from curriculum analysis, student feedback, and professional experience, offering a more holistic and validated picture of the current state of software testing education and practice. Finally, this work contributes by proposing evidence-based recommendations to improve curricula, enhance student preparation, and foster stronger integration between academia and industry.

4. Study Settings

This study aimed to investigate the alignment between modern software testing practices adopted by the industry and those addressed in the academic context. To identify training gaps and the main challenges faced by undergraduate students in IT during the learning process. To achieve this goal, the research was conducted in four stages.

4.1. Content Analysis of Testing Subjects in Software Engineering Courses

According to the [Association for Computing Machinery 2020] report, the curriculum guidelines for SE emphasize the importance of subjects that ensure the quality and reliability of software systems, covering essential verification and validation practices. Thus, using the evaluation grade provided by the Ministry of Education (MEC) as a selection criterion, undergraduate courses in SE in Brazil with a grade of four or five were selected. Table 2 shows the institutions that met the criteria mentioned. Some universities, although they were rated four or five, were not included in this list for various reasons. In some cases, the search for the course's pedagogical project or the curriculum matrix needed to verify the presence of test-related subjects was unsuccessful. In other situations, exclusion was due to information found on the MEC website indicating ongoing legal proceedings that could compromise the credibility of the SE course and/or the university itself. In all, 12 universities met the criteria, with 6 scoring 5 and 6 scoring 4. After this identification, a new analysis was conducted, focusing on the Pedagogical Course Projects (PPCs), to systematically map the content and skills related to software testing provided for each subject in the curriculum.

4.2. Survey to Identify Testing Practices Taught in IT Courses

At this stage, the survey's specific objectives and the profile of the target audience, computer science students, were defined. The construction of the instrument was based on a previous study of the literature, using the works of [Paschoal e de Souza 2018], [Melo et al. 2020] and [Martins et al. 2021] as a reference, with the aim of identifying and incorporating the main dimensions related to the teaching of software testing. Based on these guidelines, a structured questionnaire was drawn up, aligned with the research objectives and aimed at collecting data on students' perceptions, experiences and level of preparation. To ensure the clarity and validity of the instrument, a pre-test (pilot) was

Table 2. Institutions selected to analyze course syllabuses

Institutions	Acronym	MEC grade
University of Brasilia	UNB	05
Pontifical Catholic University of Rio Grande do Sul	PUCRS	05
Lutheran University Center of Palmas	CEULP	05
Federal University of Goiás	UFG	05
University Center of Grande Dourados	UNIGRAN	05
Federal University of Ceará	UFC	05
Federal Technological University of Paraná	UTFPR	04
University of the Joinville Region	UNIVILLE	04
Academia University Center	UNIACADEMIA	04
Catholic University of Salvador	UCSAL	04
Federal University of Rio Grande do Norte	UFRN	04
Federal University of Amazonas	UFAM	04

conducted with a group of volunteers. This stage enabled assessment of the questions' comprehensibility and identification of areas for improvement. The pilot test was applied to three students of IT undergraduate courses. The questionnaire was disseminated via electronic channels, particularly LinkedIn, via targeted posts and shares. In addition, the survey was disseminated via email to students in IT undergraduate courses and to coordinators of SE courses at other institutions, with the aim of broadening its reach and obtaining a representative sample of respondents. The survey questions, identified by IDs ranging from Q01 to Q27, are publicly available under an open license[1]. In the end, we received 32 valid responses from participants in five Brazilian states. The distributio of students who participated in the study in their respective writing courses: Software Engineering (12), Systems Analysis and Development (6), Information Systems (6), Computer Science (2), Computer Engineering (2), Digital Design (1) and Education in Computer Science (1).

4.3. Survey to Identify Modern Testing Practices Used in Industry

In this stage, a second survey was developed, this time targeting professionals from the software development industry who work specifically in the field of software testing. The objective was to identify the most commonly adopted modern testing practices in the industry. As with the first survey, the key information to be collected was first defined, guided by the same reference studies previously used, in order to structure the questionnaire and ensure alignment with the research goals.

Before the final deployment, the instrument underwent a pilot test with a group of volunteers, allowing for the identification and implementation of necessary adjustments to improve the clarity, effectiveness, and accuracy of the final version. The survey was disseminated through electronic communication channels, including posts and shares on LinkedIn, as well as in social media groups focused on software testing and technology. Additionally, it was shared via email with students of IT undergraduate courses and with participants of the first survey who had indicated working in the testing field at the time, inviting them to contribute from a professional perspective. All responses were documented and organized systematically, allowing for the distinction between valid and invalid data. The survey questions, identified by IDs ranging from Q01 to Q17, are publicly available under an open license[2].

[1] `https://i.postimg.cc/FKBtpMv9/Questions-Survey-Students.png`
[2] `https://i.postimg.cc/MptJMcrf/Questions-Survey-Industry.png`

Finally, 25 valid responses were obtained from participants in five Brazilian states. The highest concentration of responses came from the state of Ceará (CE), which accounted for 68% (17) of the participants. Next, the state of Pernambuco (PE) contributed 12% (3), while Minas Gerais (MG) and São Paulo (SP) each accounted for 8% (2) of the responses. The Federal District (DF) completed the sample with 4% (1) of the respondents. With regard to length of professional experience in the testing area, the majority of participants (44%; 11 responses) indicated that they had been working for between 1 and 3 years. Another 28% (7) said they had more than 5 years' experience, while 24% (6) reported between 3 and 5 years' experience. Only one participant (4%) said they had less than 1 year's experience in the area.

5. Results and Discussion

This section presents an integrated analysis of the results obtained from the evaluation of the syllabuses of software testing subjects in undergraduate courses, as well as results from the evaluation of software testing syllabi in undergraduate courses, as well as from two collection instruments: a survey administered to students and graduates in the area of computing, the syllabuses confirmed that all the universities selected had at least one subject focused on software testing. By analyzing the syllabuses, it was possible to identify whether the courses did in fact have at least one subject focused on testing and, through the pedagogical project of each course, we identified the content offered by each subject. Table 3 shows a list of the subjects focused on software testing that are taught in each of the courses at the institutions selected for the research.

Table 3. Courses offered in each institution's program

Institutions	Courses	Workload	Website
CEULP	Software Testing	60h	`https://bit.ly/4jsiLZI`
PUCRS	Software Verification and Validation	60h	`https://bit.ly/4jmvKMH`
UCSAL	Software Testing and Quality	60h	`https://bit.ly/4h49TrF`
UFAM	Software Verification, Validation and Testing	90h	`https://bit.ly/3Q5o83X`
UFC	Verification and Validation	64h	`https://es.quixada.ufc.br/`
UFG	Software Testing	64h	`https://bit.ly/3E45CpL`
UFRN	Software Testing I	60h	`https://bit.ly/4h2wyVl`
UFRN	Software Testing II	60h	`https://bit.ly/4h2wyVl`
UNB	Software Testing	30h	`http://software.unb.br/`
UNIACADEMIA	Software Testing	36h	`http://bit.ly/42iPquQ`
UNIGRAN	Verification and Validation	80h	`https://bit.ly/4gaSi05`
UNIVILLE	Software Testing I	72h	`https://bit.ly/3E5DyCt`
UNIVILLE	Software Testing II	72h	`https://bit.ly/3E5DyCt`
UTFPR	Software Testing	60h	`https://bit.ly/4gonut3`

On one hand, many courses prioritize theoretical instruction, providing a comprehensive overview of traditional software testing methods and techniques. This approach facilitates the assimilation of fundamental concepts but often lacks practical activities that enable students to apply this knowledge in real-world contexts. On the other hand, in programs that distribute testing content across multiple courses or structure it into specific modules, there is a greater tendency to integrate theory and practice. In such cases, the curriculum is reinforced by practices such as Test-Driven Development (TDD), the use of

automation frameworks, and the creation of test plans, fostering a training process more aligned with the demands of the job market.

Topics related to modern testing practices are addressed sporadically and superficially, revealing a gap between academic training and industry demands. Despite the presence of testing-focused courses in the programs analyzed, there is significant variation in the manner and depth with which these subjects are covered. This underscores the need to update curricula by integrating theory and practice and more comprehensively incorporating modern practices, thereby better preparing students for the job market.

5.1. Testing Practices Taught in IT Courses

The first survey primarily aimed to investigate students' experiences with software testing education during their undergraduate studies, assessing their familiarity with modern practices and tools used in the field. Subsequently, the research delved deeper into the students' engagement with software testing courses, including a self-assessment of the knowledge acquired and their perception of the balance between theory and practice. Participants were also able to report challenges faced during their learning process and highlight content they found lacking, providing a more comprehensive view of the gaps in the teaching of this area. In the results, we will refer to the IDs of the specific questions answered, accompanied by a general summary of the participants' responses.

5.1.1. Analysis of Participants' Knowledge of Software Testing

Most participants reported having experience with various levels of software testing (Q16), particularly Acceptance, Integration, and Unit Testing, indicating a solid foundational knowledge in the area. In contrast, Usability Testing was rarely mentioned, suggesting a potential gap in this aspect. Regarding testing techniques (Q17, Q18, and Q19), although many were familiar with functional (black-box) and structural (white-box) testing, a significant portion of participants reported not knowing or recalling the differences between these approaches, highlighting the need for more effective pedagogical strategies to consolidate these concepts.

Specifically, in structural testing (Q18), many participants had no prior contact, and among those who did, path testing was the most recognized technique, followed by control flow and data flow testing, indicating a limited exposure to these concepts. Similarly, for functional testing (Q19), while techniques such as boundary value analysis and equivalence partitioning were cited, a notable number of participants were unfamiliar with or had never applied them.

In Q20, 15.6% of participants reported no knowledge of modern testing practices, while 84.4% mentioned automated testing, demonstrating its widespread acceptance. More specialized practices, such as Test Smells (34.4%) and Flaky Tests (12.5%), were less frequently cited, indicating limited familiarity. In Q21, most participants reported experience with automated tests, especially Integration, Usability, Unit, Functional, and Acceptance tests. Graphical User Interface and Regression tests were also mentioned, reflecting concerns with stability and user experience. Conversely, specialized tests like Security, Interoperability, and Concurrency were rarely recalled. Q22 revealed that 62.5% of participants were unfamiliar with Test Smells, indicating a significant gap in teaching

this important topic for automated test quality. Q23 showed that 84.4% did not know about Flaky Tests, demonstrating limited dissemination of this concept.

In Q24, most participants reported familiarity with testing tools and frameworks, especially Selenium, JUnit, and Cypress, though some responses were generic or indicated unfamiliarity. Q25 showed more limited knowledge of test management tools, with Test Link, Test Manager, Jira, Confluence, and Jenkins being the most mentioned, but most participants reported little or no knowledge in this area. Finally, Q26 revealed that 87.5% believe software testing education in undergraduate programs needs improvement, indicating a broad consensus on the need to update and revise curricula. In Q27, the primary recommendation was to increase practical components in courses, emphasizing the use of modern industry tools. Although theory is valued, many participants expressed that insufficient practice hindered their learning, as illustrated by one comment: *"Theory is always important, but during my course, I missed practical experience. I would like to learn more tools, as we only used Pytest, unittest, Selenium, and TestLink (which I considered deplorable)."*

In Q27, participants offered various suggestions for improving software testing education. The most frequent recommendation was to increase the practical workload in courses, incorporating modern tools and integration with real-world use cases. Many acknowledged the importance of theory but emphasized that the content would be more beneficial if continuously accompanied by hands-on practice. Comments such as *"Theory should indeed be addressed, but also in parallel with practice"* illustrate this desire for a more effective balance between theoretical and practical learning. Other participants suggested the creation of multiple dedicated courses on software testing, noting that the subject is too extensive to be adequately covered in a single class. Additionally, they proposed integrating testing activities with practical projects in other courses, such as web or mobile development, allowing students to apply concepts in real.

The results indicate that although students have assimilated basic software testing content, there are significant gaps in their understanding of more advanced techniques and modern practices, such as Test Smells and Flaky Tests. The predominance of theoretical approaches, coupled with low familiarity with management tools and automated practices, suggests that current teaching does not yet adequately prepare students for the challenges of the market. The diversity in self-assessments also reveals that teaching methods are not catering equally for the different student profiles. This highlights the need to revise curricula, increasing the practical load, modernizing content and incorporating tools widely used in industry, in order to make learning more applied, inclusive and aligned with professional demands.

5.2. Testing Practices Used in Industry

The second survey aimed to investigate software testing practices adopted by industry professionals. The study covered various relevant aspects, including the most frequently used testing methodologies, tools and technologies employed in daily work, participants' experience levels with different types and techniques of testing.

5.2.1. Methodologies, Tools and Technologies Used

Analyzing the presented data, it is evident that software testing professionals adopt a diverse and adaptable approach in their processes. Regarding methodologies (Q05), organizations tend to combine various practices, with agile methods such as Scrum, Agile, and Kanban being prominent. This mixture allows greater flexibility and customization of processes, integrating practices like DevOps and, to a lesser extent, traditional methodologies such as Waterfall, highlighting the pursuit of solutions tailored to the specific needs of each project.

Concerning programming languages (Q06), respondents show a tendency to use well-established technologies, notably JavaScript and Java, which are widely employed in both web application development and robust enterprise systems. At the same time, the choice by some professionals to work with multiple languages indicates a need for versatility and adaptability to different technological contexts, whereas others prefer to specialize in a single tool, emphasizing the importance of alignment with organizational goals and ecosystems. Regarding test management tools (Q07), the data indicate a predominance of integrated solutions, with Jira standing out as the preferred option for project tracking and workflow integration. Specific tools such as TestLink and TestRail are also used, either in isolation or combined, enabling teams to tailor their management strategies according to the particularities of their testing processes.

5.2.2. Participants' Experience with Testing Types and Techniques

The analyzed data demonstrate that professionals possess extensive experience in executing various types of tests (Q08), indicating a practice that extends beyond the application of a single method. Participants have notably worked with regression(22.5%) and system testing (22.5%), which stand out in their routines, alongside integration (20.8%), unit (16%), and acceptance tests (14.2%). This diversity reflects a comprehensive approach, with no professional limiting themselves to a single form of validation.

Regarding testing techniques (Q09), there is a strong preference for functional (black-box) methods, although a significant number of respondents also reported familiarity with structural (white-box) techniques. This familiarity with both approaches suggests a tendency to combine different strategies to enhance testing effectiveness. A more detailed analysis of structural testing (Q10) shows that control flow and data flow methods are the most commonly used, followed by path testing. However, variations in the application of these techniques suggest differing levels of expertise among professionals, as well as possible inconsistencies in how these methods are classified. As for functional testing (Q11), boundary value analysis stands out as the most frequently used technique, complemented by equivalence partitioning and decision table testing. The diversity of responses may indicate different interpretations of how these methods are integrated within functional testing practices.

With respect to automated testing (Q12), functional tests remain the most widely applied, followed by regression, graphical user interface, integration, and system tests. Although approaches such as model-based testing and end-to-end testing are less frequently mentioned, the inclusion of load, performance, and stress tests highlights the

diversity of automation practices adopted. Regarding the number of test types executed, only two participants reported having performed just one of the listed test types, while the others indicated experience with a variety, suggesting that most professionals employ or have employed multiple forms of automated testing in their daily practice.

Regarding test smells (Q13), most participants reported not working with this approach. Similarly, responses to flaky tests (Q14) varied, with a significant portion of professionals indicating they do not address them. As for tools and testing frameworks (Q15), Cypress is the most widely adopted, followed by Selenium and JUnit. The use of less common tools also reflects the search for diversified solutions tailored to specific project needs. In summary, the analysis reveals that software testing professionals adopt a diverse and integrated approach, combining various types of tests, techniques, and tools. This multifaceted practice underscores their commitment to efficiency and quality, as they continuously adapt to the demands of a dynamic and evolving development environment.

5.3. Alignment of Software Testing Education with Industry Practices

Overall, the survey conducted with students revealed that 87.5% of participants acknowledge the need for improvements in software testing education. The results indicate that the content taught during undergraduate studies is predominantly theoretical, which has been identified as a barrier to the practical assimilation of concepts. The absence of courses dedicated exclusively to software testing, along with a lack of hands-on activities and use of modern tools, contributes to students not satisfactorily developing essential skills for the job market. Furthermore, although students are familiar with basic concepts and techniques, a large portion has never been exposed to more advanced practices, such as Test Smells and Flaky Tests, or to test management tools—elements considered crucial for the comprehensive training of professionals in the field.

In contrast, the survey conducted with industry professionals revealed a scenario of greater practical maturity. The professionals demonstrate consolidated knowledge in various testing techniques, ranging from basic levels (such as unit, integration, and system testing) to advanced methodologies, highlighting an effective application of testing practices. It was observed that 100% of the professionals use automated testing and management tools, which reinforces the importance of hands-on experience and continuous learning in their daily routines. Furthermore, modern concepts such as Test Smells and Flaky Tests are more widely explored in the industry—with adoption rates of 40% and 60%, respectively—indicating that the corporate environment demands a deeper and more practical mastery of testing methods.

The comparison between the two surveys highlights an important discrepancy: while academics are immersed in a predominantly theoretical environment, with 81.3% reporting knowledge of basic techniques, industry professionals show higher rates (92% knowledge of techniques and 100% adoption of automated testing). This contrast becomes even more evident when analyzing structural and functional testing, where professionals demonstrate higher percentages (72% and 76%, respectively) compared to academics (59.4% and 65.6%, respectively). Another critical point is the use of test management tools, which are adopted by all participants in the industry, while only 31.2% of academics reported having had contact with such tools. Such differences underscore the urgent need for a reassessment of academic curricula in computing courses. It is necessary for educational institutions to incorporate practical activities, laboratories, integrated

Table 4. Comparative analysis of survey results

Topics	Academia	Industry
Test Levels	96.9%	100%
Test Techniques	81.3%	92%
Structural Testing	59.4%	72%
Functional Testing	65.6%	76%
Automated Testing	84.4%	100%
Test Smells	37.5%	40%
Flaky Tests	15.6%	60%
Tools or Frameworks	81.3%	96%
Management Tools	31.2%	100%

projects, and, above all, specific courses dedicated to software testing. These measures enable a more realistic approximation of the work environment, preparing students for challenges where practical application and familiarity with modern tools are essential.

Furthermore, the analysis emphasizes the importance of raising awareness about the fundamental role of software testing in system quality. Both in academia and industry, valuing the QA field and pursuing continuous updating—through courses, certifications, and training—are crucial elements for training competent professionals and improving software development processes. quadro:analise-surveys shows the main differences in the most relevant questions.

In summary, the comparison between academic education and industry practices reveals a clear gap between the theory offered by universities and the practical demands of the market. This scenario underscores the urgent need for closer integration between academia and industry, fostering a dynamic educational environment that aligns theoretical content with the sector's real needs. Such alignment will not only contribute to the training of better-prepared professionals but also to improving the quality of developed systems and reducing costs arising from production failures.

6. Conclusion and Future Work

This study investigated the adoption of contemporary software testing practices in both academic and industrial settings, with the objective of identifying gaps and challenges across these domains. The research methodology encompassed an analysis of course syllabi and surveys conducted with students and industry professionals, thereby enabling a comprehensive comparison between academic curricula and industry practices. The findings revealed a pronounced disparity: academia predominantly emphasizes theoretical content with limited practical engagement and restricted utilization of testing tools, whereas professionals exhibit proficiency in advanced testing techniques, automation, and test management frameworks. This underscores the imperative for enhanced integration between academic instruction and industry requirements.

As future work, it is important to replicate the study on a larger scale, with a more diverse sample of academics and professionals from different regions and contexts, to increase the representativeness of the results. In addition, it is essential to investigate teaching mof theory and practice in software testing education, to overcome the limitations identified by o overcome the limitations pointed out by the students and better align academic training with market demands.

7. Statement on the use of Artificial Intelligence

The authors acknowledge the use of OpenAI's GPT-4 language model to assist with grammatical review and translation of the manuscript. Responsibility for the intellectual content remains solely with the authors.

References

Association for Computing Machinery (2020). Acm computing curricula 2020.

Barrett, A. A., Enoiu, E. P., and Afzal, W. (2023). On the current state of academic software testing education in sweden. In *2023 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, pages 397–404. IEEE.

Cammaerts, F., Tramontana, P., Paiva, A. C., Flores, N., Pastor Ricós, F., and Snoeck, M. (2024). Exploring students' opinion on software testing courses. In *Proceedings of the 28th International Conference on Evaluation and Assessment in Software Engineering*, pages 570–579.

Cheiran, J. F. P., de M. Rodrigues, E., de S. Carvalho, E. L., and da Silva, J. P. S. (2017). Problem-based learning to align theory and practice in software testing teaching. In *Proceedings of the XXXI Brazilian Symposium on Software Engineering*, pages 328–337.

Chen, N. (2011). Gate. In *Proceedings of the 33rd international Conference on Software Engineering*, pages 1078–1081.

Clarke, P. J., Davis, D. L., Chang-Lau, R., and King, T. M. (2017). Impact of using tools in an undergraduate software testing course supported by wrestt. *ACM Transactions on Computing Education (TOCE)*, 17(4):1–28.

da Silva, G. A. P., Oliveira, S. R. B., and Elgrably, I. S. (2025). Ensino e aprendizagem de testes de software a partir do uso de jogos: Uma revisão sistemática da literatura. In *Workshop sobre Educação em Computação (WEI)*, pages 36–48. SBC.

dos Santos, H. M., Durelli, V. H., Souza, M., Figueiredo, E., da Silva, L. T., and Durelli, R. S. (2019). Cleangame. In *Proceedings of the XXXIII Brazilian Symposium on Software Engineering*, pages 437–446.

Fulcini, T., Coppola, R., Ardito, L., and Torchiano, M. (2023). A review on tools, mechanics, benefits, and challenges of gamified software testing. *ACM Computing Surveys*.

Gamido, H. V. and Gamido, M. V. (2019). Comparative review of the features of automated software testing tools. *International Journal of Electrical and Computer Engineering*, 9(5):4473.

Martins, L., Brito, V., Feitosa, D., Rocha, L., Costa, H., and Machado, I. (2021). From blackboard to the office. In *Evaluation and Assessment in Software Engineering*, pages 211–220.

McManus, J. W. and Costello, P. J. (2019). Project based learning in computer science. *Journal of Computing Sciences in Colleges*, 34(3):38–46.

Melo, S. M., Moreira, V. X., Paschoal, L. N., and Souza, S. R. (2020). Testing education. In *Proceedings of the XXXIV Brazilian Symposium on Software Engineering*, pages 554–563.

Muneer, I. (2014). Systematic review on automated testing (types, effort and roi).

Myers, G. J., Sandler, C., and Badgett, T. (2011). *The art of software testing*. John Wiley & Sons.

Paschoal, L. N. and de Souza, S. d. R. S. (2018). A survey on software testing education in brazil. In *Proceedings of the 17th brazilian symposium on software quality*, pages 334–343.

Smith, J., Tessler, J., Kramer, E., and Lin, C. (2012). Using peer review to teach software testing. In *Proceedings of the ninth annual international conference on International computing education research*, pages 93–98.

Sneha, K. and Malle, G. M. (2017). Research on software testing techniques and software automation testing tools. In *2017 international conference on energy, communication, data analytics and soft computing (ICECDS)*, pages 77–81. IEEE.

Tramontana, P., Marín, B., Paiva, A. C. R., Mendes, A., Vos, T. E. J., Amalfitano, D., Cammaerts, F., Snoeck, M., and Fasolino, A. R. (2024). State of the practice in software testing teaching in four european countries. In *2024 IEEE Conference on Software Testing, Verification and Validation (ICST)*, pages 59–69.

Valente, M. T. (2024). *Software Engineering: A Modern Approach*. Self-published.

Valle, P. H. D., Barbosa, E. F., and Maldonado, J. C. (2015). Cs curricula of the most relevant universities in brazil and abroad. In *2015 International Symposium on Computers in Education (SIIE)*, pages 62–68. IEEE.

Venson, E. and Alfayez, R. (2024). Bridging theory to practice in software testing teaching through team-based learning (tbl) and open source software (oss) contribution. In *Proceedings of the 46th International Conference on Software Engineering: Software Engineering Education and Training*, ICSE-SEET '24, page 72–81, New York, NY, USA. Association for Computing Machinery.