Informatics in Education, 2023, Vol. 22, No. 1, 99–120
© 2023 *Vilnius University, ETH Zürich*
DOI: 10.15388/infedu.2023.01

An Experimental Study on a Conversational Agent in Software Testing Lessons

Leo Natan PASCHOAL[1], Silvana Morita MELO[2],
Vânia de Oliveira NEVES[3], Tayana Uchôa CONTE[4],
Simone do Rocio Senger de SOUZA[1]
[1]*Universiy of Sao Paulo – São Carlos, Brazil*
[2]*Federal University of Grande Dourados – Dourados, Brazil*
[3]*Fluminense Federal University – Niterói, Brazil*
[4]*Federal University of Amazonas – Manaus, Brazil*
e-mail: paschoalln@usp.br, silvanamelo@ufgd.edu.br, vania@ic.uff.br,
tayana@icomp.ufam.edu.br, srocio@icmc.usp.br

Received: September 2021

Abstract. Nowadays, few professionals understand the techniques and testing criteria to systematize the software testing activity in the software industry. Towards shedding some light on such problems and promoting software testing, professors in the area have established Massive Open Online Courses as educational initiatives. However, the main limitation is the professor's lack of supervision of students. A conversation agent called TOB-STT has been defined in trying to avoid the problem. A previous study introduced TOB-STT; however, it did not analyze its efficacy. This article addresses a controlled experiment that analyzed its efficacy and revealed it was not expressive in its current version. Therefore, we conducted an in-depth analysis to find what caused this result and provided a detailed discussion. The findings contribute to the TOB-STT since the experimental results show that improvements need to be made in the conversational agent before we use it in Massive Open Online Courses.

Keywords: chatbot, computer science education, software testing.

1. Introduction

Software testing aims to offer software quality based on identifying defects that persist in the software under test (Myers *et al.*, 2011). Such defects can be costly for an organization since they cause unwanted effects (e.g., security breaches, loss of data and information, damage to the environment, financial losses, among others (Tan, 2016; Zhivich and Cunningham, 2009)) for both the organizations that develop them and those that maintain the software, and can be avoided by the test activity.

Although the importance of software testing has been recognized by a significant part of the software engineering community, it is commonly overlooked by computing curricula (Fraser *et al.*, 2018). The attention given by ACM (ACM, 2016) and IEEE-CS (IEEE, 2013) reference curricula has shown it is generally included as a unit among the various topics of a software engineering course (Benitti, 2018), causing more complete test approaches are to be no longer taught (Paschoal and Souza, 2018). Due to the volume to be addressed in this field and the workload limit of the course, the testing practice is set aside (Paschoal and Souza, 2018). However, software testing learning requires more practice than is commonly addressed (Fraser *et al.*, 2018).

The lack of attention to software testing teaching has caused students to graduate in computing with no knowledge of this subject (Lemos *et al.*, 2017), and software organizations to hire unskilled professionals. According to Beneditti (Benitti, 2018), the lack of qualified professionals in the software industry may be one of the main reasons software organizations do not have a mature process to undertake the activity. Several studies have corroborated this opinion, claiming professionals with no appropriate university education and sufficient training are hired (Garousi and Zhi, 2013; Ng *et al.*, 2004; Melo *et al.*, 2020).

Professionals with no proper understanding of testing practice or necessary skills contribute to the current state of testing practice adopted by a significant number of software organizations and expressed in worldwide surveys – whose participants are industrial testing professionals – characterized by tests (or test cases) randomly established for the practice of software systems under test (Garousi and Zhi, 2013; Ng *et al.*, 2004; Melo *et al.*, 2020; Dias-Neto *et al.*, 2017). The lack of knowledge in the software testing area is believed to be a predominant factor for organizations' failure in adopting criteria that support an effective selection of test case sets (Benitti, 2018).

Graduates who work professionally in the software testing field seek alternative ways to acquire knowledge in the area. Some learning strategies used by those professionals are resources available online (Garousi and Zhi, 2013), among which are courses designed within the scope of MOOC platforms (Massive Open Online Course) (Fassbinder *et al.*, 2017; Prates *et al.*, 2018). Such courses are open alternatives offered by recognized educational institutions and that can meet demands and improve the development of some apprentices' skills.

Particularly MOOC can be seen as an initiative for professionals and organizations to improve their skills and know testing techniques and criteria that help establish test case sets (Enoiu, 2019). According to Enoiu (2019), software testing MOOCs can also be an opportunity for teachers to disseminate the techniques and technologies defined or addressed in their research in an industrial context, also supporting innovation in information technology companies. Therefore, they can potentially contribute to software testing learning in non-formal education.

However, MOOCs have limitations. Recent studies have reported interaction and feedback as the main factors for students' evasion from courses (Mittal *et al.*, 2018; Mikic-Fonte *et al.*, 2018; Aguirre *et al.*, 2018), probably because the courses available on those platforms are accessed by many students and rarely supported by teachers or tutors. Studies that leverage such discussions also mentioned the use of conversational agents[1] as a resource to supporting MOOCs. In this sense, they promote interaction between students enrolled in the courses and agents, who solve doubts and receive adequate feedback. It is in this context that the opportunities for investigations on the use of TOB-STT, a support mechanism for teaching software testing (Paschoal *et al.*, 2019) in software testing MOOCs, arise in such a context.

TOB-STT was presented as software that interacts with software testing students in natural English and solves their doubts on issues associated with software testing concepts, criteria, and techniques (Paschoal *et al.*, 2019). Although introduced in a previous study, that study focused on understanding whether the knowledge bases of the conversational agent are representing the necessary knowledge to help students in the software testing area. Therefore, an evaluation that considers the efficacy of educational support offered by the TOB-STT to software testing students has not yet been carried out. Before TOB-STT is considered in an effective deployment of a software testing MOOC, it must be evaluated regarding its satisfactorily achieving its objective. We argue here that there is still not enough evidence to confirm that the TOB-STT offers educational support for students with doubts.

This article reports an experimental study for the identification of the efficacy of the TOB-STT support to software testing students involved in an educational activity with no teacher’s support.

The paper is organized as follows: Section 2 introduces TOB-STT; Section 3 addresses studies on conversational agents in computer science education; Section 4 describes the planning of the experimental study. Sections 5 and 6 are devoted to presenting the results and threats to validity, respectively; Section 7, provided a discussions on the main results. Section 8, reports some study limitations; finally, Section 9 provides the conclusions.

2. An Overview on TOB-STT

TOB-STT is a rule-based conversational agent that has knowledge bases on software testing content, written in AIML language (Artificial Intelligence Markup Language[2]). If was defined to subsidize software testing activities taught through distance learning. In this perspective, it has an interface similar to a chat to be used by students to express doubts and opinions and receive feedback on uncertainties and convictions. Such a teaching support mechanism uses natural language to communicate and performs linguistic treatments on requests sent by students to produce appropriate feedback.

The agent was designed to support the three different stages of a conversation. Students can introduce themselves, greet the agent (start phase of the conversation), interact on the software testing content (development of the conversation), and say goodbye to the agent (end of the conversation). Regarding the interaction over the domain to which it was established, it can recognize four types of intention, shown in

[1] Conversational agents are systems that interact with their users in natural human language through natural language processing (Montenegro *et al.*, 2019).

[2] A complete description of the language is provided by Wallace (2009).

Table 1
Set of intentions recognized by TOB-STT

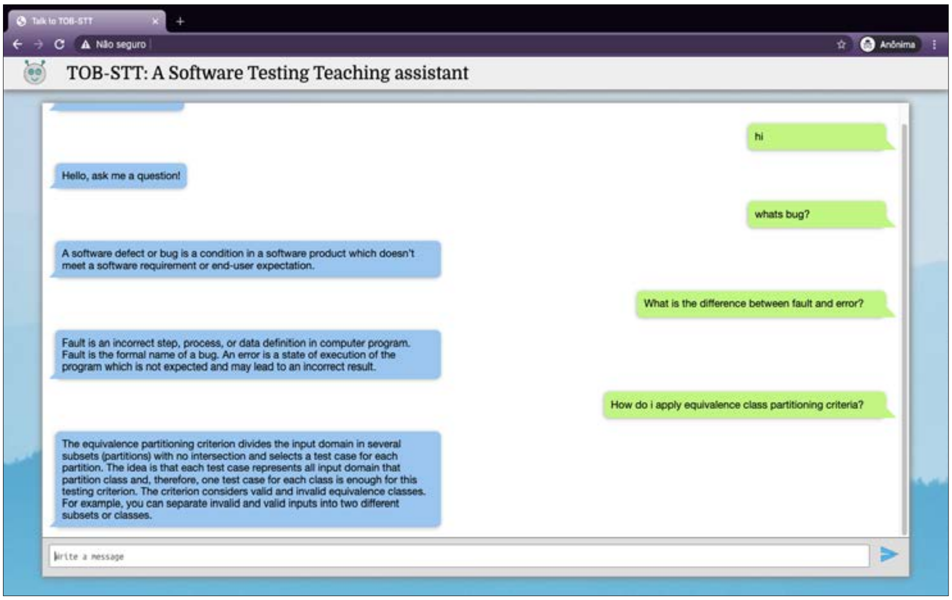
Intention	Intention description	Examples
Define	TOB-STT is expected to recognize questions or comments related to the determination of any term, concept, or jargon in the software testing area.	What is software testing? Can you describe the functional testing criteria?
Confront	TOB-STT can identify questions or comments whose nature is associated with the effect of differentiating two or more terms, concepts, or jargon in the software testing area.	What is the difference between functional testing and structural testing? Make a parallel among defects, errors, and failures.
Apply	TOB-STT can detect questions or comments related to both use and demonstration of some software testing criteria.	How can the boundary-value analysis criterion be applied? How should I use the equivalence class partitioning criterion.
Exemplify	TOB-STT can distinguish questions or comments that ask for real examples of software testing techniques and criteria.	Provide examples of the use of the boundary-value analysis criterion. Provide examples of applications of the equivalence partitioning criterion.

Table 1, together with their definition and some examples of dialogues accepted. After understanding what the student wishes to know, the agent sends a message directed to that request.

The authors (Paschoal *et al.*, 2019) provided technical information associated with the algorithms and technologies used for TOB-STT establishment. The current TOB-STT version is available for use at <`icmc.usp.br/e/89f12`>. Since it was conceived in a free software format for encouraging the community to contribute to its development, its source codes are available for use and modification by the software engineering community at <`icmc.usp.br/e/4644c`>.

TOB-STT can adapt to different device interfaces (e.g., desktop, tablets, smartphones) towards encompassing the different modes of access to educational content and educational paradigms (i.e., mobile learning and ubiquitous learning). Fig. 1 displays its interface from a web browser (Fig. 1a) and smartphone (Fig. 1b). Fig. 1a also shows an example of a dialogue between TOB-SST and a student. The student starts interaction by asking what a “bug” is. In response, the agent provides an explanation considering an existing definition in SWEBOK (Software Engineering Body of Knowledge) and tries to make clear a “bug” has the same definition of a defect. The student then asks the agent to differentiate between defect and error, ending interaction by asking it to explain how to apply the testing criterion known as equivalence partitioning.

TOB-STT is similar to other conversational agents from the literature (Mikic-Fonte *et al.*, 2018; Katchapakirin and Anutariya, 2018; Ocaña *et al.*, 2019; Herpich *et al.*, 2016; Leonhardt *et al.*, 2007; Paschoal *et al.*, 2018). The next section shows an overview of the state of the art regarding the conversational agent to support de computer science (CS) education.



TOB-STT: A Software Testing Teaching assistant

hi

Hello, ask me a question!

whats bug?

A software defect or bug is a condition in a software product which doesn't meet a software requirement or end-user expectation.

What is the difference between fault and error?

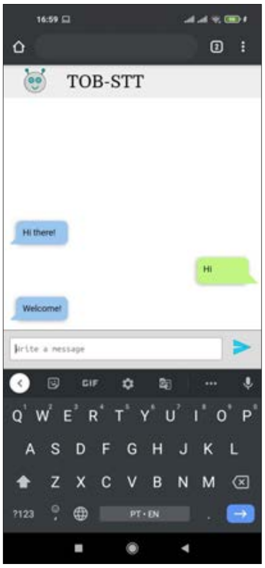
Fault is an incorrect step, process, or data definition in computer program. Fault is the formal name of a bug. An error is a state of execution of the program which is not expected and may lead to an incorrect result.

How do i apply equivalence class partitioning criteria?

The equivalence partitioning criterion divides the input domain in several subsets (partitions) with no intersection and selects a test case for each partition. The idea is that each test case represents all input domain that partition class and, therefore, one test case for each class is enough for this testing criterion. The criterion considers valid and invalid equivalence classes. For example, you can separate invalid and valid inputs into two different subsets or classes.

Write a message

(a) TOB-STT being accessed from a desktop



TOB-STT

Hi there!

Hi

Welcome!

Write a message

(b) TOB-STT being accessed from a smartphone

Fig. 1. TOB-STT interface.

3. Conversational Agents in CS Education

The exploration of a conversational agent in the computing domain, more specifically in educational practices in computing, is not new. Previous studies have shown the efforts of researchers towards proposing educational solutions based on conversational agents to problems in teaching computers, formal education, and training of professionals in the field. Conversational agents have been established for different contexts, environments, and Computing disciplines. This section discusses some studies similar to the present research.

Katchapakirin and Anutariya (2018) reported the need for a conversational agent when Thailand adopted computer content in basic education and high school. Managers recognized the lack of qualified human resources in the area of computing (especially in block-based visual programming language Scratch[3]) to teach certain content to students. Towards mitigating the problem, ScratchThAI, a conversational agent that supports students through textual conversations, was developed. The aim was to stimulate the development of computational thinking skills.

Ocaña *et al.* (2019) also focused on a conversational agent that could support students who were not studying computing in higher education, but were subjected to initiatives associated with a programming learning at basic levels of education. The agent was treated as a learning companion in an environment designed for children to learn how to program; it asked them to write their programs and, if necessary, perform debugging. It would enable students to practice programming concepts with real programs and immediate feedback.

Among studies focused on conversational agents for the teaching of computer networks, Herpich *et al.* (2016) introduced ELAI, a conversational agent available in a virtual world that simulates a laboratory dedicated to teaching computer networks. It is represented through an NPC (Non-player character) and can solve students' doubts on the contents taught.

Leonhardt *et al.* (2007) designed a conversational agent as a tool for training computer network management, which served as a support for professionals with little experience. According to the authors, such professionals tend to show limited understanding of details of management protocols, and the agent, designed to explain concepts of the computer network management domain would teach them how to obtain any given information.

In the scope of software engineering education, Paschoal *et al.* (2018) developed a prototype of a conversational agent for supporting students in developing skills associated with extracting software requirements through an interview technique. The agent assumes the role of a stakeholder, and the student interacts with it towards extracting the requirements and improving skills. Additionally, the agent has knowledge of software engineering concepts, which allows students towards their doubts.

Mikic-Fonte *et al.* (2018) developed a conversational agent that answers students' questions in a computer architecture course so that they would acquire the habit of locating information autonomously, with no direct interaction with the teacher. According to the authors, the agent would free the teacher from answering questions usually asked every semester/year. In such a course, most students' doubts tend to be on the use of assembly language emulators (e.g., "How can I install the emulator on my Linux sys-

[3] More information available at `http://bit.ly/3qd28oJ`

tem?”). The authors’ intention was to use the agent in a course on a MOOC platform maintained by the university.

Aguirre *et al.* (2018) reported the planning of conversational agent to solve difficulties faced by students while learning Java language and the object-oriented paradigm. The conversational agent is described as an additional resource to the MOOC, which uses natural spoken language. According to the authors, its main function is to suggest MOOC modules that require further student attention and explain concepts of the contents covered in the course.

4. Experimental Study

The experimental study addressed in this article was planned and conducted according to the experimental process recommended by Wohlin *et al.* (2012). Therefore, tasks for identification and scope definition were followed for planning, operation, and execution. Each task is discussed in this section.

4.1. *Scope*

Our experiment studied whether the conversational agent could provide effective educational support to students who needed help. The effect to be observed was associated with both availability and assistance of the agent in educational activities on software testing. Therefore, we analyzed the efficacy of students in undertaking an activity on software testing concepts with the help of the agent. The analysis was based on the following research question: ***Can TOB-STT support software testing students in solving their content-related questions while performing their activities?*** The following hypotheses were then generated from the question:

- **Null hypothesis:** There is no difference between the efficacy of students in undertaking an educational activity supported or not by the TOB-STT.
- **Alternative hypothesis:** There is a difference between the efficacy of students in undertaking an educational activity supported or not by the TOB-STT.

After the research question had been raised, the goals of the experiment were specified in the following five parts, as a paradigm similar to the GQM (Goal-Question-Metric) (Basili *et al.*, 1994): ***analyze*** TOB-STT, ***for the purpose of*** verifying its efficacy in supporting students, ***with respect to*** students’ efficacy in undertaking educational activities, ***from the point of view*** of the researchers, ***in the context of*** computer science graduate students studying software testing.

The experimental study was conducted in two Brazilian educational institutions, namely Federal University of Grande Dourados (UFGD) and Fluminense Federal University (UFF), in the context of Verification, Validation and Software Testing (VV&T) and Quality and Testing (QT) courses, both part of undergraduate programs in Information Systems. In the period of the experiment study, the course offered by UFGD had 21 students, and the course at UFF had 17.

4.2. *Planning*

After the planning of the experiment discussed in this section, the variables and the necessary instruments for its development were established.

Selection of variables

In the scientific method of experimentation, variables are used as a mechanism to measure the relationship between cause and effect. According to Wohlin *et al.* (2012), they are called independent and dependent variables. The former present the cause that affects the result of an experimental process, whereas the latter shows the effect produced. According to this understanding, the following independent variables were considered in the experimental study:

Participants' knowledge, which refers to students' previous knowledge of software testing content. Undergraduate students had already attended classes on software testing concepts and terminologies approximately eight weeks before the experiment.

Educational activity, which consists of the activity on software testing undertaken by the students.

Teaching support mechanism (factor), which represents the initiative that guides the setup of the treatments adopted in an experiment. The following two were considered:

- **Treatment A:** the conversational agent is at the students' disposal to solve their doubts while undertaking the educational activity.
- **Treatment B:** the conversational agent is not at the students' disposal; therefore, they undertake the educational activity with no support to resolve doubts.

The following dependent variables were considered:

Efficacy, which portrays the efficacy of students in recognizing the defects injected in an activity on concepts, terminologies, and definitions in the software testing area. Below is the formula that measured efficacy:

$$E_{(\mathrm{i})} = \frac{n_{(\mathrm{i})}}{TOTAL}$$

where

E = represents the value assigned to efficacy,

i = denotes the student, i.e., $\{student_1, student_2 \ldots student_z\}$,

n = is the number of defects correctly identified, i.e., predicted in the test oracle[4] and were recognized by the students, and

total = denotes the number of defects predicted by the test oracle.

[4] A software artifact that helps you decide whether or not the software test output was a success.

Instrumentation description

The experiment required the preparation and recovery of some materials, which assumed different functionalities (e.g., helping the teacher to instruct students participating and collecting data to be further used in the analyses).

The educational activity was established after the description of the instruction material. It consisted of the analysis of an argument of a test team[5] on the importance of the activity within a software organization and its prioritization in a software development process. The argument was a text addressed to the team that managed the organization asking for a document that explained the functioning of the test and why it should be prioritized. The students assumed the role of testers and analyzed the argument, listing the defects identified in a table. The argument involved concepts, definitions of test subjects, and examples of tests conducted in source code.

TOB-STT is part of the study. We use the available version organized by the authors (Paschoal *et al.*, 2019) for web access. Therefore, no server preparation was necessary to make the agent available.

A tutorial designed by the authors (Paschoal *et al.*, 2019) supported the development of activity and introduction of the conversational agent to the students, clarifying its functionalities. Examples of interactions and a link to access the agent were also provided. An informed consent form was created to explain the experiment to the students regarding the use of the data and the abandonment of the study at any time with no losses.

Finally, a form for collecting the student's perception of the conversational agent was prepared. The instrument was based on the study by Herpich *et al.* (2016) and consisted of nine statements based on the five-point Likert scale, which ranges from strongly disagree to strongly agree. Such statements alternate between negative and positive for controlling participants' hypothetical tendency to agree with them (Lewis, 2018). The assertions are listed in what follows:

1. The interaction with TOB-STT for the first time was not encouraging.
2. The answers provided by TOB-STT were related on the topic questioned.
3. TOB-STT was unable to answer the questions asked.
4. By using TOB-STT, I could acquire the knowledge I wanted.
5. TOB-STT did not provide reliable information in its responses.
6. TOB-STT contributed to the accomplishment of the task.
7. TOB-STT provided answers slowly.
8. TOB-STT has an easy-to-use interface.
9. I was dissatisfied with TOB-STT.

4.3. *Operation and Execution*

After the definition of the instruments, the experiment execution was prepared and performed.

[5] A test team is a group of individuals who work in the software testing activity of a software organization.

Preparation

During the semester of the planning stage, the professors taught subjects that involved software testing at their respective universities (i.e., UFGD and UFF). The course curricula were similar, and both were offered to students of the final year of the undergraduate program. The context was selected according to the adequacy, and the samples used in the experimental study were selected for convenience. The number of necessary classes was established in function of the schedule of each course so that the experiment would not hinder its progress.

Since the contents are part of the course, no special tactics were necessary; a class on testing fundamentals and another on functional testing would be taught normally. A third class on the activity to be developed after eight weeks was offered. The activity would take place after students have acquired knowledge on the subject, and was planned to be taken at the time of the class. However, it was not mandatory for students, and it would be conducted in a way the results would not interfere with their grades. It aimed at encouraging students' participation in the class as an opportunity for them to review the content and observe aspects to be fixed.

A form with the nine statements was prepared on Google[6] for collecting students' feedback. It would be available for those who would participate in the experimental group after the activity, which was organized in a way to be accessed through computers in the computer labs of each educational institution. It would also facilitate the execution, especially because the students of the experimental group would undertake the activity while interacting with TOB-STT.

Finally, a learning management system was prepared to facilitate receiving students' submissions, i.e., a list of defects found in the argument on the importance of prioritizing software testing. The environment was essential for the experimental group since the students should provide feedback on their perceptions after using the conversational agent.

Execution

The experimental study was conducted in three classes at each educational institution. This section describes the activities developed during the classes.

- **Class 1:** In the first meeting, the professor of each course presented the fundamentals of software testing, the objectives of the test, the process, terminologies, test cases (i.e., what they are, their constitutive parts, and order of execution: cascade test cases, independent test cases), testing techniques (what they are, their aim, specification-based testing, implementation-based testing), test criteria, test steps (planning, design, execution, analysis), and test phases (unit, integration, system). As each subject was addressed, examples were given and doubts were resolved. Eventually, an exercise was performed for fixing the content, and doubts arisen were resolved.

[6] More information available at: `http://bit.ly/3nEc212`

- **Class 2:** In the second meeting, the professor of each course explained the functional testing technique, also known as a black-box based on specifications and documentation of the software to derive test requirements. The steps for applying functional test criteria were discussed, and the most well-known ones were presented, with emphasis on equivalence partitioning and boundary value analysis. Examples were provided as each criterion was taught step-by-step. As in the first meeting, the students eventually developed an activity towards exercising the criteria learned. The professors resolved the arisen doubts.
- **Class 3 (experimental group):** In the third meeting, the professor of the Verification, Validation, and Software Testing course at UFGD taught the class in a computer lab, which had a computer for each student with access to the Internet. The professor explained the educational activity and made it, together with the informed consent form and the links to the conversational agent and the form with the statements about the perception of use available in a learning management system. Students were informed that the activity would be individual, not evaluative, but only to fix the content. They should read it and interact with TOB-STT, considered a tutor -specialist in the domain, and use no other material for consultation. They were given 60 minutes to complete the activity, and should eventually send the table with the defects identified and give feedback through the available form.
- **Class 3 (group control):** In the third meeting, the professor of the Quality and Testing course at the UFF developed the activity in the laboratory, similarly to the UFGD professor. Each student was provided with a computer with access to the learning management system in which the educational activity was available and an option to send the table with the defects identified. They were also informed the activity was individual and no support mechanism should be used, including consultation with teaching materials. They were given 60 minutes the perform the activity and send files to the professor.

After the experiment, the collected data were analyzed, as addressed in the next section.

5. Results

This section is devoted to the analyses performed with the obtained data. It describes the decisions made during the analyses and then analyses the efficacy of the conversational agent. Finally, it discusses the students' perceptions who interacted with the TOB-STT.

5.1. *Efficacy Recognition*

The defects identified and listed by the students in both control and experimental groups were analyzed, following the test oracle. The efficacy value was then calculated for each participant of the experiment, and different analyses were performed on the data collected. In particular, descriptive analyses and statistical inference were used.

Boxplots were used in the descriptive analysis to recognize outliers[7] in each variable. Since the sample was relatively small (i.e., less than 30 per sample group), the outliers were considered in all analyses.

Finally, the Shapiro-Wilk test checked the normality of the data in the inferential statistics. Since the experiment considered one factor and two treatments, Student's t-test was applied to data that followed a normal distribution, and the Mann-Whitney test was used for those that did not, during the planning of the experiment analysis.

5.2. *Efficacy Analysis*

Regarding efficacy, the performance of both the experimental group and the control group was not expected by the researchers, since efficacy values greater than 50% were expected, indicating students would identify at least 50% of the defects injected in the educational activity. However, as shown in Table 2, the average of the recognized defects was 27% in both groups.

By directing attention to the median, we found that the groups were able to obtain a median equal to 30%. This means that 50% of the efficacy values were less than 30%. The sample standard deviation revealed that the variation in the number of the identified defects in the control group was equivalent to the experimental group.

The box plots in Fig. 2 support the graphical analysis of students' efficacy in identifying defects in each group. Students who did not use the conversational agent to identify defects obtained a greater variability than those supported by TOB-STT. Therefore, the agent may be contributed to the less oscillation in variability on the efficacy of the experimental group.

Finally, inference tests checked if the efficacy of the students undertaking the educational activity on software testing with the support by TOB-STT was different from that with no support. The normality of the data was analyzed according to a 95% confidence level. Shapiro-Wilk test revealed (i) the efficacy data for the experimental group did not follow a normal distribution, since the p-value resulted in 0.0000028, and (ii) the efficacy data for the control group did not follow a normal distribution – its p-value was 0.0000093. Therefore, the Mann-Whitney test was applied and provided a 0.484352 p-value, and the null hypothesis was not rejected.

Table 2
Descriptive statistics of the efficacy values

	Group	
	Control	Experimental
Mean	27%	27%
Median	30%	30%
Standard deviation	0.13	0.13

[7] Outliers represent atypical values from variations in the sample groups.

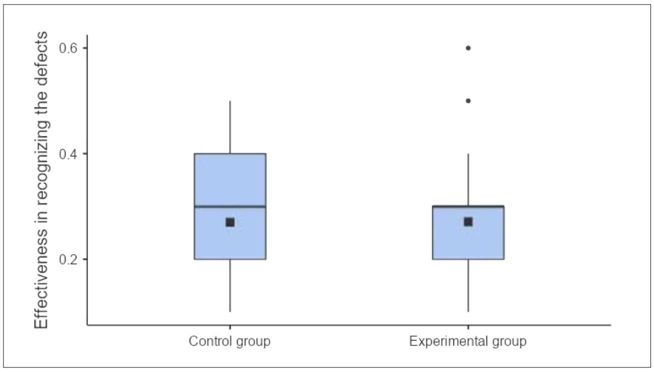


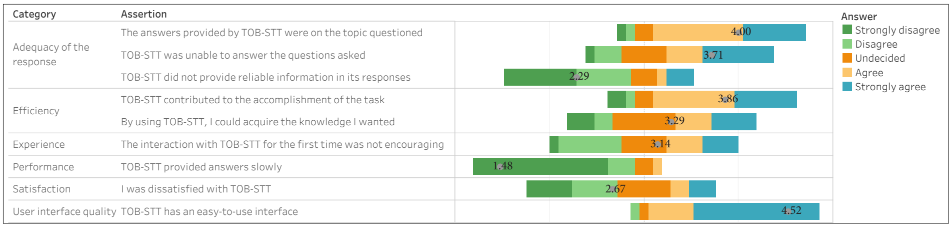
Fig. 2. Comparison of the efficacy values in the identification of defects.

The study shows that students with and without the support of TOB-STT achieved the same efficacy in the activity, indicating this support was not as effective as expected. The efficacy of the educational support may be related to different aspects, and we believe the students' perception of the agent can offer shreds of evidence that contribute to understanding the reasons why TOB-STT was not able to increase the efficacy of the student in the identification of defects during the educational activity.

5.3. *Feedback from Participants*

An exploratory analysis was conducted with data on the students' perceptions who used TOB-STT (see Fig. 3). The data set shows that the students mostly agreed on the ease of the interface use, whereas that of higher disagreement is related to the time TOB-STT takes to provide its answers. In this case, since the statement was negative, we understand the students were satisfied with the agent's performance in offering answers.

Towards a better understanding of the results of the analysis of efficacy, the students' perceptions were organized into six categories, namely adequacy of the response, performance, efficiency, experience, user interface quality, and satisfaction. The former grouped more than one statement and addresses aspects of students' perception of the quality of the interaction promoted by TOB-STT. According to the category adequacy of the response of Fig. 3, most students claimed (i) the answers provided by TOB-STT was



Category	Assertion	
Adequacy of the response	The answers provided by TOB-STT were on the topic questioned	4.00
	TOB-STT was unable to answer the questions asked	3.71
	TOB-STT did not provide reliable information in its responses	2.29
Efficiency	TOB-STT contributed to the accomplishment of the task	3.86
	By using TOB-STT, I could acquire the knowledge I wanted	3.29
Experience	The interaction with TOB-STT for the first time was not encouraging	3.14
Performance	TOB-STT provided answers slowly	1.48
Satisfaction	I was dissatisfied with TOB-STT	2.67
User interface quality	TOB-STT has an easy-to-use interface	4.52

Fig. 3. Students' perceptions of the TOB-STT.

following the subject of the question; (ii) the information provided by the TOB-STT is reliable; and (iii) the TOB-STT was unable to answer some questions.

As some students indicated that the agent was unable to answer some of their doubts, we believe that this may have contributed to the result observed in the efficacy analysis. Students may have asked some specific questions about a software testing concept, and the agent was unable to help them. Another possible interpretation is they may have asked questions not related to the domain.

In a given statement, the students informed whether they believed TOB-STT had contributed to the performance of the activity. As shown in the efficiency category of Fig. 3, most answers corroborate the hypothesis raised at the beginning of the study, i.e., the conversational agent offers support to the activity. This result shows the additional support provided confidence to the students, i.e., they were convinced the agent would help them. However, students were undecided whether the agent was able to contribute to increasing knowledge about software testing.

We try to understand the students' experience when interacting with the conversational agent during the educational activity. In particular, students indicated whether the experience was not encouraging. The experience category in Fig. 3 shows the Likert scale offered no option that attracted special attention from the students. Therefore, we believed that some students were excited by the experience of using the TOB-STT, while others were not.

The last statement addressed student's satisfaction with TOB-STT. Similarly to the assertion related to the experience, the data show some students were satisfied, whereas others were not very pleased (see the satisfaction category in Fig. 3).

The results provided some evidence that contributes to the analysis of the efficacy of the educational support. Those that drew the most attention refer to the agent not being able to offer answers to some doubts. However, the students believed it helped them in completing the activity. As the results of efficacy, presented in Section V-B, indicated that the support offered by the conversational agent did not promote greater efficacy in the identification of defects by software test students, the feeling of support was manifested in the experimental session.

6. Threats to Validity

Although this study adopted an experimental process, it is not free from threats to validity, which were mitigated by some initiatives undertaken during the planning and conducting. The main threats and actions are presented in that follows.

- **Reliability of measures:** efficacy was used as a metric to check whether the student who has access to TOB-STT can identify the defects more accurately than the student who does not. It has been used in experimental studies on software testing education (de Jesus *et al.*, 2020; Paschoal *et al.*, 2020). To mitigate any threats to validity, the test oracle was established by a professor in the software testing course with more than 20 years of experience. In addition to this, the comparison among the defects identified by the students with what was expected by the test oracle was made by a single person who has worked in research on software testing education. Therefore, a poorly formulated form was avoided, since it might jeopardize the viability of the research.

- **Single method bias:** the efficacy of the educational support offered by TOB-STT was analyzed through an educational activity. Ideally, it is believed efficacy is observed throughout the development of a larger set of activities, thus providing a context closer to a learning environment. Students can do better in one activity than in another because of their affinities, preferences, etc. However, more classes would be necessary for a larger set of activities, compromising the schedule of the courses. The performance of a single educational activity has been adopted in an experimental study of the theme (Paschoal *et al.*, 2020) towards not interfering with the teaching of other subjects and the courses' calendar.
- **Generalization of results:** the results from an experimental process are expected to be generalizable. In this sense, the choice of the participants may make generalization unfeasible, since they provide the data used during analyses. Students were selected for our experiment because they were enrolled in software testing subjects at two educational institutions, which makes the sampling heterogeneous. The threat was mitigated through the selection of the academic context.
- **Risk of error in the analysis:** in some experimental studies, tests may not indicate the relationship between cause and effect, although it may exist. Therefore, specific care must be devoted to both treatments and analyzed data. The threat was evidenced in our experiment and it was mitigated by a replication of the experiment with Software Testing and Inspection students at the University of Sao Paulo (USP). The students were selected by convenience, were in the last year of their undergraduate programs, had received training, and undertook the educational activity with the support of TOB-STT after eight weeks. The data were collected and analyzed, and eventually, the Mann-Whitney test again shows no statistical difference in the efficacy of the students supported by TOB– STT (USP students) with no support (UFF students), since the p-value was 0.959729.

7. Discussion

In this section, we discuss some interesting results of the analyses presented in Section 5. These results were obtained through the experimental study execution on TOB-STT conversational agent and relating them to the findings of other researchers in the area to glimpse a possible explanation to the results obtained. A key point is a failure of rejecting the null hypothesis (i.e., there is no difference between the efficacy of students in undertaking an educational activity supported or not by the TOB-STT). To find evidence of the causes that influenced these results, we revisited the students' feedback about their perceptions on the TOB-STT (see Section 5.3). In a detailed analysis of the feedback results, the third assertion draws attention since most experimental study participants reported that the TOB-STT fails to correctly answer the participants' questions. According to Quiroga Perez *et al.* (2020) and Molnár and Szüts (2018), one of the problems that can affect conversational agents is student frustration when they cannot get the answers to what they are looking for, in which case the conversation does not flow as expected. Thus, the agent is unable to give effective learning support.

To obtain evidence to justify this perception, we retrieved the logs of interactions produced between the students and TOB-STT towards evaluating whether the students have received the answers expected during the interaction. The interactions records analysis to evaluate the usefulness of the responses generated by the TOB-STT to software testing students was based on the strategy defined in AbuShawar and Atwell (2016). During the evaluation, the quality of the responses was measured by observing and classifying the conversational pairs[8] produced throughout an interaction. The classification of the answer into a specific category is based on the analysis of the conversational pairs. The categories adopted in this study are presented as follows.

- **Correct answer:** the conversational agent correctly understood the question issued by the student answering in the right way for the interaction. For example, the student asked: "what is a bug?" and the conversational agent correctly explained that a bug is a divergence of a code specification from what was developed.
- **Incorrect answer:** the conversational agent did not correctly understand the question made by the student and so could not decide the correct answer, responding inappropriately to the interaction. For example, the conversational agent answers "what is a functional testing technique" when the student asks: "what is a bug?". This type of answer is inappropriate because the conversational agent failed to capture the statement and offered an incorrect answer. The incorrect answer can be divided into partially related and unrelated.
    - **Partially related:** in this case, the answer provided by the conversational agent is not correct, but it is about the same subject. For example, the student asks the conversational agent what a defect in software is, and the agent answers what a software failure is.
    - **Unrelated:** in this case, the answer issued by the conversational agent is not correct and is about a different subject. For example, the student asks the conversational agent what is a defect in the software, and the agent, rather than answering the student, will ask the student: "How are you?".
- **No Answer:** the conversational agent was unable to answer the question made by the student and consequently left the student unanswered or issued a warning that it did not know how to answer the student's question. For example, the student asks the conversational agent "What is a bug?" and TOB-STT replies that it cannot answer.

Whereas the strategy proposed by AbuShawar and Atwell (2016) based on a subjective metric, we invited a software testing expert who did not take part in experiment execution to classify the interaction between students and TOB-STT. We included an expert who did not participate during the experiment execution to avoid bias in the interactions classification. The expert has five years of experience in education and researching software test-

[8] In this study, conversational pairs represent the questions asked by the student and the response issued by the conversational agent to that question.

ing, shown by publications in conferences and journals of Computer Science Education and Software Testing, Verification, and Validation. The expert analyzed 619 interactions between the TOB-STT and the 21 students in the experimental group and classified them according to the proposed method. From this classification, we analyze the results.

In the first moment, we checked if all 21 students interacted with the TOB-STT. We found that all students talked to the TOB-STT, asking questions pertinent to the educational activity subject that they were doing. Fig. 4 presents the box plot with the number of interactions made by the participants. We noticed that there were no outliers, and thus there were no discrepant values of interaction. Additionally, we observed that the TOB-STT interacted on average 29 times with each student (value represented by the black square in the box plot). The median value, represented by the line in the box plot, indicates that 50% of the participants interacted more than 27 times with the conversational agent. The other 50% of the participants interacted less than 27 times. Through this analysis, it was possible to obtain evidence that the students interacted adequately with the TOB-STT to use it as a support to perform the activity. Based on results, some students interacted more than others, but the number of interactions per student remains normal from a statistical point of view (Fig. 4).

We analyzed the classification made by the expert. Fig. 5 shows that the conversational agent correctly answered the students' doubts and questionings in 39.74% of the interactions made by the students. However, in 42.33% of interactions, the students did not receive a response from the TOB-STT. This last result is disturbing, given that the students failed to get the answers expected from the conversational agent, and the agent failed to support in solving the activity. Furthermore, in 17.84% of the interactions, the student received an incorrect response from the conversational agent, which indicates that the agent is unable to understand the students' interactions. In this case, the agent offered an answer that may have confused and increased the difficulty of solving the educational activity.

According to Hobert (2019), an important factor in successful learning by conversational agents is natural language understanding and response generation. In the case that an agent cannot understand the student's intent, it cannot generate the appropriate responses. Consequently, the students tend to reconsider using the conversational agent.

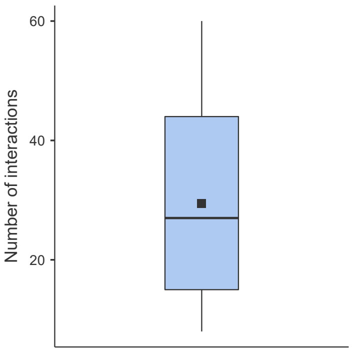


Fig. 4. Number of interactions made by the students.

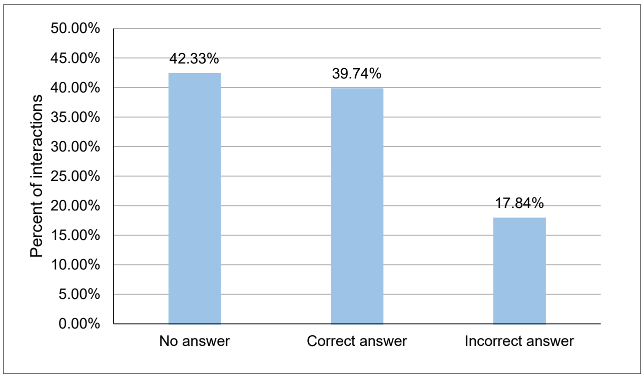


Fig. 5. Proportion by type of answer classified.

The log analysis results suggest that the student's performance in the activity may have influenced the omission or inconsistency of the answers offered by the agent. As a result, the agent failed in providing educational support to the student during software testing educational activity. It is important to reiterate that they were instructed during the experiment that the TOB-STT is an expert in software testing, so the students assumed that the answers offered by the conversational agent were correct. That means that the students may have harmed themselves while performing the activity by relying on the responses provided by the conversational agent.

The last analysis on the usefulness of the responses generated by the TOB-STT corresponded to the classification of the incorrect answers into the classes partially related and unrelated. According to Fig. 6, most of the incorrect responses from the TOB-STT were about a different subject than the one requested by the learners in the interaction.

The analysis from the interactions points out that the TOB-STT has some difficulties interacting with the student, failing to answer the student on the subject and answering some questions incorrectly. That means that the conversational agent needs improvement. One way to improve the conversational agent is to extend its knowledge

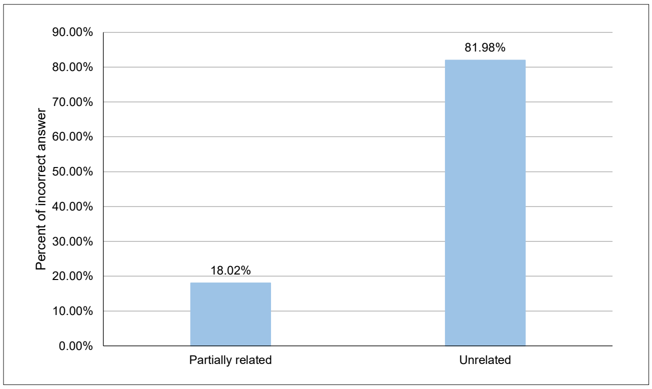


Fig. 6. Incorrect answers classification.

base by creating new conversational pairs in the agent's database (Sandoval, 2018). Another way is to inspect the inference model used by the conversational agent during the search for answers to interactions. This inspection should find if there are any problems in the pattern matching algorithm implementation. The pattern matching algorithm used in TOB-STT may contain defects. Even more drastically, a design technique that performs natural language processing more accurately can be the solution for the conversational agent to improve its interaction and increase the percentage of correct answers (e.g., a model based on machine learning (Hiremath *et al.*, 2018)).

8. Limitations of the Study

Although the study was conducted systematically, with planning and mitigation of threats to validity, it has some limitations. The experimental group was comprised of students from the same university who might have been less effective without the help of the conversational agent. Since the study did not collect additional data for checking this assumption, the following research question remains open: "Would the efficacy of the students who interacted with TOB-STT be lower than that with no support?". The experimental session was replicated at USP and revealed the agent was unable to increase the efficacy of the results; however, the student's preliminary knowledge was not investigated, which may lead to the emergence of new theories.

A way to address those limitations would be to experiment with a different design. Instead of dividing the students into groups (control and experimental) based on their home institution, each course could be randomly divided – the groups would undertake the educational activity with and without the conversational agent, respectively. However, the division would require different spaces (two computer labs at each university) and at least one more professor for the activities. While one group would undertake the activity with no conversational agent, the other would be in another laboratory participating with the support of the agent. Another alternative could be through a paireddesign experimental (Wohlin *et al.*, 2012).

9. Concluding Remarks

TOB-STT conversational agent was designed for supporting students who have no access to professors. It was first described by the authors (Paschoal *et al.*, 2019), who evaluated the agent's knowledge of the software testing domain. We believe that TOB-STT can be introduced in a software testing MOOC; however, the efficacy of educational support offered by TOB-STT has not been investigated. We conducted a study to understand the impact of TOB-STT when it is used as a mechanism to support teaching in a scenario where the student does not have the support of the professor. A controlled experiment was planned and developed; some software testing students undertook an educational activity supported by the TOB-STT, while others students performed the same activity without it. The experimental sessions demonstrated the support offered did not significantly impact the performance of the software testing activity. Based on experimental results, we analyzed the interaction logs. We found that the TOB-STT did not provide adequate responses to most students' questions. We believe that the efficacy was affected by the omission and inconsistency of the answers offered by the conversational agent during the educational activity. Improvements must be made on the conversational agent before being incorporated into some software testing MOOC. Among the possible improvements to be implemented in TOB-STT, we can highlight: (i) extension of the conversational agent's knowledge base; (ii) training the model that represents knowledge with new examples of interactions; (iii) improve the design technique that characterizes the linguistic treatments performed by the conversational agent; among others.

Acknowledgments

The authors acknowledge CAPES – Financing Code 001, FAPESP (Process 2020/051912), CNPq (Process 314174/2020-6 and Process 312922/2018-3), and PROPP/UFGD (Sig-Proj protocol: 329332.1866.8276.06042019) for their financial support.

References

AbuShawar, B., Atwell, E. (2016). Usefulness, localizability, humanness, and language-benefit: additional evaluation criteria for natural language dialogue systems. *International Journal of Speech Technology*, 19(2), 373–383.

ACM (2016). Curricula Recommendations. Accessed on January 10, 2021.

Aguirre, C.C., Kloos, C.D., Alario-Hoyos, C., Muñoz-Merino, P.J. (2018). Supporting a MOOC through a conversational agent. Design of a first prototype. In: *International Symposium on Computers in Education*, pp. 1–6.

Basili, V.R., Caldiera, G., Rombach, H.D. (1994). The goal question metric approach. In: Marciniak, J.J. (Ed.), *Encyclopedia of Software Engineering* (Vol. 2) (1st ed.). John Wiley & Sons, Hoboken, New Jersey, pp. 528–232.

Benitti, F.B.V. (2018). A methodology to define learning objects granularity: a case study in software testing. *Informatics in Education*, 17(1), 1–20.

de Jesus, G.M., Ferrari, F.C., Paschoal, L.N., de Souza, S.R.S., de Paula Porto, D., Durelli, V.H.S. (2020). Is It Worth Using Gamification on Software Testing Education? An Extended Experience Report in the Context of Undergraduate Students. *Journal of Software Engineering*, 6, 1–19.

Dias-Neto, A.C., Matalonga, S., Solari, M., Robiolo, G., Travassos, G.H. (2017). Toward the characterization of software testing practices in South America: looking at Brazil and Uruguay. *Software Quality Journal*, 25(4), 1145–1183.

Enoiu, E.P. (2019). Teaching software testing to industrial practitioners using distance and Web-based learning. In: *International Workshop on Frontiers in Software Engineering Education*, pp. 73–87. Springer.

Fassbinder, A.G.O., Fassbinder, M., Barbosa, E.F., Magoulas, G.D. (2017). Massive open online courses in software engineering education. In: *IEEE Frontiers in Education Conference*, pp. 1–9.

Fraser, G., Gambi, A., Rojas, J.M. (2018). A preliminary report on gamifying a software testing course with the code defenders testing game. In: *European Conference of Software Engineering Education*, pp. 50–54.

Garousi, V., Zhi, J. (2013). A survey of software testing practices in Canada. *Journal of Systems and Software*, 86(5), 1354–1376.

Herpich, F., Nunes, F.B., Voss, G.B., Medina, R.D. (2016). Three-dimensional virtual environment and npc: a perspective about intelligent agents ubiquitous. In: *Handbook of Research on 3-D Virtual Environments and Hypermedia for Ubiquitous Learning*. IGI Global, ???, pp. 510–536.

Hiremath, G., Hajare, A., Bhosale, P., Nanaware, R., Wagh, K. (2018). Chatbot for education system. *International Journal of Advance Research, Ideas and Innovations in Technology*, 4(3), 37–43.

Hobert, S. (2019). How are you, chatbot? Evaluating chatbots in educational settings. *Lecture Notes in Informatics*, 17, 259–270.

IEEE (2013). *Computing Curriculum Efforts*. Accessed on January 10, 201.

Katchapakirin, K., Anutariya, C. (2018). An architectural design of scratchthai: A conversational agent for computational thinking development using scratch. In: *International Conference on Advances in Information Technology*, pp. 1–7.

Lemos, O.A.L., Silveira, F.F., Ferrari, F.C., Garcia, A. (2017). The impact of Software Testing education on code reliability: An empirical assessment. *Journal of Systems and Software*.

Leonhardt, M.D., Tarouco, L., Vicari, R.M., Santos, E.R., da Silva, M.S. (2007). Using chatbots for network management training through problem-based oriented education. In: *IEEE International Conference on Advanced Learning Technologies*, pp. 845–847.

Lewis, J.R. (2018). The system usability scale: past, present, and future. *International Journal of Human–Computer Interaction*, 34(7), 577–590.

Melo, S.M., Moreira, V.X.S., Paschoal, L.N., Souza, S.R.S. (2020). Testing Education: A Survey on a Global Scale. In: *Brazilian Symposium on Software Engineering*, pp. 554–563.

Mikic-Fonte, F.A., Llamas-Nistal, M., Caeiro-Rodríguez, M. (2018). Using a Chatterbot as a FAQ Assistant in a Course about Computers Architecture. In: *IEEE Frontiers in Education Conference*, pp. 1–4.

Mittal, A., Vigentini, L., Djatmiko, M., Prusty, G., Sharma, Y., King, M.E. (2018). Mooc-o-bot: using cognitive technologies to extend knowledge support in moocs. In: *IEEE International Conference on Teaching, Assessment, and Learning for Engineering*, pp. 69–76.

Molnár, G., Szüts, Z. (2018). The role of chatbots in formal education. In: *2018 IEEE 16th International Symposium on Intelligent Systems and Informatics (SISY)*, pp. 000197–000202. `https://doi.org/10.1109/SISY.2018.8524609`

Montenegro, J.L.Z., Costa, C.A., Righi, R.R. (2019). Survey of conversational agents in health. *Expert Systems with Applications*, 129, 56–67.

Myers, G.J., Sandler, C., Badgett, T. (2011). *The Art of Software Testing*. John Wiley & Sons, ???.

Ng, S., Murnane, T., Reed, K., Grant, D., Chen, T.Y. (2004). A preliminary survey on software testing practices in Australia. In: *Australian Software Engineering Conference*, pp. 116–125. IEEE.

Ocaña, J.M., Morales-Urrutia, E.K., Pérez-Marín, D., Tamayo-Moreno, S. (2019). How to Create a Pedagogic Conversational Agent for Teaching Computer Science. In: *Advanced Online Education and Training Technologies*. IGI Global, ???, pp. 114–134.

Paschoal, L.N., Souza, S.R.S. (2018). A survey on software testing education in brazil. In: *Brazilian Symposium on Software Quality*, pp. 334–343.

Paschoal, L.N., Oliveira, M.M., Chicon, P.M.M. (2018). A chatterbot sensitive to student's context to help on software engineering education. In: *XLIV Latin American Computer Conference*, pp. 839–848. IEEE.

Paschoal, L.N., Turci, L.F., Conte, T.U., Souza, S.R.S. (2019). Towards a Conversational Agent to Support the Software Testing Education. In: *Brazilian Symposium on Software Engineering*, pp. 57–66.

Paschoal, L.N., Oliveira, M.M., Melo, S.M., Barbosa, E.F., Souza, S.R.S. (2020). Evaluating the impact of Software Testing Education through the Flipped Classroom Model in deriving Test Requirements. In: *Brazilian Symposium on Software Engineering*, pp. 570–579.

Prates, J.M., Garcia, R.E., Maldonado, J.C. (2018). MOOCs on the Context of Software Engineering Teaching and Training: Trends and Challenges. In: *IEEE Frontiers in Education Conference*, pp. 1–9.

Quiroga Perez, J., Daradoumis, T., Puig, J. (2020). Rediscovering the use of chatbots in education: A systematic literature review. *Computer Applications in Engineering Education*, 28. `https://doi.org/10.1002/cae.22326`

Sandoval, Z.V. (2018). Design and Implementation of a Chatbot in Online Higher Education Settings. *Issues in Information Systems*, 19(4), 44–52.

Tan, G. (2016). A Collection of Well-Known Software Failures. `http://www.cse.psu.edu/gxt29/bug/softwarebug.html`

Wallace, R.S. (2009). The anatomy of ALICE. In: *Parsing the Turing Test*. Springer, ???, pp. 181–210.

Wohlin, C., Runeson, P., Hst, M., Ohlsson, M.C., Regnell, B., Wessln, A. (2012). *Experimentation in Software Engineering*. Springer Publishing Company, Incorporated, ???.

Zhivich, M., Cunningham, R.K. (2009). The real cost of software errors. *IEEE Security Privacy*, 7(2), 87–90.

L.N. Paschoal received a bachelor’s degree in computer science from the University of Cruz Alta (UNICRUZ) in 2017 and a master’s degree in computer science and computational mathematics from the University of São Paulo (USP) in 2019, where he is currently pursuing the Ph.D. degree in computer science and computational mathematics with ICMC. His current research topics include conversational systems, software engineering education, and software testing education.

S.M. Melo is an Assistant Professor at the Faculty of Exact Sciences and Technology (FACET) Federal University of Grande Dourados (UFGD). She holds a Ph.D. degree in Computer Science and Computational Mathematics from the Institute of Mathematics and Computer Science (ICMC) University of São Paulo (USP), received in 2018. Her research interests focus on computing education, software testing, and experimental software engineering.

V.O. Neves received a Ph.D. in computer science from the University of São Paulo (USP), Brazil, in 2015. She is an Assistant Professor at the Institute of Computing of Fluminense Federal University (UFF). Her research focuses on testing complex systems, including robotic systems, system-of-systems, and microservices.

T.U. Conte received a Ph.D. in software engineering from the Federal University of Rio de Janeiro (UFRJ) in 2009. She is an Associate Professor with the Institute of Computing (IComp), Federal University of Amazonas (UFAM). Her research interests include the intersection between software engineering and human–computer interaction, software quality, human-centred computing, and empirical software engineering.

S.R.S de Souza received a joint M.Sc. and Ph.D. in computer science from the University of São Paulo (USP), Brazil, in 1996 and 2000, respectively. From 2010 to 2011, she was a Visiting Scientist with the University of Southampton, U.K. She has been an Associate Professor of software engineering with the Institute of Mathematics and Computer Science (ICMC), University of São Paulo (USP), since 2005. In the past, she was a Lecturer with the Department of Informatics, State University of Ponta Grossa (UEPG), Parana, Brazil, from 1991 to 2005. She researches software testing, software engineering experimentation, and software testing education.