

# A Chatbot for Goal-Oriented Requirements Modeling

Danilo Arruda[1], Matheus Marinho[1], Eric Souza[2], and Fernando Wanderley[1(✉)]

[1] Universidade Católica de Pernambuco, UNICAP, Recife, Pernambuco, Brazil
darruda.silva2@gmail.com, matheuzmarinho@gmail.com,
fernando.wanderley@unicap.br
[2] Universidade Nova de Lisboa, UNL, Lisboa, Portugal
er.souza@campus.fct.unl.pt

**Abstract.** Goal-Oriented Requirements Engineering approaches, in which the KAOS framework plays a key role, have been widely used for eliciting software requirements because they provide an easier way of communicating among stakeholders. However, the goal-oriented requirements modeling is not an easy way for novice requirements engineers. These professionals need more support in creating KAOS models. Recent studies have focused on the applicability of Artificial Intelligence techniques (e.g., Natural Language Processing – NLP) to support Requirements Engineering activities. In this sense, this paper aims to describe a way to support requirements elicitation for novice requirements engineers through the use of NLP within a chatbot. The chatbot (*KAOSbot*) acts as a KAOS modeling assistant. To evaluate our hypotheses about perceived efficacy from the novice requirements engineers perspective, we performed a quasi-experiment concerning *KAOSbot's* perceived ease of use, perceived usefulness and intention to use. The results show that *KAOSbot* tool is a promising approach for specifying KAOS models because it was perceived as easy to use, useful, and the participants intend to use it in the future.

**Keywords:** Goal-Oriented Requirements Engineering · KAOS modeling · Chatbot · Natural Language Processing · Quasi-experiment

## 1 Introduction

Goal-oriented requirements engineering (GORE) is a paradigm concerned with the use of goals for eliciting, documenting, and modifying requirements [30]. The main motivations of GORE approaches are [8]: (i) greater understanding of the client's objectives; (ii) better communication among project stakeholders; (iii) ability to trace the origin of a software requirement; (iv) conflict management among software requirements. The KAOS framework is the most commonly used GORE approach [8,28,32], and makes it possible to build a model of the whole

© Springer Nature Switzerland AG 2019
S. Misra et al. (Eds.): ICCSA 2019, LNCS 11622, pp. 506–519, 2019.
https://doi.org/10.1007/978-3-030-24305-0_38

system, not just the software part of it [21]. However, the goal modeling task is not an easy way for novice requirements engineers [22,30]. As a consequence, it requires more support in the KAOS modeling tasks by less experienced professionals [30].

Recent studies have focused on the applicability of Artificial Intelligence techniques to support Software Engineering activities [5,14]. For example, the needs to extract information from textual documents used as requirements source has motivated a lot of researches on the application of Natural Language Processing (NLP) in requirements engineering area [4,9,11,25].

In this sense, this paper aims to describe a way to support requirements elicitation for novice requirements engineers through the use of NLP within a chatbot. For this, we develop the KAOSbot, a proof-of-concept chatbot for KAOS modeling that acts as modeling assistant for novice requirements engineers. To evaluate our hypotheses about perceived efficacy from the novice requirements engineers perspective, we performed a quasi-experiment to check *the perceived ease of use and the perceived usefulness* for the KAOSbot tool when producing KAOS models. In addition, we also evaluate the participants' intention to use the tool in the future. This quasi-experiment was performed following the best practices of Evidence-Based Software Engineering (EBSE) community. The results show that *KAOSBot* tool is a promising approach for novice requirements engineers specify KAOS models because it was perceived as easy to use, useful, and the participants intend to use it in the future.

The rest of this paper is organized as follows: Sect. 2 discuss about the background theory contained the main concerns for our work: (i) Goal-Oriented Requirements Engineering including KAOS framework, (ii) Natural Language Processing for Requirements Engineering and (iii) Chatbot. Section 3 describes the architecture and operation of KAOSbot. Section 4 presents the evaluation of KAOSbot. Section 5 presents the related works to our paper and Sect. 6 the main conclusions about our work.

## 2 Background

This section presents a brief overview of the three essential elements of our approach: (i) Goal-Oriented Requirements Engineering, (ii) Natural Language Processing for Requirements Engineering, and (iii) Chatbot.

### 2.1 Goal-Oriented Requirements Engineering

Goal-Oriented Requirements Engineering (GORE) [29] is a paradigm that concerns in use goals in the various activities of the Requirements Engineering (RE) such as eliciting, elaborating, structuring, specifying, analyzing, negotiating, documenting, and modifying requirements. The KAOS approach [29] is the most commonly used GORE approaches [8,28,32]. The KAOS model helps to fit problem descriptions by allowing requirements engineers to define and manipulate concepts relevant to problem description and improves the domain analysis

by providing a systematic approach for discovering and structuring requirements. KAOS also clarifies the responsibilities of all the project stakeholders and facilitates the stakeholder's communication about the requirements by offering a graphical model to discussion [21]. The main advantage of KAOS is its ability to align requirements for business goals. This alignment increases the chances that the new development will add value to business [1].

In KAOS, a goal can be defined as the intent statement that the system must achieve. It is refined into sub-goals that explain HOW that goal should be achieved. Goal refinement ends when a sub-goal is performed by an agent. Then, this sub-goal becomes a requirement or expectation. An agent can be a human (environment agent) or an automated component (system agent). Requirements must be satisfied by a system agent. Expectations must be satisfied by an environment agent.

This work aims to support less experienced requirements engineers on creating KAOS models through a chatbot tool.

### 2.2 Natural Language Processing for Requirements Engineering

Natural Language Processing (NLP) is a subset of Artificial Intelligence that concerns the development of computer programs related to understanding, writing or speech in a Natural Language (NL). [6] NLP covers a set of techniques for manipulating and analyzing NL in an automated manner [11,13,25]. Text mining is one of the NLP techniques used to derive high-quality information from text, and there are some available tools in the literature to facilitate its processing (e.g., Stanford CoreNLP [16]). Text mining has often been used to improve and automate RE tasks [15]. For example, it is used to detect inconsistencies in requirements [9], and construct domain models from textual documents [13,19].

In this work, we develop a chatbot tool to interact with user in a textual way. In this sense, NLP techniques was used to interpret what the user writes.

### 2.3 Chatbot

Thanks to advances in artificial intelligence and NLP techniques, conversational systems have been increasingly used [31]. Chatbots are systems that interact with humans to answer questions on a specific domain [26] and they are used in several domains such as customer service, education, and financial advising [31]. Through an interface, people talk by voice or by typing with a computer system in the same way they would chat others people. Because there is no need for installation packages, chatbots are fast and trouble-free to be used in practice [20]. In general, a chatbot can understand the context of a conversation and learn from the chats, improving itself over time by using Machine Learning techniques [2].

We have created a chatbot tool responsible for conducting a conversation with the user to elicit requirements within the context of the KAOS goal-oriented requirements engineering method.

## 3 KAOSbot

This section describes a proposed solution to facilitate generating KAOS models by less experienced requirements engineers. First, we introduce the approach to elicit goals through a chatbot tool. After, we described the chat flow process (KAOSbot Interface) and the algorithm used to identify the goal concept in natural language sentences written in English (KAOSbot Service).

### 3.1 The Approach

We have proposed a chatbot-based tool to support goal-oriented requirements elicitation with the KAOS framework. This tool, named KAOSbot, has the role of hold a chat with the user to extract relevant information about a domain in question. Therefore, KAOSbot identifies Goals, Expectations, Requirements, and Agents through NLP techniques.

KAOSbot is composed of two parts: the *graphical interface* and *service*. The *graphical interface* is responsible for conducting the conversation with the user. On another hand, the KAOSbot's *service* is responsible for getting natural language sentences and extracting KAOS concepts in them.

Figure 1 shows a chat example between the human user and the chatbot. The user interacts with the KAOSBot interface through textual messages written in English. The KAOSbot Interface forwards the user's message to the KAOSbot service. Then, the KAOSbot Service identifies and extracts the KAOS concepts from the user's message using NLP techniques. Next, KAOSbot Service returns KAOS concept extracted to the KAOSbot Interface. After receiving the KAOS concept, KAOSbot Interface knows precisely where to represent the concept into the model, through the chat context.

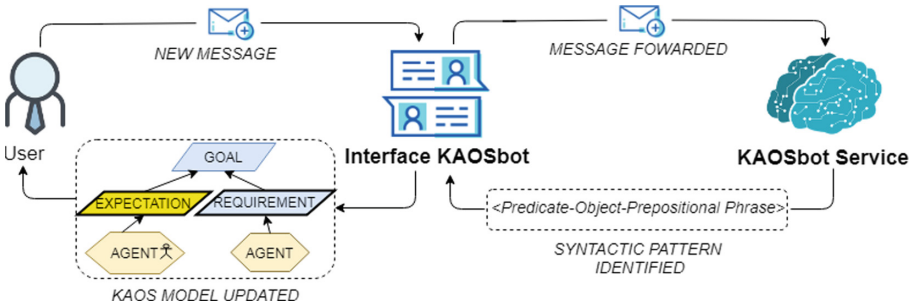



**Fig. 1.** Approach overview

### 3.2 KAOSbot Interface

Figure 2 shows the three areas that makes the KAOSbot's graphical interface: legend, modeling, and chat. The legend area shows the graphic representations of

KAOS concepts (Fig. 2-left). The modeling area shows the KAOS model which is being generated (Fig. 2-middle) according to messages typed on chat (Fig. 2-right). Through this chat, KAOSbot extracts useful information related to KAOS concepts and updates the model in the modeling area, providing continuous visual feedback on how current the model is.

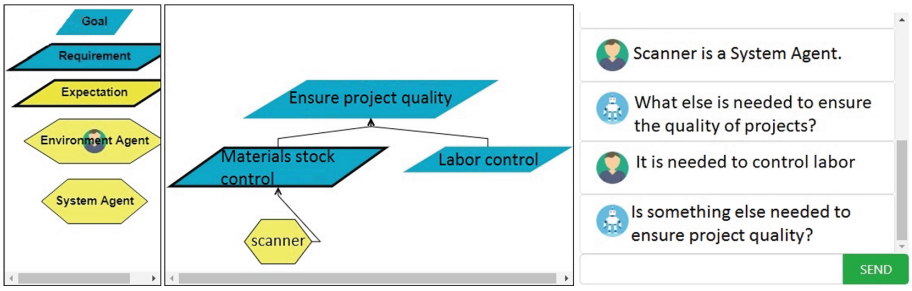



**Fig. 2.** KAOSbot interface

The KAOSbot conducts a chat flow as shown in Fig. 3. The chatbot starts by asking what the primary goal of the user is. Next, It then wants to know, at an abstract high-level, what goals should be realized for that the primary goal is satisfied. After achieving the primary goal and all its main sub goals, KAOSbot conducts a conversation in order to refine each identified subgoal and extract its requirements and expectations. KAOSbot refines the goals until a new subgoal is performed by an agent. KAOSbot then asks whether the agent responsible for the new subgoal is an Environment Agent or a System Agent. If the agent is a System Agent, then the new subgoal becomes a requirement. If the agent is an Environment Agent, then the new sub goal becomes an expectation.

### 3.3 KAOSbot Service

The KAOSbot Service is responsible for implementing syntactic patterns to extract KAOS concepts from natural language sentences.

To make this extraction, we use syntactic trees. Syntactic trees are structures generated so that the checking of syntactic patterns can be made. These structures are extracted using the Stanford CoreNLP [16], an open-source Java implementation of a probabilistic natural language parser trained on the Wall Street Journal document collection. KAOSbot uses the version 3.9.1 of the Stanford Parser.

The process of KAOS goal extraction works according to Fig. 4. First, the user's message is transformed into the syntactic tree through the Stanford CoreNLP library. Second, the parsed tree is submitted to the algorithm of identification of syntactic patterns, which verifies the pattern that fits the generated syntactic tree.

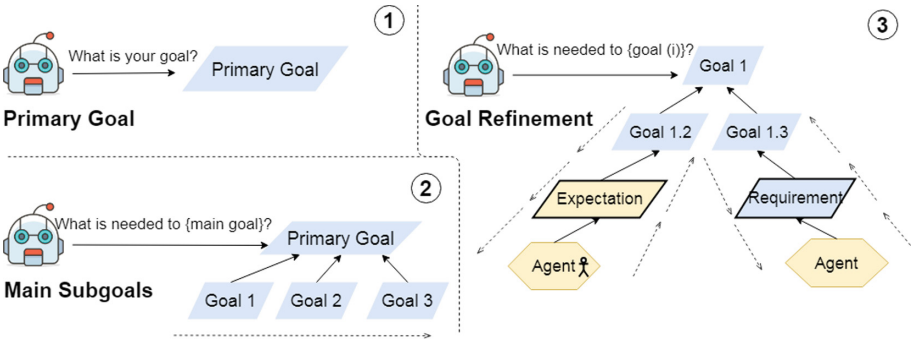



**Fig. 3.** KAOSbot conversation flow




**Fig. 4.** Processing a NL message to identify goal

The main part of the goal extraction is to identify the corresponding syntactic pattern in the parsing tree. The KAOSbot uses the following patterns: <Predicate-Object-Prepositional Phrase> and <Subject-Predicate-Object-Prepositional Phrase>. The predicate is referred to as the goal of the KAOS model, and performs the action on the entity described by the object. The prepositional phrase contains important information about the action or about the subject of the sentence. When a subject is detected in the sentence, it is considered as the Agent responsible by the Goal (i.e., the predicate). For instance, the triplet extracted from the parsing tree for the sentence "My goal is to ensure the quality of projects" is of the type <Predicate-Object-Prepositional Phrase> and has value <ensure-the quality-of projects>.

We developed the pattern-based grammatical algorithm to extract the goal-related triplets from the Stanford Parser syntactic trees. This algorithm is based on the algorithm presented in [4], which was implemented to extract <Predicate-Object-Prepositional Phrase> triplets from textual documents, extracting and modeling goals. We also used the algorithm presented in [23] to extract <Subject-Predicate-Object> triplets from text as part of a methodology that generates summaries of documents [24]. However, important improvements were made to the context of KAOSbot. For example, (i) the Prepositional Phrase was added to the <Subject-Predicate-Object> triplet because of the important information it carries about the Object of the sentence, becoming <Subject-Predicate-Object-Prepositional Phrase>; and (ii) several goals can now be extracted through the identification of connectives in the user's message. For instance, in the sentence "It is necessary to save ticket request and to show classes", the KAOSbot through

the identification of the "and" connective can to extract the following goals: "save ticket request" and "show classes". The pseudocode of the four main subroutines of the algorithm used in this paper to extract goals from syntactic trees can be seen in Algorithm 1.

**Algorithm 1.** Goal Extraction Algorithm

**Require:** $parent(tree)$ returns the parent of tree
**Require:** $getSubTreeCC(tree)$ returns the sub tree with label CC
**Require:** $highestVP(tree)$ returns the immediate tree with label VP found
**Require:** $leaves(tree)$ returns the terminal nodes of tree
**Require:** $leavesVerb(tree)$ returns the terminal nodes of tree with label in [VB,VBD,VBG,VBN,VBP,VBZ]

```
procedure GoalExtraction(parsedSentence)
    triplets ← []
    predicates ← PredicatesExtraction(parsedSentence)
    for predicate in predicates do
        treeVP ← parent(predicate)
        object ← ObjectExtraction(treeVP)
        prepositionalPhrase ← PrepositionalPhraseExtraction(treeVP)
        triplets ← triplets + (predicate, object, prepositionalPhrase)
    return triplets
procedure PredicatesExtraction(parsedTree)
    treeS ← deepestSentenceTree(parsedTree)
    treeVP ← highestVP(parsedTree)
    predicates ← leavesVerb(treeVP)
    return predicates
procedure ObjectExtraction(treeVP)
    object ← []
    for childTree in children(treeVP) do
        if label(childTree) in [NP,ADJP] then
            object ← object + leaves(childTree)
    return object
procedure PrepositionalPhraseExtraction(treeVP)
    pp ← []
    for childTree in children(treeVP) do
        if label(childTree) is PP then
            object ← object + leaves(childTree)
    return pp
```

The main function of the algorithm receives the parsing tree of the user's message as input. The algorithm identifies the predicates containing in sub trees, that is, nodes labeled with verb tags like VB, VBD, VBG, VBN, VBP or VBZ. Then, for each predicate extracted, its parent node is obtained, the treeVP. The treeVP is passed to the functions ObjectExtraction and PrepositionalExtraction in order to get both object and prepositional phrase. The output of main function GoalExtraction is the <Predicate-Object-Prepositional Phrase> triplet filled.

ObjectExtraction takes as input the treeVP subtree. The object is found by seeking in the VP subtree for all leaves of Noun Phrase (NP) or Adjective Phrase (ADJP) sub trees of the treeVP. PrepositionalPhraseExtraction also receives the treeVP as input. However, this function searches for all leaves in PP subtrees of the treeVP. Both object and prepositional phrase can be empty strings if no results are found.

PredicateExtraction identifies the deepest S node from the parsing tree. After, the highest VP tree is obtained to extract their verb leaves nodes, that is, words in the treeVP children which are tagged as VB, VBD, VBG, VBN, VBP or VBZ.

## 4 A Quasi-experiment

This section describes a **quasi-experiment** to evaluate perceived efficacy (i.e., *perceived ease of use* and *perceived utility*) for the KAOSbot tool when creating KAOS models. In addition, this experiment also evaluates the participants' intention to use the tool in the future. Thus, the research questions for our quasi-experiment are as follows:

**RQ1**: *Is the KAOSbot tool perceived as easy to use when creating KAOS models?*
**RQ2**: *Is the KAOSbot tool perceived as useful when creating KAOS models?*
**RQ3**: *Do participants intend to use the KAOSbot tool in the future?*

### 4.1 Experiment Design

This section discusses the design of the experiment to evaluate the three research questions. Following the Goal-Question-Metric (GQM) process [3], the goal of this experiment is to **analyze** the perception of the participants **for the purpose of** verifying the perceived efficacy of the tool **with respect to** the perceived ease of use, the perceived usefulness, and intention to use the tool in the future **from the point of view of** novice requirements engineers, **in the context of** undergraduate students in Computer Science.

*Context of the Experiment:* We focused our evaluation on novice requirements modelers since one of our goals is to provide an easy and useful tool that will help less experienced modelers to specify KAOS models. The participants of the experiment are **29** undergraduate students in Computer Science at the Universidade Católica de Pernambuco (UNICAP), Brazil. They have already completed the *"requirements engineering"* and *"software engineering"* disciplines and are currently in the final year of the course[1]. They were asked to accomplish this experiment as a part of a series of optional lab exercises of the course. All the students were volunteers and were aware of the practical and pedagogical purposes of the experiment, but they did not know the experimental hypotheses nor did they had previous knowledge on KAOSbot. Although we plan to perform other experiments with more experienced participants, Gorschek [10] recommends to first perform initial evaluations in lab environments.

[1] The UNICAP Computer Science course lasts at least 4 years.

*Hypotheses Formulation:* We formulated three null hypotheses, defined in a one-tailed manner, as we want to analyze the effect of the use of our tool on the variables. Each null hypothesis and its alternative are presented as follows:

**H1-0:** There is no significant perceived ease of use of the KAOSbot tool during the creation of the KAOS model/**H1-a:** The tool to create KAOS is perceived as ease of use.

**H2-0:** There is no significant perceived usefulness of the KAOSbot tool during the creation of the KAOS model /**H2-a:** The tool to create KAOS is perceived as useful.

**H3-0:** There is no significant intention to use the KAOSbot tool in the future /**H3-a:** Participants intend to use the tool in the future.

*Selected Variables and Experimental Objects:* The independent variable of interest is the use of our method with nominal values. Hence, the experiment uses only one treatment: verify how the KAOSbot tool creates KAOS models. The dependent variables are perception-based, assessing the participants' perceptions of the KAOS modeling performance using the KAOSbot tool. They are based on Technology Acceptance Model (TAM) [7], a widely applied theoretical model which uses empirical support through validations and replications [12] to analyze user acceptance and usage behavior of emerging information technologies [7]. The perceived efficacy [7] of the method can be broken down into two subjective dependent variables:

- **Perceived Ease of Use (PEOU):** Indicates the degree to which the participant believes that learning and using our tool would not require significant effort.

- **Perceived Usefulness (PU):** Indicates the degree to which the participant believes that using our tool will increase her/his job performance within an organizational context.

In addition, we evaluated the following variable:

- **Intention to use (ITU):** Indicates which the participant intends to use our tool.

These three subjective variables were measured using a 5-point Likert scale questionnaire with a set of 12 closed-questions: 6 questions for PEOU, 4 for PU, and 2 for ITU[2]. They were formulated using the opposing statement format. So, each question contains two contradictory statements representing the max and min possible values (5 and 1), where 3 is considered a neutral perception. The aggregated value is the arithmetical mean of the answers to the questions associated with each perception-based variable. We used Cronbach's alpha test [18] to evaluate the reliability of the questionnaire.

[2] The questionnaire can be found at https://bit.ly/2CN0aFd.

*Experiment Design:* The participants were randomly assigned to two different groups (each using one experimental object). Table 1 summarizes the design of the experiment. In the first two parts of the experiment, the participants watched video classes about KAOS and the KAOSbot tool. The comprehension of the videos class may also affect the perception of the tool. We alleviated the influence of this factor by selecting two representative and illustrative examples of a complexity suitable for application in the 40 min slots available for the execution of the experiments.

**Table 1.** Experiment design

| Groups | Part 1 | Part 2 | Part 3 |
|---|---|---|---|
| **A** | Introduction to KAOS | KAOSbot (Object1) | Experimental questionnaire |
| **B** | Introduction to KAOS | KAOSbot (Object2) | Experimental questionnaire |

*Analysis Procedure:* We chose statistical tests for their robustness and sensitivity to analyze the data collected. Thus, in all the tests we decided to accept a probability of 5% of committing a Type-I-Error [33], rejecting the null hypothesis when it is true. As the number of participants is not large, we tested the normality of the data distribution with the Shapiro-Wilk test [27]. To verify our hypotheses, we used the T-test to one sample when data could assume the normal distribution. However, we applied the Wilcoxon-test when the data could not assume the normal distribution.

**Table 2.** The descriptive statistics for PEOU, PU, and ITU

| Group | PEOU | | | | | |
|---|---|---|---|---|---|---|
| | Min. | Max. | Med. | Mean | Std. Dev. | p-value |
| A | 3.67 | 5 | 4.33 | 4.26 | 0.45 | 0.95 |
| B | 3 | 5 | 4.25 | 4.17 | 0.57 | 0.256 |
| A and B | 3 | 5 | 4.33 | 4.21 | 0.52 | 0.388 |
| PU | | | | | | |
| A | 3 | 5 | 4.5 | 4.37 | 0.52 | 0.393 |
| B | 2 | 5 | 4 | 3.95 | 0.85 | 0.137 |
| A and B | 2 | 5 | 4.25 | 4.14 | 0.74 | 0.007 |
| ITU | | | | | | |
| A | 3 | 5 | 4.5 | 4.30 | 0.72 | 0.012 |
| B | 2 | 5 | 4.5 | 4.25 | 0.77 | 0.008 |
| A and B | 2 | 5 | 4.5 | 4.27 | 0.73 | 0.000 |

### 4.2 Discussion of the Quasi-experiment Results

The use of multiple items to measure the same construct requires the examination of the questionnaire's reliability. We used Cronbach's alpha [18], and the result for the questionnaire was 0.862. This means that the questionnaire is reliable (Cronbach's alpha is higher than 0.7 [18]).

In order to finalize our analysis, we need to check whether, in general (the two groups together), participants perceive the tool as easy to use and useful and that participants intend to use it in the future. For this, we must consider the samples of the groups as a single sample, and check if the final mean is significantly greater than 3 (neutral value). We applied the Shapiro-Wilk test in the single sample (Union of the data of group A and B) to verify if the single sample has a normal distribution (Rows with group *"A and B"* data from Table 2).

The descriptive statistics data (mean and medians higher than 3), apparently, shows that the participants perceived KAOSbot as being easy to use, useful and they intend to use it in the future. However, we must verify this result checking the hypotheses. For this, we applied the Shapiro-Wilk test to check the normality of the distribution for all variables (column *p-value* in Table 2). The results show that PEOU has a normal distribution (PEOU $> 0.05$) and PU and ITU does not have a normal distribution (PU and ITU $< 0.05$). Thus, to finalize, we applied the one-sample T-test in PEOU and Wilcoxon-test in PU and ITU data to confirm if the mean is higher than 3 (PEOU $= 0.000$, PU $= 0.000$, and ITU $= 0.000$). As the results of the Wilcoxon-test and one-sample test were lesser than 0.05, **we can accept that KAOSbot tool was perceived as ease to use and useful, confirming hypotheses H1-a and H2-a. In addition, participants intend to use the KAOSbot tool in the future, confirming hypothesis H3-a.** This conclusion is reinforced by the positive answers obtained from the participants through the questionnaire, such as, "*By using the tool, it becomes clearer what are the goals, needs, and desires for the software in question. The tendency is for the resulting model to be closer to the desired model for the software.*"

### 4.3 Threats to Validity for the Quasi-experiment

Certain issues may threaten the validity of this experiment. About *internal validity*, the main threats are the learning effect, participants' experience, information exchange among participants, and understandability of the documents. The learning effect was mitigated by ensuring that each group of participants worked with only one experimental object. Participants' experience was not an issue as none of them had previous experience with KAOSbot tool. To minimize the information exchange among participants, they were monitored by the experimenters to avoid communication biases while performing the tasks. Understandability of the material was alleviated by performing a pilot study, with the translation of the materials to Portuguese (the participants' native language), and the removal

of any doubts during the execution of the experiment by the experimenters. Concerning *external validity*, the main threats are representativeness of the results and the size and complexity of the tasks. The representativeness of the results may be affected by the experimental objects used and the participants' context selected. About the selection of experimental objects, we mitigated this by considering two different experiment objects with a set of artifacts with similar size and complexity. Regarding size and complexity of the tasks, we used small tasks since an experiment requires participants to complete the assigned tasks in a limited amount of time. The main *construct validity* threats respect the measures applied in the data analysis and the reliability of the questionnaire. We mitigated this by using measures that are commonly used in other software engineering experiments, and the variables are based on Technology Acceptance Method (TAM) [7,12]. The reliability of the questionnaire was tested with the Cronbach test [18]. Finally, *conclusion validity* threat is the validity of the statistical tests applied. We chose the most common tests employed in empirical software engineering due to their robustness and sensitivity [17].

## 5 Related Work

There are several papers about chatbot in the literature [2,20,31]. However, to the best of our knowledge, we only know about [19] dealing with chatbot for software engineering. [19] presents a modeling bot to interpret natural language sentences to generate metamodels. Our approach elicits requirements within the context of the KAOS goal-oriented requirements engineering method. In [4], the authors do not have produced a chatbot, but they present a computational method (NLP-KAOS) that uses NLP and text mining techniques to support requirements engineers in extracting and modeling goals from textual documents. This work also employs NLP and text mining techniques for identifying goals, however our approach focuses on the interaction with novice requirements engineers to facilitate the requirements elicitation through the use of a chatbot tool.

Our approach joins the chatbot concepts used in SOCIO with the text mining algorithms used in NLP-KAOS to extract goals from text documents.

## 6 Conclusions and Future Works

In this paper, we present an approach to support goal-oriented requirements elicitation for novice requirements engineers through the use of NLP within a chatbot tool. This chatbot is named KAOSbot, because it acts as a KAOS modeling assistant. Then, we show how KAOSbot works and how it was implemented. Finally, we evaluated our approach through a quasi-experiment with respect to the perceived efficacy (i.e., perceived ease of use and perceived utility) for the KAOSbot tool when creating KAOS models. In addition, we also evaluate the participants' intention to use the KAOSbot in the future. The results show that KAOSBot tool is a promising approach for specifying KAOS models because it was perceived as easy to use, useful, and the participants intend to use it in the future.

For future works, we plan to improve our NLP algorithm to make KAOSbot cover more KAOS components such as Obstacles and Operations. Also, we plan to provide accessibility for people with motor limitations through the use of speech for the generation of KAOS models and to aggregate a machine learning system that indicates possible non-functional requirements during the conversation.

## References

1. Almisned, F., Keppens, J.: Requirements analysis: evaluating KAOS models. J. Softw. Eng. Appl. **3**(09), 869 (2010)
2. Baby, C.J., Khan, F.A., Swathi, J.: Home automation using IoT and a chatbot using natural language processing. In: 2017 Innovations in Power and Advanced Computing Technologies (i-PACT), pp. 1–6. IEEE (2017)
3. Basili, V.R., Rombach, H.D.: The tame project: towards improvement-oriented software environments. IEEE Trans. Softw. Eng. **14**(6), 758–773 (1988)
4. Casagrande, E., Woldeamlak, S., Woon, W.L., Zeineldin, H.H., Svetinovic, D.: NLP-KAOS for systems goal elicitation: smart metering system case study. IEEE Trans. Softw. Eng. **40**(10), 941–956 (2014)
5. Casamayor, A., Godoy, D., Campo, M.: Mining textual requirements to assist architectural software design: a state of the art review. Artif. Intell. Rev. **38**(3), 173–191 (2012)
6. Chowdhury, G.G.: Natural language processing. Annu. Rev. Inf. Sci. Technol. **37**(1), 51–89 (2003)
7. Davis, F.D.: Perceived usefulness, perceived ease of use, and user acceptance of information technology. MIS Q. **13**, 319–340 (1989)
8. Espada, P., Goulão, M., Araújo, J.: Measuring complexity and completeness of KAOS goal models. In: 2011 First International Workshop on Empirical Requirements Engineering (EmpiRE), pp. 29–32. IEEE (2011)
9. Gervasi, V., Zowghi, D.: Reasoning about inconsistencies in natural language requirements. ACM Trans. Softw. Eng. Methodol. (TOSEM) **14**(3), 277–330 (2005)
10. Gorschek, T., Garre, P., Larsson, S., Wohlin, C.: A model for technology transfer in practice. IEEE Softw. **23**(6), 88–95 (2006)
11. Ibrahim, M., Ahmad, R.: Class diagram extraction from textual requirements using natural language processing (NLP) techniques. In: 2010 Second International Conference on Computer Research and Development, pp. 200–204. IEEE (2010)
12. King, W.R., He, J.: A meta-analysis of the technology acceptance model. Inf. Manag. **43**(6), 740–755 (2006)
13. Kof, L.: From requirements documents to system models: a tool for interactive semi-automatic translation. In: 2010 18th IEEE International Requirements Engineering Conference (RE), pp. 391–392. IEEE (2010)
14. Kurtanović, Z., Maalej, W.: Automatically classifying functional and non-functional requirements using supervised machine learning. In: 2017 IEEE 25th International Requirements Engineering Conference (RE), pp. 490–495. IEEE (2017)
15. Luisa, M., Mariangela, F., Pierluigi, N.I.: Market research for requirements analysis using linguistic tools. Requirements Eng. **9**(1), 40–56 (2004)
16. Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S., McClosky, D.: The Stanford CoreNLP natural language processing toolkit. In: Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pp. 55–60 (2014)

17. Maxwell, K.: Applied Statistics for Software Managers. Prentice Hall, Upper Saddle River (2002)
18. Nunnally, J.C., Bernstein, I.H.: Psychometric Theory. McGraw-Hill, New York (1978)
19. Pérez-Soler, S., Guerra, E., de Lara, J.: Assisted modelling over social networks with socio. In: MODELS (Satellite Events), pp. 561–565 (2017)
20. Rahman, A., Al Mamun, A., Islam, A.: Programming challenges of chatbot: current and future prospective. In: 2017 IEEE Region 10 Humanitarian Technology Conference (R10-HTC), pp. 75–78. IEEE (2017)
21. Respect-IT: A KAOS Tutorial (2007)
22. Rolland, C., Souveyet, C., Achour, C.B.: Guiding goal modeling using scenarios. IEEE Trans. Softw. Eng. **24**(12), 1055–1071 (1998)
23. Rusu, D., Dali, L., Fortuna, B., Grobelnik, M., Mladenic, D.: Triplet extraction from sentences. In: Proceedings of the 10th International Multiconference Information Society-IS, pp. 8–12 (2007)
24. Rusu, D., Fortuna, B., Grobelnik, M., Mladenić, D.: Semantic graphs derived from triplets with application. Informatica **33**(3), 357–362 (2009)
25. Sawyer, P., Rayson, P., Cosh, K.: Shallow knowledge as an aid to deep understanding in early phase requirements engineering. IEEE Trans. Softw. Eng. **31**(11), 969–981 (2005)
26. Shah, A., Jain, B., Agrawal, B., Jain, S., Shim, S.: Problem solving chatbot for data structures. In: 2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC), pp. 184–189. IEEE (2018)
27. Shaphiro, S., Wilk, M.: An analysis of variance test for normality. Biometrika **52**(3), 591–611 (1965)
28. Souza, E., Moreira, A.: Deriving services from KAOS models. In: Proceedings of the 33rd Annual ACM Symposium on Applied Computing, pp. 1308–1315. ACM (2018)
29. Van Lamsweerde, A.: Goal-oriented requirements engineering: a guided tour. In: Proceedings Fifth IEEE International Symposium on Requirements Engineering, pp. 249–262. IEEE (2001)
30. Van Lamsweerde, A.: Goal-oriented requirements enginering: a roundtrip from research to practice [enginering read engineering]. In: Proceedings 12th IEEE International Requirements Engineering Conference, pp. 4–7. IEEE (2004)
31. Vasconcelos, M., Candello, H., Pinhanez, C., dos Santos, T.: Bottester: testing conversational systems with simulated users. In: Proceedings of the XVI Brazilian Symposium on Human Factors in Computing Systems, p. 73. ACM (2017)
32. Wanderley, F., Araujo, J.: Generating goal-oriented models from creative requirements using model driven engineering. In: 2013 International Workshop on Model-Driven Requirements Engineering (MoDRE), pp. 1–9. IEEE (2013)
33. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A.: Experimentation in Software Engineering, 1st edn. Springer-Verlag, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29044-2. http://link.springer.com/10.1007/978-3-642-29044-2