An Experimental Evaluation of Peer Testing in the Context of the Teaching of Software Testing

Jacson R Barbosa, Pedro Valle, José Maldonado, Márcio Delamaro
Instituto de Ciências Matemáticas e de Computação (ICMC)
Universidade de São Paulo (USP)
São Carlos/SP, Brazil, 13560-970
Email: {jacsonrb, pedrohenriquevalle}@usp.br,
{jcmaldon, delamaro}@icmc.usp.br

Auri M. R. Vincenzi
Departamento de Computação
Universidade Federal de São Carlos
São Carlos/SP, Brazil, 13565-905
Email: auri@dc.ufscar.br

***Abstract*—Context: Software testing is an important task in software product quality assurance. Alternatives approach for teaching and training students and professionals have been addressed in many studies, recently. In this article, peer testing is investigated as one alternative approach to mitigate the current lack of well-trained professionals. Objective: To verify the efficiency and efficacy of peer testing in order to create test cases to promote software testing education and training. Method: Conduct a controlled experiment in which subjects design and implement test cases for three small software units. Results and conclusions: In this study, it has been observed that peer testing was significantly more efficient than individual testing (with t-Test). On the other hand, it has not been observed significant differences in the efficacy of peer testing.**

***Keywords*—Software Testing Education; Peer Testing; Experimental Evaluation**

I. Introduction

The software industry seeks to develop high-quality software products and systems. In this perspective, verification and validation activities, especially software testing, have been investigated and improved over the years. Specifically, Software testing aims to run programs or models with specific entries, making sure these artifacts behave as expected, leading to a detailed failure analysis by means of debugging to eliminate defects that cause failures [1].

Software testing to be productive and useful should be carried out with the support of testing tools and strategies established based on evidence, like any other software engineering activity. Despite software testing being recognized as an important activity in quality assurance of software products, many students feel unmotivated to learn contents related to software testing [2]. In this scenario, some researchers observed that it is essential to develop instruments that facilitate the teaching and training of software testing, in order to increase motivation to work with the software testing [3].

In this context, it is essential to investigate the use of different approaches to aid the teaching and learning software testing activities and processes in the perspective of establishing the cost-effective testing strategy. To characterize the main approaches used in software testing education Valle, Barbosa and Maldonado (2015) [4] carried out a systematic mapping in wich they identified 11 different initiatives to support software testing education, as: Educational Modules, Educational Games, Test Driven Development (TDD) and Integrated Teaching of Software Testing and Programming and Peer Testing [4].

In this scenario, peer testing has been investigated since it is known to provide a playful and competitive learning, in which students learn from each other. The aim is to produce shreds of evidence in order to contribute to enriching the teaching and learning environments and underlying processes. The existing approaches include peer and individual testing [4].

Peer testing allows collegial and competitive learning, in which students learn from each other [5]. It has the potential to improve the quality of generated test cases, as it allows the exchange of knowledge among participants, contributing to the identification of quality deviations and corrective actions adoption during the tests. In this approach, there is a person responsible for preparing test cases and another for evaluating and giving suggestions about the test cases generated, who serves as a reviewer of the deliverables [6, 7].

The main objective of this paper is to investigate the use of peer testing as one alternative approach to mitigate the current lack of well-trained professionals. By means of a controlled experiment, it is presented the comparison of efficiency and effectiveness of peer testing with individual testing in the preparation of test cases in the context of software testing teaching at the university. In this paper, functional testing is the focus. The most commonly used criteria for functional testing are Equivalence Class Partitioning (ECP) and Boundary Value Analysis (BVA) [8, 9]. Functional testing also called black box testing is based only on the specification of the program under test, which is evaluated according to the user's point of view since only the inputs and outputs of the program are provided [10].

This paper is organized as follows. Section II describes the main concepts of software testing and peer testing. Section III discusses initiatives of using peer testing in software testing teaching. In Section IV are described the planning of the controlled experiment and in Section V, the specific steps during the preparation and implementation of the experiment. Section VI synthesizes the results concerning the effectiveness and efficiency of the peer testing in developing test cases and carrying out the testing activity. Section VII summarizes the experimental threats to validation. Finally, in Section VIII,

conclusions and future research are discussed.

II. Background

A. Software Testing

Software testing provides directions to execute programs or models with specific entries to observe whether these artifacts behave as expected. It involves performing a detailed dynamic analysis of the software under testing in order to identify potential failures and, subsequently, by means of debugging, to eliminate defects. Software testing is a dynamic activity since it is based on running programs or models [1, 11].

Software testing techniques are usually either Functional, or Structural, or Defect-based and testing strategies are established using their complementary aspect in terms of efficiency and efficacy [1, 10]. Each technique induces a set of software testing criteria. As mentioned before, this paper focuses on Functional Testing. Functional testing criteria are used in most of the software development testing level, such as unit, integration, system, and acceptance testing, regardless of the software development paradigm used. In this technique, the software is evaluated according to the user's point of view. To use this technique, it is necessary to run the program under test with the selected test data from the input domain, according to the previously planned testing criterion [1, 10].

In general, exhaustive testing is impossible to be applied in practice during software testing activity, because it demands the heavy use of resources, especially time [12]. Thus, it is necessary to use criteria for test activity. Each criterion divides the input domain into different subdomains and, consequently, test suites can be used to satisfy a given criterion [1]. The most commonly used criteria for functional testing are Equivalence Class Partitioning and Boundary Value Analysis. One advantage of the functional testing is that it requires only the specification of software to derive test requirements. Thus, this technique can be used in any paradigm, such as procedural, object-oriented, among others. A disadvantage of this technique is that it cannot ensure that critical parts of the code were tested, because functional testing does not parse the source code [1, 8].

The defect-based test technique, using the mutation test criterion, enables the exploration of common defects in the software development process, since the software being tested is modified countless times, producing a set of mutant programs. Choosing test cases that show the difference in behavior between the original software and the mutants is attributed by the tester [1].

According to the competent programmer hypothesis and the coupling effect it is possible to argue that given a set of test cases capable of distinguishing a set of selected mutants and representing many of the most common defects, it is capable of revealing the existence of other types of defects [1].

In addition to testing the software, the mutation test criterion can also be considered as a technique to aid in the process of improving test cases to make them more efficient, that is, mutation testing can be used as a technique to evaluate the quality of test cases [13].

B. Peer Testing

Peer testing leads students and practitioners attempt at finding defects in the code or test cases developed by their peers. This technique has attracted researchers' attention for its feature of promoting competition, fun, and excitement in teaching the fundamentals of programming and software testing [14, 5]. In this way, peer testing often reveals problems in artifacts (codes and test cases) that were drawn up by students or practitioners, since they are assessed by partners that have reviewer's role. One of the main advantages of using this peer review is that it allows testers to have contact with a lot of ideas for the development of codes and test cases, exchange knowledge and learn with their peers [5].

According to the study by Clark (2004), peer testing offers the following benefits: i) observe the importance of software testing; ii) increase the quality of the product tested; iii) work on the product being tested; iv) promote communication between co-workers; v) increase collaboration between the teams; and vi) increase learning [15].

III. Related Work

From an informal literature review, the authors could not identify researchers that assess the efficiency and effectiveness of peer testing in the scope of construction test cases processes. Papers were identified that had investigated the use of peer testing to complement the teaching of computing. The paper proposed by Smith (2012) describes the experience in incorporating tests by pairs of students in a data structure course. The author aimed to frame the test as a competitive and fun activity, allowing students to learn from each other and to demonstrate the importance of testing [5]. McDowell et al. (2002) in an introductory course observed that good quality and high rate of completion are obtained using pair programming [16].

Since there are very few controlled experiments assessing the use of peer testing for constructing functional testing cases, a controlled experiment is described in the next sections.

IV. Experimental Study Definition

In this section, the planning of a controlled experiment to compare peer and individual testing is presented, using the process proposed by Wohlin [17].

Figure 1 illustrates an overview of the experiment described in this work. To start the experiment, each subject signed a consent form and also filled in a form characterizing previous knowledge related to the content covered in the study.

Then students were divided into two groups. In the first group, the students individually carried out three software testing sessions. In the second group, the students carried out the same session, but in pairs, i.e., applying peer testing. In which one of the partners of the pair elaborates the test cases to satisfy the criteria of partitioning in classes of equivalence and analysis of the limit value, then it consults the partner to verify if it agrees with the test cases generated, if there is disagreement with relation to some test case both should agree. In the next exercise, there is the position exchange, that is, who was creating the test cases will now only give suggestions to the colleague.

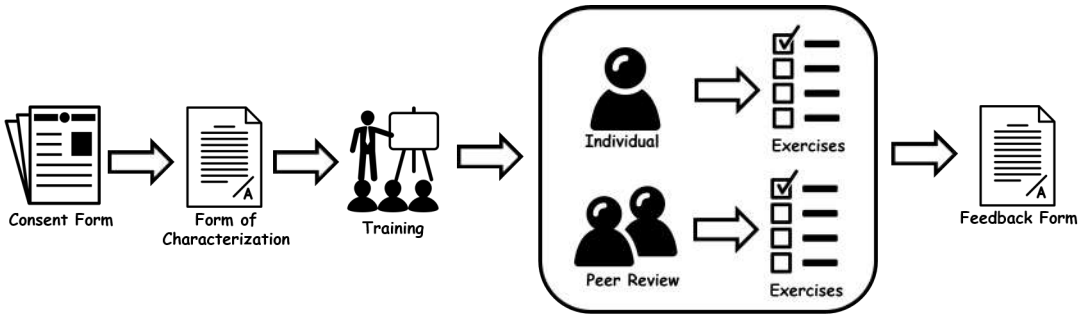

Figure 1. Overview of the controlled experiment

It is important to note that both groups carried out the testing activity using the same software. In the end, the subjects filled in a feedback form about the experiment.

In the next subsections, we present the objectives, research questions, selection of subjects, formulation of the hypotheses, experiment design and the instrumentation of the performed experiment.

A. Specific Aims

Our experiment can be summarized using the template by [17], as follows:

Analyze peer testing for the construction of test cases

for the purpose of assessing peer testing

with respect to effectiveness and efficiency

from the point of view of the researcher

in the context of software testing education for undergraduate students.

B. Definition of Research Questions and Metrics

To achieve the purpose of the experiment, the questions below are defined:

- RQ_A: Is peer testing more effective than individual testing for the construction of test cases?
- RQ_B: Is peer testing more efficient than individual testing for the construction of test cases?

The metrics associated with the experiment are:

- Effectiveness: The mutation score of the functional test cases generated to measure the quality of the set of test cases.
- Efficiency: Construction time of the functional test cases.

C. Selection of Subject

The subjects (27 undergraduate students) were drawn from the Software Testing and Inspection discipline offered in the seventh period of the Computer Science course of the ICMC-USP. To participate in the study, the students needed:

- To express interest in participating in the study, by signing the consent form.
- To answer the characterization form of each subject's level of knowledge.
- To participated in the training on functional testing (Equivalence Class Partitioning and Boundary Value Analysis).

The subjects were divided at random and assigned to work in pairs or individually.

D. Selection of software

We selected three software (MaxMin2, ExactMatch, and PartialSorting) out of thirty-two previously used in another type of experiment [18], minimizing the risks associated with the interpretation of the same specifications.

The first software (MaxMin2) is a program to obtain the maximum and minimum values of a given set. The second one (ExactMatch) is a program that implements the exact-match algorithm to search in strings. Finally, the third one (PartialSorting) is a program to get the first k elements of an ordered set of size n.

E. Hypothesis Formulation

In this work the following hypotheses are formulated from RQ_A and RQ_B:

- **Null hypothesis ($H0_A$):** Peer testing technique and individual testing technique are equally effective for a construction of test cases.
- **Alternative hypothesis ($H1_A$):** Peer testing has not the same effectiveness as individual testing for the construction of test cases.
- **Null hypothesis ($H0_B$):** Peer testing and individual testing are equally efficient for the construction of test cases.
- **Alternative hypothesis ($H1_B$):** Peer testing has not the same efficiency individual testing for the construction of testing cases.

F. Experimental Design

The following items were defined in the design of the experiment:

- **approach:** A factor and two treatments.

- **factor:** Construction method of software test cases.
- **treatment:** Peer testing and individual testing for the construction of test cases.
- **control:** Individual testing for the construction of test cases.

The Table I illustrates an example of how the subjects were allocated in the experiment. The undergraduate class was randomly divided into two groups in which students participated in the experiment either individually or collaboratively.

Table I. EXAMPLE OF THE DISTRIBUTION OF SUBJECTS

Exercises		
Subjects	**Peer testing**	**Individual testing**
1,2	X	
3,4	X	
5,6	X	
7		X
8		X
9		X
...	...	..

At the end of the experiment, there was the same amount of results generated from peer testing and individual testing.

G. Instrumentation

A brief description of the documents used in the experiment follows:

- **Consent form:** The subjects signed a form to express agreement to participate in the experiment.
- **Characterization Form:** The subjects answered a questionnaire about their knowledge in software testing and other supplementary information.
- **Training material:** We conducted a training exercise with the subjects about the criteria for functional software testing (on Equivalence Class Partitioning and Boundary Value Analysis) and on peer testing.
- **Exercises:** The subjects faced three software test session, according to the level of difficulty on the computer, as classified by experimenters as easy, intermediate or difficult. To carry out the activity, we have been used a standard laboratory at ICMC-USP with the following basic software: Java, Eclipse, JUnit, and LibreOffice.
- **Template for test cases:** The subjects created test cases according to the established template. The students also used the template to record the beginning and the end of the sessions.
- **Feedback questionnaire:** The subjects answered a feedback questionnaire about the experiment. In this questionnaire, there was a question on the subjects authorizing or not the use of their data in the experiment.

V. IMPLEMENTATION OF THE EXPERIMENT

Following the definition of the experiment's protocol, it was performed a pilot study in order to verify the acceptability of the experimental design and the corresponding documents and process. The pilot study was conducted at UFSCar (Federal University of São Carlos) with seven students from the 4th period of the computer engineering course; two doubles worked running peer testing and three students working individually.

After the pilot study, considering the students' suggestions, the protocol was updated: using software specifications which had already been implemented and validated in a previous experiment; including the specification of test cases with JUnit, as complementary activity and adapting the Template for test cases. Once the Protocol was consolidated, the experiment was run at ICMC-USP. During the training, in addition to the presentation of functional criteria, it was presented the roadmap to be followed by the doubles that made use of peer testing. Immediately after completion of the training, it was performed the random distribution of the subjects to begin the settlement of sessions in pairs or individually.

VI. ANALYSIS AND DISCUSSION OF THE RESULTS

A. Reducing the Data Set

Initially, all the collected data were reviewed in order to verify their consistency. Among the 27 participants of the experiment, two subjects did not deliver any of the expected results, and another did not deliver the results concerning the last two. In this case, these 3 participants were excluded from the set, remaining 24 subjects. They were randomly divided into two group: who worked alone (eight subjects), and that worked by applying peer testing (eight doubles).

All subjects were informed after training that they would have 100 minutes to complete the settlement of the three sessions, however, some participants did not submit the solution of the third session to the repository or did it with a bad quality level. Given that, it was decided to discard the third session from the analysis.

B. Characterization of the Participants

Concerning participants' profile, all of them had prior knowledge/experience in functional test, probably acquired during the first half of the discipline of Software Testing and Inspection. However, 57.1% of the participants did not know the peer testing. Given this, the training was necessary to level the knowledge of all participants.

C. Descriptive Analysis of the Data

Table II presents the mean and standard deviation (SD) for the key metrics (mutation coverage metric (MC) and time to resolution (TR) of each session) associated with the experiment. The mutation coverage metric is the score of mutation of the test cases set generates with Pitest tools [19], i.e., the effectiveness of the functional test criteria. The efficiency is defined by the least amount of time (in minutes) for each session.

Thus, from Table II, it can be seen that the subjects who worked in pairs presented the lowest average time. With respect to the mean mutation coverage, the doubles also achieved the best performance, suggesting that the groups working in pairs can eliminate more mutants.

Table II. DESCRIPTIVE STATISTICS OF THE METRICS

Test session	Treatment	TR (min)		MC (%)	
		Mean	SD	Mean	SD
Session 1	Peer Testing	26.87	10.86	57.12	16.48
	Individual testing	39.25	9.49	53.25	33.28
Session 2	Peer Testing	18.87	3.52	83.75	15.60
	Individual testing	30.62	9.85	62.37	42.98

By analyzing the figures 2 and 3 which have the mutation coverage in percentage, it is observed that the group who worked in pairs presented a minor range and higher value in two sessions in relation to the other group.

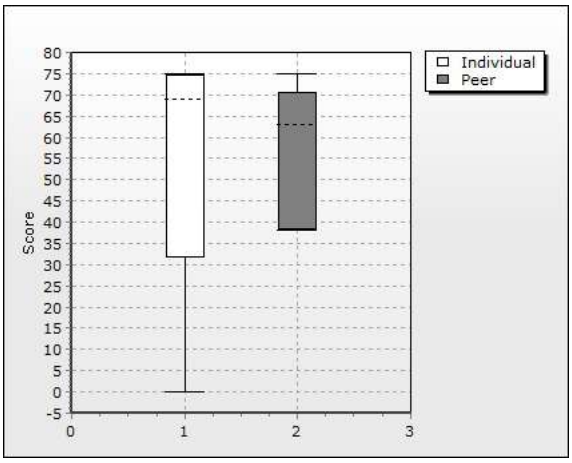

Figure 2. Box Plot of the Coverage Criteria Analysis of mutants-Session 1

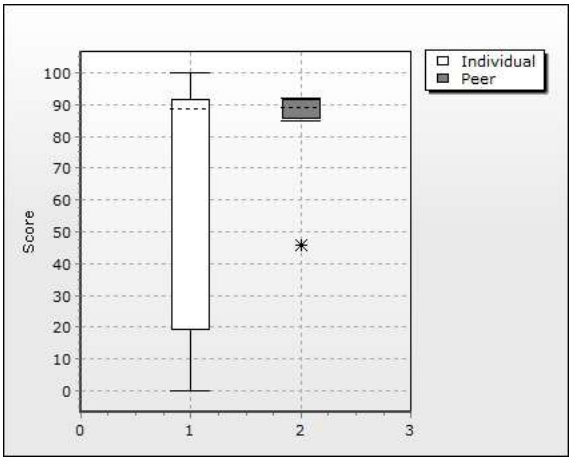

Figure 3. Box Plot of the Coverage Criteria Analysis of mutants-Session 2

Concerning the time (Figures 4 and 5), the team that worked in pairs managed to conclude the activities faster (in both sessions) and with a minor variation (second session).

D. Analysis of the Hypotheses

To evaluate the hypothesis presented in Section IV-E, the t-test was conducted to compare the two independent samples (a factor with two treatments). In this experiment, using the t-Test aims to check if there is a significant difference (p is less than 0.05) in the development of test cases between subjects who worked collaboratively (pairs) and those who worked alone.

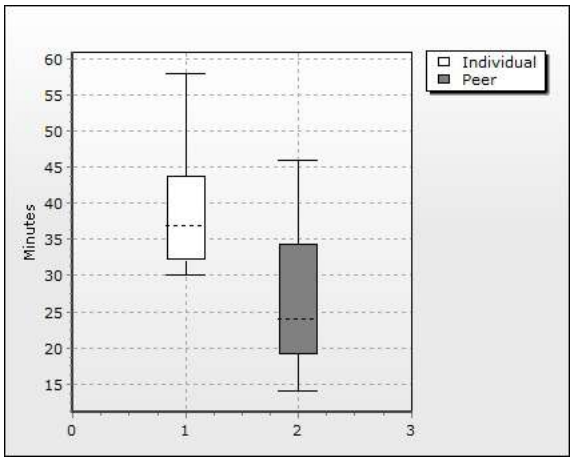

Figure 4. Box Plot of the time to resolution of Session 1

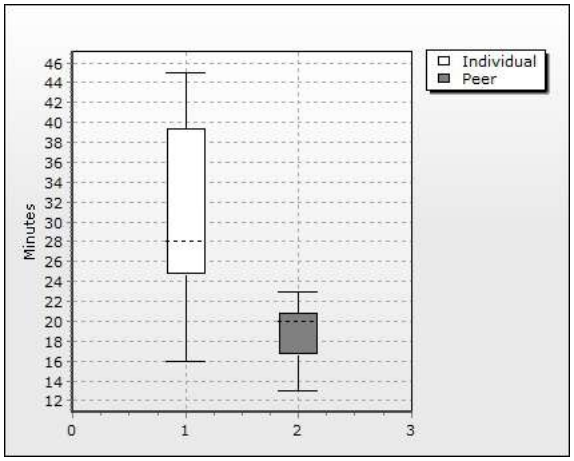

Figure 5. Box Plot of the time to resolution of Session 2

1) Effectiveness: Considering that the value is not significant in the t-test for the first session (p = 0.7297) and for the second (p = 0.1942), the null hypothesis can not be denied. As a result, the hypothesis that the peer testing is more effective than the individual testing in the construction of test cases cannot be accepted.

2) Efficiency: Whereas the value is significant in testing for the first session (p = 0.0294) and for the second (p = 0.0067), it is possible to deny the null hypothesis. Due to that, we can accept the hypothesis that the peer testing is more efficient than the individual testing in the construction of test cases.

VII. EXPERIMENTAL VALIDITY

Following, it is presented the risks to the validity of the results of the experiment. The risks identified are classified in conclusion validity, internal validity, and external validity.

A. Conclusion Validity

It refers to find out correct conclusion regard between treatment and experiment's result. The templates of the test cases can be ambiguous difficulting the collection of relevant information. To solve this, the templates have been validated with an expert. Another factor that may have influenced is

the descriptions of software, compromising the quality of the set of test cases. The maximum time set for the resolution of the exercises also represents a threat, since the time was not enough for most of the subjects to conclude the tasks.

B. Internal Validity

It occurs when a factor not identified impacts the causality relationship between treatment and result, without experimenter's knowledge. The fact that the selection of the participants has not been adequately random can be considered a threat to internal validity, once they were taken from the availability of students.

Another threat would be plagiarism which is a common risk in conducting experimental studies in the academic environment with the possible exchange of information among participants on the tasks of the study. However, this threat is not relevant, since the participants pledged not to communicate during the study. In addition, it should be highlighted that the results were not used in the assessment for the discipline.

side eye 00

C. External Validity

It refers validity the experiment's results generalization for industrial practice. The experience of the participants can be considered as a threat to external validity. The sample selected was quite heterogeneous. However, they are undergraduate students and most have little or no professional experience. Finally, the development environment can be classified as a threat to validity, as well. The academic production environment used by participants may not simulate the entire industrial development environment.

VIII. Conclusions and Future Work

Since there are few experimental studies investigating the use of peer testing in the context of software testing, this paper contributes in perspective to advance the state of the art concerning peer testing. This study aimed to define and implement a protocol of a controlled experiment to evaluate the effectiveness and efficiency of the construction process of test cases when the subjects use peer testing compared to individual testing. It has been observed that peer testing was significantly more efficient than individual testing (with t-Test). On the other hand, it has not been observed significant differences in the efficacy of peer testing.

As future work, it would be interesting to replicate such an experiment with students and experienced professional, eventually with teams with distinct levels of expertise and experiences.

Acknowledgments

We acknowledge the Brazilian research agency CAPES and CNPq for funding this research.

References

[1] M. E. Delamaro, J. C. Maldonado, and M. Jino, *Introdução ao Teste de Software*. Elsevier, 2016.

[2] S. Jia and C. Yang, "Teaching software testing based on cdio," *World Transactions on Engineering and Technology Education*, vol. 11, no. 4, 2013.

[3] W. Wong, A. Bertolino, V. Debroy, A. Mathur, J. Offutt, and M. Vouk, "Teaching software testing: Experiences, lessons learned and the path forward," in *XXIV Conference on Software Engineering Education and Training (CSEE&T)*. Honolulu, USA: IEEE, 2011.

[4] P. H. D. Valle, E. F. Barbosa, and J. C. Maldonado, "Um mapeamento sistemático sobre ensino de teste de software," in *Simpósio Brasileiro de Informática na Educação*, 2015, p. 71.

[5] J. Smith, J. Tessler, E. Kramer, and C. Lin, "Using peer review to teach software testing," in *Proceedings of the Ninth Annual International Conference on International Computing Education Research*. ACM, 2012.

[6] K. E. Wiegers, *Peer reviews in software: A practical guide*. Addison-Wesley Boston, 2002.

[7] P. Rigby, B. Cleary, F. Painchaud, M. A. Storey, and D. German, "Contemporary peer review in action: Lessons from open source development," *IEEE Software*, 2012.

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

[9] I. Sommerville, *Engenharia de Software*, 9th ed., P. P. Hall, Ed., São Paulo, 2011.

[10] M. A. Khan and M. Sadiq, "Analysis of black box software testing techniques: A case study," in *International Conference and Workshop on Current Trends in Information Technology*. IEEE, 2011.

[11] R. S. Pressman, *Engenharia de software*, 7th ed. McGraw Hill Brasil, 2011.

[12] R. S. Wazlawick, *Engenharia de Software: conceitos e práticas*. Rio de Janeiro: Elsevier, 2013.

[13] R. Just, "The Major mutation framework: Efficient and scalable mutation analysis for Java," in *Proceedings of the International Symposium on Software Testing and Analysis (ISSTA)*, San Jose, CA, USA, July 23–25 2014, pp. 433–436.

[14] K. Anewalt, "Using peer review as a vehicle for communication skill development and active learning," *Journal of Computing Sciences in Colleges*, 2005.

[15] N. Clark, "Peer testing in software engineering projects," in *Sixth Australasian Conference on Computing Education*. ACM, 2004.

[16] C. McDowell, L. Werner, H. Bullock, and J. Fernald, "The effects of pair-programming on performance in an introductory programming course," *SIGCSE Bull.*, vol. 34, 2002.

[17] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in software engineering*. Springer Science & Business Media, 2012.

[18] S. do Rocio, S. de Souza, M. Paiva Prado, E. Francine Barbosa, and J. C. Maldonado, "An experimental study to evaluate the impact of the programming paradigm in the testing activity," *CLEI Electronic Journal*, vol. 15, 2012.

[19] H. Coles, T. Laurent, C. Henard, M. Papadakis, and A. Ventresque, "Pit: A practical mutation testing tool for java (demo)," in *Proceedings of the 25th International Symposium on Software Testing and Analysis*, ser. ISSTA 2016. New York, NY, USA: ACM, 2016, pp. 449–452. [Online]. Available: http://doi.acm.org/10.1145/2931037.2948707