A Chatterbot Sensitive to Student's Context to help on Software Engineering Education

Leo Natan Paschoal
Institute of Mathematics and Computer Science
University of São Paulo
São Carlos, São Paulo, Brazil
leonatanpaschoal@gmail.com

Myke Morais de Oliveira
Center of Computational Science
Federal University of Rio Grande
Rio Grande, Brazil
myke.oliveira09@gmail.com

Patricia M. Mozzaquatro Chicon
Human and Social Sciences Center
University of Cruz Alta
Cruz Alta, Brazil
patriciamozzaquatro@gmail.com

***Abstract*—Requirements extraction is an important element of the software development process. One of the most used techniques for requirements extraction is the interview. Initiatives to support the training and technical training of computing students in this area are being proposed, such as the development of support mechanisms. These initiatives are proposed by the fact that computing students are graduating with limited practical knowledge in requirements extraction. In parallel, chatterbots have been investigated as tools with the capacity to support the training of students from different areas of knowledge, since the main characteristic is verbal conversational behavior. In medicine, for example, they can take on the role of a sick patient to train students to extract information about the patient's symptoms. One subject that has been explored in the context of educational chatterbots is context awareness, so that the chatterbot can present the right information for the right user. These surveys start from the premise that not every student has the same knowledge as their peers on the subject. Thus, in this research work in full paper we describe a chatterbot that offers support to Software Engineering Education, focusing mainly on the requirements extraction, which assumes the role of a stakeholder. A prototype of a chatterbot that is sensitive to student's context is presented, as well as preliminary results on the impact of this support mechanism in Software Engineering Education.**

***Index Terms*—Conversational Systems, Requirements Engineering Education, Software Development Process, Ubiquitous Learning.**

I. Introduction

The development of software is done by means of a sequence of steps which constitute a process. There are several models that support the development of software, but regardless of the software development process model used, one step is the requirements extraction. This is the stage where the computing professional will understand what the customer needs, how the software system will behave, what its features and characteristics [26]. To conduct the requirements extraction, several techniques have been developed since the emergence of Software Engineering. These techniques support the professional and seek to reduce the difficulties related to this stage. Among the existing techniques, one of the most traditional is the interview [22]. In this case, the professional will have to conduct the activity to make questions to the client, in order to understand what the client needs. When applying the technique, the professional must ask appropriate questions, so that the customer can answer them without the need of technical computing skills, seeing that not all stakeholder will be in the computing area.

After extracting requirements, this professional will write the requirements document. If he makes a mistake while completing the requirements document, the software will be subjected to contain faults, which will affect the quality of the document and consequently the software. Fault may induce the software to commit errors, such as an unexpected return, which may cause a failure[14]. It is known that a faulty software system can be critical [29], generate cost [4] and reduce user confidence [25]. In this sense, the activity of extracting requirements is important and has a direct relation with the software quality. Faced with this, the Software Engineering teaching becomes indispensable in the computing courses curriculum, where the requirements extraction is one of the contents to be taught. Although requirements extraction is taught, studies mention that students graduate and enter into job market with little or no experience in the role of requirements analyst [11]. Given this fact, the academic environment as responsible for the training of these subjects, is committed to offer subsidies to improve the quality of training.

Studies have been conducted to support the Software Engineering teaching and requirements extraction. In the specific requirements extraction context, educational games have been developed [17]. In some of these, students are submerged in a simulated day-to-day reality of a Software Engineer and will have to conduct some of the activities to extract requirements. Other types of resources are also used in order to train the student to face the reality of the profession of a Software Engineer, more specifically as requirements analyst/engineer. One tool to support student training, which can provide guidelines, interacting with the user in natural language that can be used in this area is a chatterbot. Chatterbots are programs that have the ability to talk to the user in natural language, simulating a human being, so the user believes that he is interacting with another person [8], [27].

Chatterbots have been used in different areas of human knowledge, such as in the area of Health [3] and Computing [2]. In the context of health teaching, for example, chatterbots can take on the roles of sick patients, the student has to interview the chatterbot and try to identify which disease it

may have, which exams he/she needs to request for and/or even medicines he/she will need to prescribe [16]. In the area of Computer Networking teaching, for example, chatterbot can be used to support the student or professional in training, as in handling and checking existing problems on the network [15].

A study front has been conducted aiming to incorporate features in these systems, so that the chatterbot is aware of the user context [13]. In this regard, the chatterbot knows the user's location, who is with him, what is the user's previous knowledge about the subject, among other informations. [19], [20], [18] when reporting their work, mention that these chatterbots are moving towards Ubiquitous Computing, because they present system characteristics with a certain level of ubiquity. In the work of [12], the authors present ELAI, a chatterbot to support the Computer Networks teaching that considers the student's expertise on the subject of the discipline by assisting him in moments of doubt. To do this, before the student has access to chatterbot, he must answer a test about the subjects that are taught in the discipline.

Taking into account the context presented, we believe that a chatterbot to support the development of technical skills of requirements extraction can be developed. Being aware that students with different knowledge about Software Engineering can make use of it, we attribute some characteristics to the chatterbot, so that it has capacity to know what the students prior knowledge about Software Engineering, since the more knowledge about Software Engineering the greater its chances of students having a larger domain on the requirement extraction technique. Thus, if the student has a higher level of expertise on the subject than its peers, it may be frustrated by using a chatterbot that has a facilitator knowledge base for training.

Therefore, we developed a chatterbot to support the Software Engineering teaching, focusing mainly on requirements extraction training with the interview technique, which has the ability to become aware of the student's previous knowledge about the content and that keep up with the student' performance in the learning of a Software Engineering discipline. In this article, we present the design and development of a chatterbot prototype to support Software Engineering teaching, with different knowledge bases that are used to support students with different levels of expertise. A pilot study was conducted to investigate the impact on students' learning of Software Engineering. Subsequently, an experimental study was conducted to investigate the instructional efficiency of the chatterbot sensitive to student's context.

This article is structured as follows. Section 2 presents the developed chatterbot. Section 3 presents a pilot study that was conducted. Section 4 presents the results of an experimental study on chatterbot viability. Finally, section 5 presents the final considerations.

II. Development of the chatterbot

This section has the purpose of presenting the chatterbot conception that we call Ubibot (an acronym for A Chatter**bot** with a **Ubi**quity Level). We have developed it to support the learning in Software Processes Models and Requirements Engineering. Thus, it has knowledge bases that were built to help students of Software Engineering courses when these students have doubts about these subjects. For this work an effort was coordinated between the department's teachers and tutors for the initial modeling of the chatterbot knowledge base. These teachers are used to teach Software Engineering courses and are often faced with students with similar questions. In this sense, knowledge bases were built with questions that were more frequent and subjects that generated greater questions among students.

Besides having knowledge bases in order to help the student in learning, we have built some knowledge bases aiming to assign the chatterbot not only the role of a tutor, but also a stakeholder who is interested in a software for a sector of the university. In this sense, Department's teachers assisted in the definition of the requirements of this software. It was defined that the scope of the software project would be a conference management system, to organize submission and proofreading, similar to EasyChair[1]. The purpose of developing these knowledge bases is to contribute with the development of students' skills in extracting requirements. They were built so that the student had to do several questions to the chatterbot, in order to extract the necessary information.

The chatterbot knowledge bases were developed using the Pattern Matching Technique. The language used to support the construction of chatterbot knowledge was a language based on Extensible Markup Language (XML). We use an inference machine that runs an algorithm called Graphmaster [24]. We chose the Pattern Matching Technique, the XML language and an inference machine with the Graphmaster algorithm, as they are described in the academic literature of the area as being more suitable to the implementation of chatterbots for educational purposes [10], [7]. For the context of this work, we use an open source inference machine called Program O[2], which is implemented in the Hypertext Preprocessor language.

For the construction of the chatterbot was also considered the treatment of information that it sends to the student, taking into account the student's learning context. In this sense, we rely on the work of [12] to define what types of information about the user to consider. We chose to develop a chatterbot that offers answers to the user about their questions, considering their level of expertise, similar to the work of [12]. In this regard, we instantiate the identification module developed in the work of [12] for Software Process Models and Requirements Engineering. We also propose an evolution for the module, since the original module is not able to identify how much the student has learned during the course. To this end, we have developed the chatterbot as a module for the Learning Management System (LMS) Moodle, since we use information about the students that are available in this LMS.

The Ubibot identifies information about the student in

[1]https://easychair.org/conferences.cgi

[2]https://www.program-o.com/

order to assist him in taking into account his real needs and knowledge about the subject he is studying. For this purpose, when offering help to the student, it identifies the student's previous knowledge and the knowledge evolution, the latter only when the teacher makes available the student's grades in the LMS Moodle. In order to identify prior knowledge, a chatterbot module, called MINE, has been developed. It is similar to the one developed by [12] composed of ten objective questions that were extracted from some editions of the National Examination Performance of Students (ENADE)[3] and the National Examination for entrance into Post-Graduation in Computer Science (POSCOMP)[4] related to the content of Software Process Models and Requirements Extraction.

After the student answers the questions that compose the MINE, the system performs the sum of the numbers that translate the student's answer options, totaling an average. In this sense, the system uses the delimiter factor, built and evidenced in Table 1 to identify prior knowledge. The classification was based on the works of [12] and [21].

TABLE I
IDENTIFICATION OF PRIOR KNOWLEDGE OF THE STUDENT

Level of Expertise	Limiting Factor
Basic expertise	$0 \leq$ correct answers ≤ 4
Intermediate expertise	$4 <$ correct answers ≤ 7
Advanced expertise	$7 <$ correct answers ≤ 10

When the student accesses the Moodle environment, the chatterbot will initially identify their prior knowledge about the subjects mentioned, and then it will be available to the student in an HTML block within the course. In this perspective, when questioned by the student, the chatterbot will be aware of the level of knowledge to present answers considering this information. In order to provide custom adaptation, the help messages on the topics of Software Engineering were separated according to the three levels of experience defined. In this sense, the chatterbot when interacting with the student modifies the degree of complexity of their answers, in order not to frustrate the student during the learning of the content. According to [12] this is necessary because a discipline can be made up of students with different levels of knowledge. Thus, a student with a basic level of knowledge, for example, when asking a question, expects the answer to be easy to understand. In contrast, a student with advanced knowledge does not expect a trivial answer. To support the classification of a response to different levels of knowledge, teachers who assist in modeling chatterbot knowledge were consulted.

As mentioned, we created an extension for the module that identifies the student's previous knowledge. In this extension, the aim is to ascertain the student's performance in the discipline he is studying. Thus, we built a module that we call MIDE that analyzes the students' grades that are enrolled in Moodle in Software Engineering courses. When there is a grade in the course available in the Moodle environment, the chatterbot, through MIDE, verifies the grade obtained by the student and registers an update in the previous knowledge. This update basically consists of transforming the grade scored by the student on a scale of zero to ten points and reclassifying the student context. To illustrate the operation of the chatterbot, we constructed a diagram (Figure 1) that represents the main activities that the chatterbot performs to be able to interact with the user taking into account the learning context.

Besides the ability to modify its help messages according to the student's learning context, an important characteristic to be considered by the Ubibot is that it is able to adapt to different types of interfaces, considering its access by means of any device, which are important characteristics to a system directed to the ubiquitous learning. The interface adaptation component was made up using the Boostrap[5] framework. It has been inserted into the Moodle environment as a theme to make the environment provide access-related services on any type of computing device that has access to a browser. The main Ubibot interface is shown in Figure 2.

III. PILOT STUDY

This section discusses the planning and conduction of a pilot study that was conducted with Software Engineering students. We conducted the study with the purpose of verifying students' perceptions about the chatterbot, as well as investigating chatterbot's ability to act as a stakeholder. In this sense, the section was structured so that it is possible to describe how the intervention happened, what activities the students performed, what instruments were used to collect data, and finally, the results that were obtained.

A. Intervention

The Ubibot was socialized with 15 students of an undergraduate course in Computer Science from a higher education institution. These students were studying Software Engineering. The intervention took place over a period of five weeks. We chose the weeks in which the students were learning the contents that are treated in the chatterbot knowledge bases. In the first week, the authors presented the chatterbot to the students, explaining the operation of the system and presenting a brief user tutorial. In addition, at the first meeting the students were explained that their participation was voluntary and therefore the students could decide if they would participate in the research.

At the first week's meeting, participants accessed the environment to interact with the chatterbot. In parallel there was the execution of the MINE module that collected the data referring to the students' level of knowledge. In this way, the students had to answer the questions of the form. In the initial contact with the environment, it was observed that thirteen students (87%) were in the basic knowledge level, that is, they could not answer more than four questions in the MINE questionnaire, in contrast two of them (13%) have managed to

[3]http://portal.inep.gov.br/enade

[4]http://www.sbc.org.br/educacao/poscomp

[5]https://goo.gl/Qv4DUW

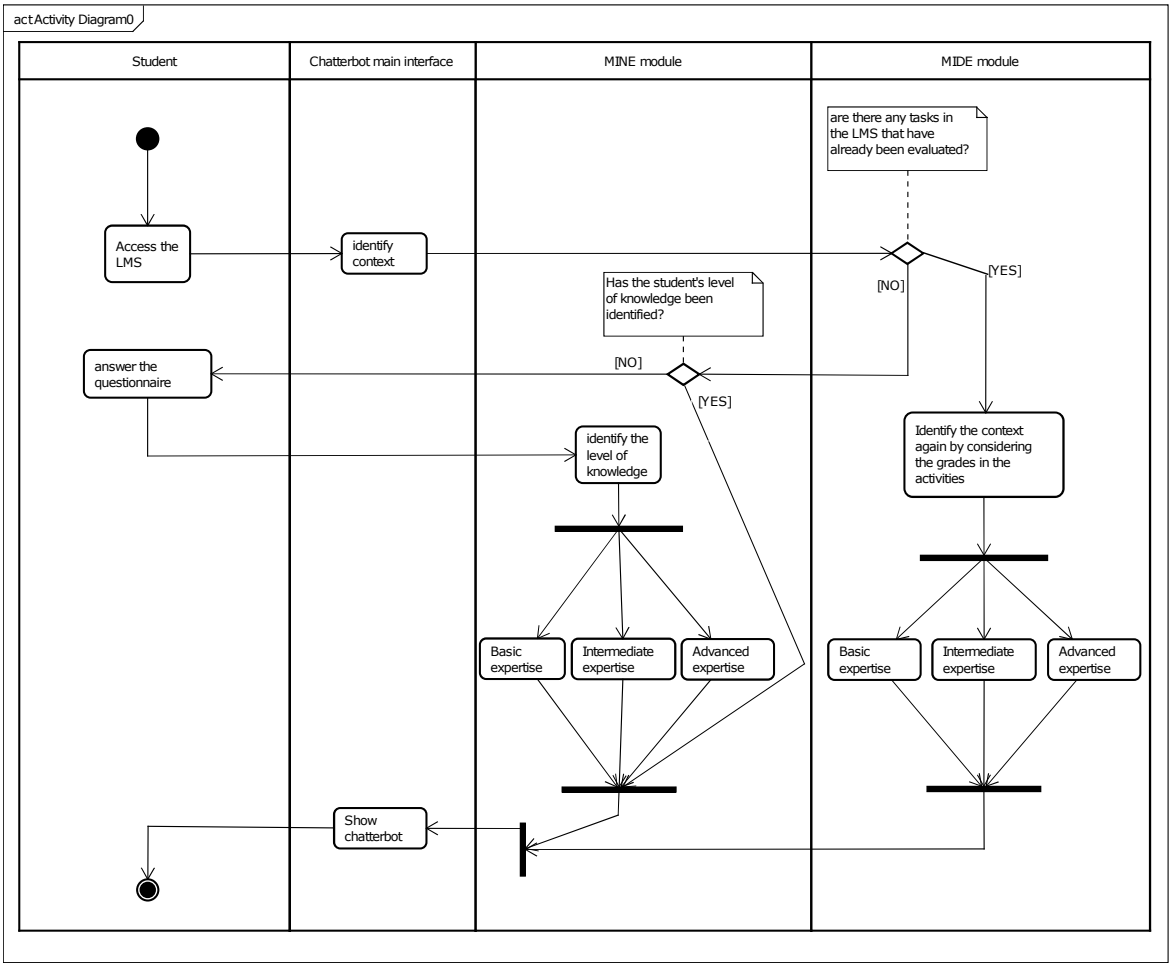

Fig. 1. The main activities carried out by chatterbot

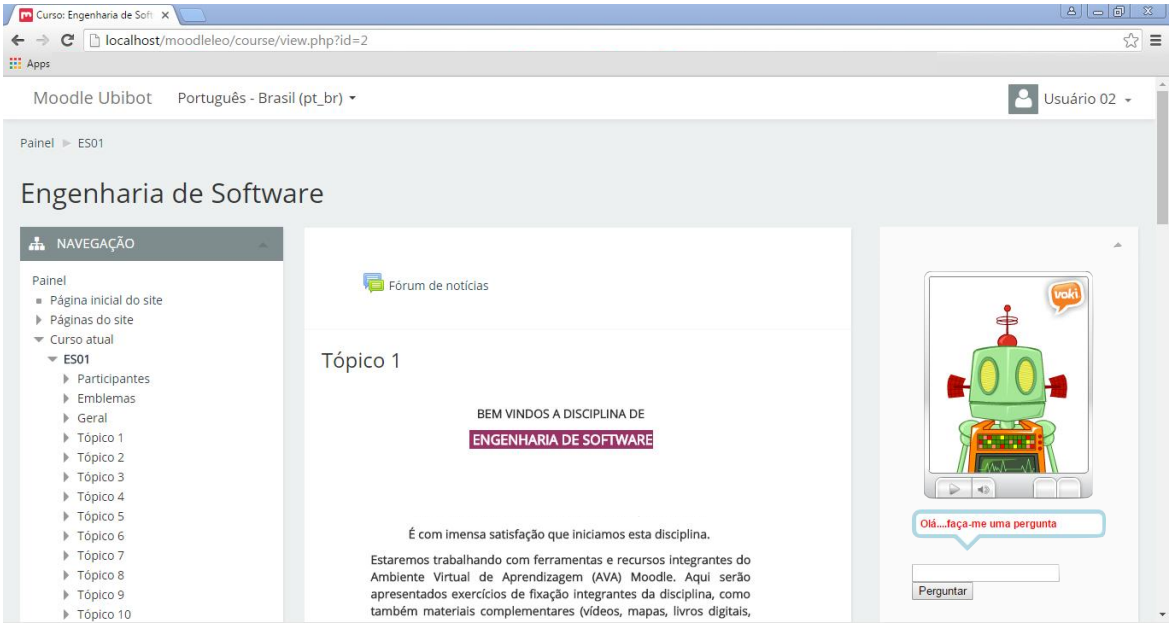

Fig. 2. Chatterbot main interface (in Portuguese)

answer more than seven questions and have gained the level of advanced knowledge.

In the subsequent weekly meetings, the researchers made themselves available to address participants' questions about using chatterbot. During this period the students perform activities in the discipline with the help of chatterbot, in order to familiarize themselves with the system.

In the course of these weeks, three activities were requested for the students to perform, with a defined deadline, similar to the activities carried out in courses taught in the distance learning mode, in which the student must accomplish the activity and send it by the LMS. After receiving the resolutions of each activity, the researchers made the evaluation of them and make available the students' grades in the LMS. At the same time, the MIDE module was in execution, to change the students contexts.

In the fifth meeting two tasks were made available for the participants to perform. Students should use the chatterbot as a support mechanism to accomplish them. The first task was proposed with the aim of the participants evaluating the knowledge bases of the chatterbot, that is, ascertaining if the knowledge bases were well trained, if they could meet the students' demands, and if the chatterbot could help them considering the students context. In this sense, students had to answer some questions about Software Process Models and Requirements Engineering. The second task, in turn, was designed to get students to train their skills in extracting requirements. In this way, the students were asked to interview the chatterbot in order to extract the functional and non-functional requirements and from that writing the requirements document. Thus, each participant would have to build and deliver the requirements document. This document was later evaluated by the researchers, which aims to assess if the students were able to extract all the requirements that were expected.

After using the chatterbot, the students delivered the two tasks performed and answered the evaluation questionnaires provided by the research authors. Therefore, through the questionnaires it was possible to collect data on aspects inherent to the operation of the system and through each task it was possible to have knowledge about how much the chatterbot can help the students in the accomplishment of tasks and on the possibility of using it as training facility requirements extraction.

B. Instrumentation

For the pilot study context, we prepared three activities that were executed during the five weeks in which students had the opportunity to interact with the chatterbot. In addition, two tasks were prepared for the students to perform in a face-to-face meeting, held in the fifth week. The first task was a list of exercises with questions, in which the students would have to talk about the subject, being able to consult chatterbot to assist in the answers. In the second task, it was specified that the students should construct a requirements document, from the chatterbot conversation. To support the achievement of the second task, a template has been made available.

Besides the activities, two instruments were built for data collection. The first one consists of a survey that aims to verify the students' perceptions regarding Ubibot, composed of ten objective questions. The questionnaire was validated with the Cronbach's Alpha Test[6], in order to prove that the internal consistency of the questionnaire is considered good since $\alpha = 0.87$. Table 2 presents the questions prepared to evaluate the chatterbot.

TABLE II
SURVEY QUESTIONS

Question	Type
(1) When interacting with chatterbot for the first time, was the experience exciting?	Likert scale (Mandotory)
(2) Did the chatterbot contribute to the execution and conclusion of tasks?	Likert scale (Mandotory)
(3) When using chatterbot did you have the opportunity to create your own knowledge?	Likert scale (Mandotory)
(4) Using chatterbot will you improve your knowledge?	Likert scale (Mandotory)
(5) Was there a delay in the responses provided by chatterbot?	Likert scale (Mandotory)
(6) Are the answers provided by chatterbot repetitive?	Likert scale (Mandotory)
(7) Has chatterbot crashed or interrupted its use?	Likert scale (Mandotory)
(8) Does chatterbot integration with Moodle result in a user-friendly interface?	Likert scale (Mandotory)
(9) Is chatterbot unreliable?	Likert scale (Mandotory)
(10) Do you want to continue using chatterbot?	Likert scale (Mandotory)

The second questionnaire used in the research was an adaptation of the System Usability Scale (SUS)[5]. This instrument aims to evaluate the usability of products[5]. According to [1] it is often used in systems evaluations. In the study of [12] the authors recommend the use of this instrument to ascertain the usability of chatterbot. The SUS is composed of ten questions, all of which are based on the Likert scale, whose response options can range from "Strongly Disagree" (1 point) to "Strongly Agree" (5 points). Table 3 presents the SUS issues that were used to measure agent usability.

TABLE III
SYSTEM USABILITY SCALE

Question	Type
(1) I think that I would like to use the Ubibot frequently.	Likert scale (Mandotory)
(2) I found the Ubibot unnecessarily complex.	Likert scale (Mandotory)
(3) I thought the Ubibot was easy to use.	Likert scale (Mandotory)
(4) I think that I would need the support of a technical person to be able to use the Ubibot.	Likert scale (Mandotory)
(5) I found the various functions in the Ubibot were well integrated.	Likert scale (Mandotory)
(6) I thought there was too much inconsistency in the Ubibot.	Likert scale (Mandotory)
(7) I would imagine that most people would learn to use the Ubibot very quickly.	Likert scale (Mandotory)
(8) I found the Ubibot very cumbersome to use.	Likert scale (Mandotory)
(9) I felt very confident using the Ubibot.	Likert scale (Mandotory)
(10) I needed to learn a lot of things before I could get going with the Ubibot.	Likert scale (Mandotory)

In an application of the SUS, when obtaining the answers of the user, it is necessary to calculate the SUS score. The score of each item can range from 0 to 4. In order to obtain the odd scores, the answers of the odd questions are subtracted by 1, while the even scores are given by 5 minus the value of the answers of the even questions. The partial SUS score, which can range from 0 to 100, is found by the sum of its scores multiplied by 2.5. Finally, to obtain the global SUS score, the mean values obtained by the partial SUS are calculated.

C. Results and Discutions

This section aims to present the results of the students perceptions about the Ubibot functionalities, the evaluation of usability and the data about the interaction between the students and Ubibot while the chatterbot played the role of a stakeholder.

1) Perception of students: This section seeks to present the results of the first questionnaire applied. The results obtained through the survey are synthesized and presented in Figure 3. It should be noted that all questions had the same response options. In Figure 3 are presented only the response options that were pointed out by participants.

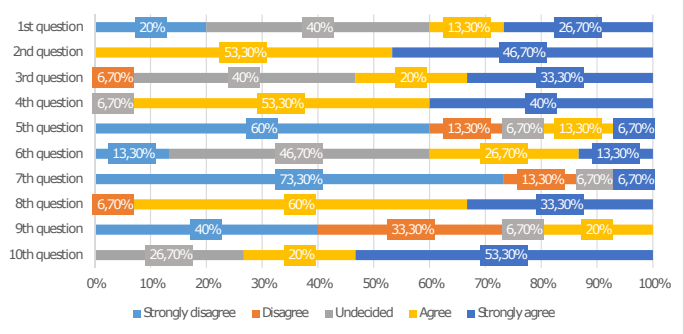

Fig. 3. Results of the ten likert scale questions

The first question was intended to understand if the students were excited when they used chatterbot for the first time. As a result, 26.7% agreed totally, 13.3% agreed in part, 40% were undecided and 20% disagreed completely. With these results, it is possible to notice that some participants did not find the experience exciting. This result may be directly related to the aid that the chatterbot issued, because most students (87%) received help in the first week of a knowledge base belonging to the basic context, so perhaps for this reason they were not so excited during the first experiment.

The second question had the goal of ascertaining if the agent contributed in some way to the accomplishment of the tasks carried out by the students. The results of this question were positive, since 53.3% of the respondents indicated the option ”Strongly agree” and 46.7% ”Agree”. The results on the graph show that Ubibot has been able to assist the participants according to their needs. However, as it turned out, there was some time during the implementation of activities that the agent failed to fully assist a portion of the sampling.

The third question was to verify if Ubibot offered opportunities for students to construct new knowledge in an autonomous way. Among respondents, 40% of the evaluators were undecided, 33.3% fully agreed, 20% agreed in part and one (or 6.7%) participant disagreed. According to the data obtained in this question, the vast majority of respondents agreed in some way that Ubibot allows the student to be able to create their own knowledge.

The fourth questioning had the goal of detecting whether students believe that Ubibot will allow them to improve their knowledge through utilization. Its possible to verify satisfactory results, since 93.3% of the students agreed, with 40% of the respondents fully agreeing, while 53.3% agreed partially. In this sense, it was possible to verify that chatterbot is a support mechanism with great potential that can contribute to the construction of knowledge.

The fifth question aimed to verify if the chatterbot took too much time to help the student. The results show that 60% disagreed completely, revealing that the responses are instantaneous. However, the results show that some students may have identified that at certain times there was delay, this can be observed through the 6.7% of participants who fully agreed and 13.3% partially agreed. It should be noted that the delay in the responses emitted by the chatterbot may be related to several factors such as machine processing that supports the execution of the chatterbot and the operation of the interpreter. A concise response can only be traced through new studies to analyze the factors that may cause this delay in response.

In the sixth question we tried to find out if the answers given by chatterbot were repetitive. The results show that 46.7% of participants judged indecision, 26.7% partially agreed, 13.3% fully agreed and 13.3% disagreed completely. Participants were expected to disagree with the affirmative. However, the discrepant results show that there is a need to expand knowledge bases, since there are some bases that were constructed with the premise of several questions for a response, for example the following input patterns ”talk about the cascade model” and ”describe on the cascade model” have the same response to a given context and few random resources with response options. One improvement option that can be done is to create several answers to the question using an XML language structure called rondom.

When questioned on the seventh question about catching chatterbot crashes or interruptions, 73.3% disagreed completely, 13.3% disagreed, 6.7% were undecided, and 6.7% totally agreed. Although there was one participant who fully agreed, there was a predominance of negative outcomes (86.6%), which suggests that most evaluators found no interruptions or blockages during application.

The eighth question had as goal to verify if the integration of Ubibot in Moodle managed to result in a friendly interface. Repercussion reveals that the participants rated the interface as friendly, given that 33.3% agreed totally and 60% agreed in part. In parallel, one participant revealed that he disagreed in part.

In the ninth question, the responses ranged from ”Strongly disagree” to ”Agree”, With 73.3% of the students judging that the agent is trustworthy. Based on the results, it is possible to infer that the index found is probably associated with readiness in the recommendations. In other words, it is possible to assume that 20% of the sample presumed that the agents recommendations failed to meet all the needs of the students. However, it is worth mentioning that these 20% agreed in part, that is, they only agreed with one of the characteristics of the whole and therefore it is presumed that some needs of these participants were attended by the chatterbot.

Finally, in the tenth question we asked the students if they would like to continue using chatterbot in the discipline. Among the results, 53.3% were found to be interested, 20% partially agreed, which favored the inclusion of interests at certain times, and 26.7% are undecided and did not give their opinion. It is important to note that there was no discord, which is a demonstration of performance and perhaps the interest of students in continuing to use chatterbot.

In general, the results presented demonstrate that the Ubibot was able to satisfy the students with satisfaction during the interaction, meeting our expectations, both from an educational and technical point of view. From the educational point of view, it was able to offer support in the execution of activities, the opportunity for students to build new knowledge, and support to improve their understanding of the contents. On the technical aspects, the chatterbot was able to act as a synchronous mechanism, not delaying to respond to the requests of the student. However, there is a need to expand chatterbot knowledge bases. Another characteristic observed is that the students recognized that chatterbot has a friendly interface. Although the data reveals an user-friendly interface, we chose to use an instrument established to examine the usability of the chatterbot. Thus, the next section will present the usability assessment.

2) Usability: The fifteen respondents answered the ten questions related to the usability assessment of the integration. The partial scores of each user are shown in Figure 4.

According to the partial SUS scores, its minimum value was 60 points (users 6 and 11), while its maximum was 97.50 points (user 12). The mode values were 65 points (users 1, 5, 9 and 10). The average was 70.33 points, which represents the overall SUS score, suggesting that the usability of the integration obtained a good acceptance by the users, a result considered good, since, according to [1] the average value of the overall SUS is 69.5 points. [23] infers that a SUS score higher than 68 points is considered above the average and consequently good.

The Figure 4 presents a comparison between the Ubibot SUS result and the SUS midpoints for different types of product interfaces. These data were reported by [1], which points out that products that have a web application interface will have good usability if they have a SUS value of at least 68.2 points and 65.9 points for mobile products. Considering the ability of Ubibot to be accessed by both desktops and mobile devices, from the results found it is possible to infer that Ubibot meets the usability requirements.

3) Chatterbot performance as a Stakeholder: We evaluated the requirements documents constructed by the students from the original requirements document that was built by the group of professors of our department, who assisted in the modeling of chatterbot knowledge. We analyzed the functional requirements that each student was able to extract, the non-functional requirements and the actors / users of the system. The original requirements document written by the professors of our department specifies a total of 13 functional requirements and 5 non-functional requirements. In addition, the document reveals that there are three possible users for the system: the system administrator, the author who submits the article and the reviewer, and the author can assume the role of reviewer. On the number of authors, 13 students were able to identify the existence of two actors (reviewer and authors), the other students described that the system would have three actors, as expected by the specification.

When analyzing the functional requirements that each participant was able to extract from the chatterbot, we observed that only one student was able to recognize all the requirements that were expected, that is, that were defined by the teachers in the software specification. The Figure 5 shows the results obtained, in which it is possible to observe the number of functional requirements that each student was able to identify. From these results, we observed the median number of requirements that the students identified in order to have a better idea of the total number of requirements that the participants identified. As such, we obtained a median of 10. The standard deviation value was also calculated, revealing a number equal to 1.56. This data reveals that there was not much discrepancy between the number of functional requirements that the student group was able to extract.

Regarding non-functional requirements, it was observed that four students were able to identify the five non-functional requirements that were predicted. Three students described in the requirements document only one requirement. We calculated the median and standard deviation values of the results and found that the median was equal to 3 and the standard deviation value was 1.56, which indicates that 50% of participants could identify more than 3 non-functional requirements. Figure 6 shows the number of nonfunctional requirements that each student was able to extract.

From the results obtained, also considering the perceptions of the students that used the chatterbot, we believe that it can be used as a mechanism to support the training of requirements. When analyzing studies that discuss the use of ”real world” projects that require clients and / or stakeholders, such as the study of [9], we believe that our chatterbot can be exploited when there is lack of ”real world” clients, since they have the skills to simulate a stakeholder.

After conducting the pilot study, we verified the need to investigate the impact of chatterbot's ability to interact with the student in a context-aware manner. In the pilot study evaluations, we do not care about our chatterbot being context-sensitive and supporting the student. Therefore, it was necessary to analyze if chatterbot help messages can influence the use of the it. Regarding this issue, we created an experimental study that is described in the next section.

IV. Experimental study

We conducted an experimental study, setting up a protocol that was established based on the recommendations of [28]. The objective of the experimental study was to analyze the instructional efficiency of Ubibot, verifying if the information emitted by Ubibot can influence the accomplishment of the activities carried out by the students. To do so, we have

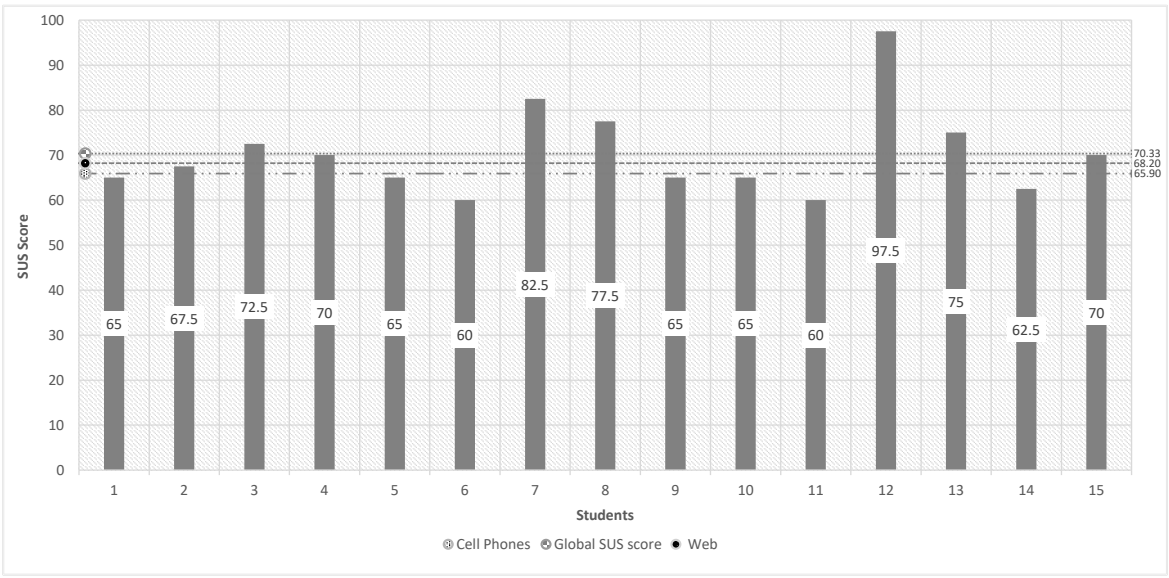

Fig. 4. Score SUS by Students

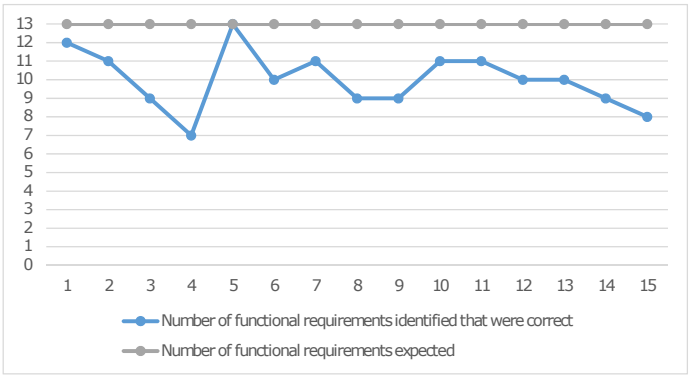

Fig. 5. Functional requirements extracted by the student from interaction with chatterbot

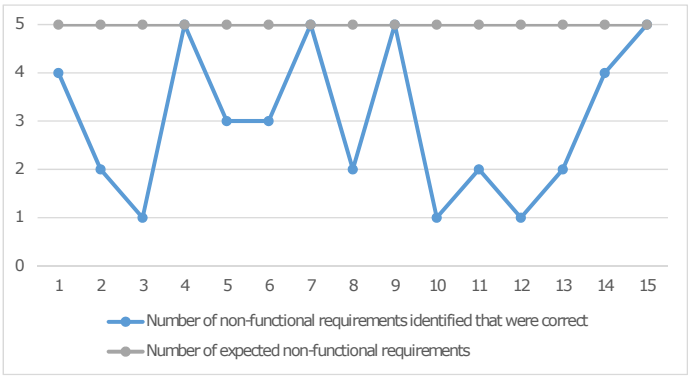

Fig. 6. Non-functional requirements extracted by the student from interaction with chatterbot

created another version of our chatterbot. This version we call OutContextbot (an acronym for a chatter**bot** with**out context** awareness), it does not have the context identification modules (MINE and MIDE) and their knowledge bases mix the three levels that the user could be. Therefore, we conducted the study comparing the Ubibot chatterbot that assists the student in a conscious way with the OutContextbot chatterbot that assists the student without being aware of context.

The goals of the experiment were specified in parts as a paradigm similar to Goal/Question/Metric (GQM): **to analyze** the instructional efficiency of the chatterbot sensitive to student's context in solving students' questions compared to the instructional efficiency of a chatterbot that is not sensitive to student's context **with the purpose of** verifying its applicability, **with respect to** the interaction with the students, **from the point of view** of the researchers, **in the context** of students of an undergraduate course in Computer Science, enrolled in a Software Engineering discipline. These goals were stipulated from the definition of a research question: *does the context-sensitive chatterbot present more efficient instructions than chatterbot without context-sensitivity?*

To answer this question, we defined as metrics the students feedback who would interact with chatterbots. To measure the feedback, we elaborate a question: "Does the chatterbot contribute to the achievement of the concept test?". To support, we use a Likert scale with five response options, varying between Strongly disagree and Strongly agree. Thus, we prepared an intervention so that a class composed of 30 Software Engineering students participated in the experiment. We invited a Software Engineering class to participate in our experiment, different from the one that participated in the pilot study, in order to avoid bias in the experimental study. From the total of 30 students who were invited, only 19 students accepted to participate in the experiment. Therefore, the class was divided in two groups, a group composed of 10 people used Ubibot and the other group used the OutContextbot.

For the context of this experiment, students had not learned

the Software Process Models content. As a result of this, we used two meetings at a time outside the discipline, one per week. In the first week, students had an expositive lecture about the content. Already in the second week, we made chatterbots available to the class. We randomly assigned the addresses in which the chatterbots were hosted, so that each student in the course had one of the versions of our chatterbots. For this reason, 10 students were selected to interact with Ubibot and the other 9 students with OutContextbot. After the distribution of the addresses, a theoretical test of the content was delivered to the students and the students were informed that they should use the chatterbots to answer the questions about the exercises.

It is important to point out that we have built an environment conducive to the experiment, so that students could access chatterbots because they were hosted on local servers. This environment was configured in order to prevent students from using other teaching materials available in the LMS (the teaching materials of the course available in Moodle were concealed for students) or on the Internet. In addition, it is important to make it clear that we had not previously informed the students that they would take the theoretical test, i.e. the students only learned that they would take the test in the second class. The theoretical test encompassed only the subject of software process models and was prepared by the authors of this research from issues extracted from previous editions of the ENADE and POSCOMP exams, since these exams are generally prepared by specialists in the area. This test was not designed for evaluation, but rather to provide conditions to students in which they are faced with difficult situations where might emerge questions, forcing them to use chatterbots to assist them in the resolution of these questions.

After the students take the theoretical test, we ask them to give a feedback about the real contribution of the chatterbot in conducting the theoretical test, considering the messages that the chatterbot issued during the interaction. This feedback was to answer the question mentioned earlier in this section. By using the Likert scale as a response option, each response option was converted to a score ranging from zero to four points, with zero being assigned to the "Strongly disagree" and 4 assigned to the "Strongly agree" alternative. From this point, we transform the answers into quantitative data, aiming to perform a hypothesis test. In this sense, to support our study, from our research question, we defined two hypotheses and applied the hypothesis test. The null hypothesis is that the responses emitted by Ubibot are as efficient as those emitted by OutContextbot (i.e. *Ho: Efficiency (Ubibot) = Efficiency (OutContextbot)*). The alternative hypothesis is that there is a difference in the efficiency terms from messages issued by Ubibot and OutContextbot (i.e. *Ha: Efficiency (Ubibot)* $\neq$ *Efficiency (OutContextbot)*).

After converting student answers into quantitative data, we analyzed the data. To perform the statistical tests, a confidence interval was adopted for α = 0.05. After alpha is specified, we calculate the normality of the data using the Shapiro-Wilk test. When we applied the test, we observed that the samples were not Gaussian, since p-value for the data of the group that used Ubibot was p = 0.0184 and the p-value for the data of the group that used the OutContextbot was p = 0.0148. In this sense, considering that our samples are independent and the distribution is not normal, we used the Mann-Whitney test. The Mann-Whitney test resulted in a p-value lower than α, given that p = 0.0250. Based on this, we reject the null hypothesis (Ho) and we can identify that one of our chatterbots stands out from the other.

When we did a descriptive analysis of the data using descriptive statistics, we could observe that the median scores of the group that used Ubibot is higher than the median of the group that used the OutContextbot. What is observed in the results is that the group that used the Ubibot presented a greater agreement index in relation to the instructions emitted by the chatterbot than the group that used the OutContextbot chatterbot. Thus, it can be deduced that chatterbot presents more efficient instructions than chatterbot without context-sensitivity.

V. Conclusions

This article aimed to present the conception and evaluation of a chatterbot called Ubibot that is able to act as a tutor and a stakeholder. When acting as a tutor, it identifies information about the student (level of knowledge and student performance in the discipline), with the purpose of supporting the student in a context-aware manner. By acting as a stakeholder, it intends to train requirements extraction skills. It is important to emphasize that the performance of a stakeholder is independent of the student's learning context. Regarding design, we present technical characteristics of chatterbot development and operation. In the evaluation, we tried to find out if it was able to satisfactorily attend the students, both in the sense of acting as a context-aware tutor, or if their interface did not have problems that could contribute to the minimization of its use. We have also constructed another chatterbot, the one with no ability to identify the student's context, in order to compare if the student believes that the assistance of a context-sensitive chatterbot are more efficient than the assistance of a chatterbot without the ability to emit information in a aware-context manner.

In future work, we believe that we could conduct some experiments to evaluate our chatterbot with a larger group of students, since in both the pilot study and the experimental study, our sampling is not significant. We need to conduct studies with a larger group of students. We would also like to expand Ubibot's knowledge base and experiment with approaches that favor active learning methodologies such as Project Based Learning in order to get students to use chatterbot in the absence of a client or stakeholder. In addition, thinking about the benefits that the community can bring, we are trying to turn our chatterbot into an open educational resource.

REFERENCES

[1] A. Bangor, P. Kortum, and J. Miller. Determining what individual sus scores mean: Adding an adjective rating scale. *Journal of Usability Studies*, 4(3):114–123, 2009.

[2] L. Benotti, M. C. Martinez, and F. Schapachnik. A tool for introducing computer science with automatic formative assessment. *IEEE Transactions on Learning Technologies*, 11(2):179–192, 2017.

[3] T. W. Bickmore, L. M. Pfeifer, D. Byron, S. Forsythe, L. E. Henault, B. W. Jack, R. Silliman, and M. K. Paasche-Orlow. Usability of conversational agents by patients with inadequate health literacy: Evidence from two clinical trials. *Journal of Health Communication*, 15:197–210, 2010.

[4] F. Brady. Cambridge university study states software bugs cost economy $312 billion per year. available at: <http://bit.ly/2lmwt8a>, 2013.

[5] J. Brooke. Sus: A retrospective. *Journal of Usability Studies*, 8(2):29–40, 2013.

[6] L. J. Cronbach. Coefficient alpha and the internal structure of tests. *Psychometrika*, 16(3):297–334, 1951.

[7] K. Denecke, S. Lutz Hochreutener, A. Pöpel, and R. May. Talking to ana: A mobile self-anamnesis application with conversational user interface. In *International Conference on Digital Health*, pages 85–89, New York, NY, USA, 2018. ACM.

[8] O. V. Deryugina. Chatterbots. *Scientific and Technical Information Processing*, 37(2):143–147, 2010.

[9] M. L. Fioravanti, B. Sena, L. N. Paschoal, L. R. Silva, A. P. Allian, E. Y. Nakagawa, S. R. Souza, S. Isotani, and E. F. Barbosa. Integrating project based learning and project management for software engineering teaching: An experience report. In *49th ACM Technical Symposium on Computer Science Education*, pages 806–811, 2018.

[10] S. Ghose and J. J. Barua. Toward the implementation of a topic specific dialogue based natural language chatbot as an undergraduate advisor. In *International Conference on Informatics, Electronics and Vision*, pages 1–5, 2013.

[11] R. Q. Gonalves and M. Thiry. Development of a game to support the teaching of requirements engineering: The requirements island. In *9th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*, pages 358–3691, Florianpolis, Brazil, 2017.

[12] F. Herpich, F. B. Nunes, G. B. Voss, and R. D. Medina. Three-dimensional virtual environment and npc: A perspective about intelligent agents ubiquitous. In F. M. M. Neto, R. de Souza, and A. S. Gomes, editors, *Handbook of Research on 3-D Virtual Environments and Hypermedia for Ubiquitous Learning*, chapter 21, pages 510–536. IGI Global, Hershey, EUA, 2016.

[13] P. Hsuan, C. R. Dow, K. H. Chen, Y. Y. Chen, and Y. H. Li. An ubiquitous teaching assistant using knowledge retrieval and adaptive learning techniques. In *International Conference on Integration of Knowledge Intensive Multi-Agent Systems*, pages 121–126, 2007.

[14] IEEE. Ieee standard glossary of software engineering terminology. Technical report, 1990.

[15] M. D. Leonhardt, L. Tarouco, R. M. Vicari, E. R. Santos, and M. d. S. d. Silva. Using chatbots for network management training through problem-based oriented education. In *7th IEEE International Conference on Advanced Learning Technologies (ICALT)*, pages 845–847, 2007.

[16] V. López, E. M. Eisman, and J. L. Castro. A tool for training primary health care medical students: The virtual simulated patient. In *20th IEEE International Conference on Tools with Artificial Intelligence*, pages 194–201, 2008.

[17] S. Ouhbi, A. Idri, J. L. Fernández-Alemán, and A. Toval. Requirements engineering education: A systematic mapping study. *Requirements Engineering*, 20(2):119–138, 2015.

[18] L. N. Paschoal, P. M. M. Chicon, and G. A. M. Falkembach. Ubibot: um agente conversacional ciente do contexto de aprendizagem do usuário. *RENOTE - Revista Novas Tecnologias na Educação*, 14(1):01–10, 2016.

[19] L. N. Paschoal, P. M. M. Chicon, and G. A. M. Falkembach. Concepção, implementação e avaliação de um agente conversacional com suporte à aprendizagem ubíqua. *RENOTE - Revista Novas Tecnologias na Educação*, 15(1):01–10, 2017.

[20] L. N. Paschoal, P. M. Mozzaquatro, A. L. Krassmann, and M. O. Binelo. Ubibot: Agente inteligente consciente do contexto de aprendizagem do usurio integrado ao ambiente moodle. In *XXI Congreso Internacional de Informtica Educativa (TISE)*, pages 95–104, 2016.

[21] C. C. Possobom, A. R. K. Mühlbeier, F. B. Nunes, A. Carvalho, R. B. Gomes, and R. D. Medina. Uma aplicação dinâmica para detectar o nível de conhecimento do aluno. In *XXIII Ciclo de Palestras Novas Tecnologias na Educação*, pages 01–10, 2014.

[22] R. Pressman and B. Maxim. *Software Engineering-8ª Edition*. McGraw Hill Brasil, São Paulo, Brasil, 2016.

[23] J. Sauro. Measuring usability with the system usability scale (sus). available at: <https://measuringu.com/sus/>, Feb. 2011.

[24] B. A. Shawar and E. Atwell. Chatbots: can they serve as natural language interfaces to qa corpus? In *6th International Conference Advances in Computer Science and Engineering*, pages 183–188, 2010.

[25] T. J. Shippey. *Exploiting Abstract Syntax Trees to Locate Software Defects*. PhD thesis, University of Hertfordshire, School of Computer Sciences, United Kingdom, 2015.

[26] I. Sommerville. Software testing. In I. Sommerville, editor, *Software Engineering*, chapter 8, pages 144–163. Pearson Addison-Wesley, Boston, Massachusetts, EUA, 2016.

[27] A. F. van Woudenberg. A chatbot dialogue manager, chatbots and dialogue systems: A hybrid approach. Master's thesis, Faculty of Management, Science and Technology - Open University of the Netherlands, Heerlen, Netherlands, 2014.

[28] C. Wohlin, P. Runeson, M. Host, M. C. Ohlsson, B. Regnell, and A. Wesslen. *Experimentation in Software Engineering*. Springer Berlin Heidelberg, 2012.

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