A Collaborative Approach to specify Kernel Sentences using Natural Language

Leandro Antonelli[1], Alejandro Fernandez[1 2], Nicolas Ruffolo[1],
Emiliano Sansone[1], Diego Torres[1 2 3]

[1]Lifia, Fac. de Informática, UNLP, La Plata, Bs As, Argentina
[2] Comisión de Investigaciones Cientificas (CICPBA)
[3] Departamento de Ciencia y Tecnología, UNQ

{leandro.antonelli, alejandro.fernandez, nicolas.ruffolo,
emiliano.sansone, diego.torres}@lifia.info.unlp.edu.ar

Abstract—Requirements engineering is a critical part of software development. Errors in the requirements, if not found and corrected early in the engineering process, become costly problems later on. Analysts commonly rely on Use Cases or Users Stories to capture requirements. However, there is domain knowledge that these artifacts don't capture well (for example, business rules and given-then-when scenarios). Such domain knowledge is generally distributed among multiple stakeholders and domain experts with complementing perspectives. Therefore, it is important to use a collaborative technique with a simple artifact to acquire and validate their knowledge. Kernel sentences is a linguistic definition about small sentences (with only one verb) written in active voice. Some authors relate kernel sentences to business rules. We argue that kernel sentences are adequate to use in the collaborative acquisition and they can be used as the input to produce more complex artifacts. This paper proposes a collaborative approach to acquire and validate kernel sentences. The process has three main activities: acquisition of the kernel sentences, validation of them, and assessment of the activity of the experts who participate in the activity. This paper also describes a prototype to support the process. Finally, the paper shows the result of a preliminary evaluation with promising results about the applicability of the process.

Keywords- requirements; specifications; kernel sentences; collaboration; natural language

1. Introduction

Requirements engineering is a critical stage of software development. Errors made at this stage can cost up to 200 times to repair when the software is delivered to the client [3]. There are two main philosophies of software development life cycle: classic and agile. Both strategies organize the development process and deal with the requirements in different ways. In a classic life cycle, an extensive documentation is produced and consumed. It consists for example of software requirement specification

with hundreds of Use Cases. In agile development, the communication of the requirements relies on a specific role within the team the, product owner, and on a simple documentation artifact, the User Story. User Stories have minimum information and constitute “an invitation to talk” [1].

Requirements described as Use Cases or as User Stories define the goals, the scope and the functionality of the software system. Nevertheless, software applications are “packed knowledge about the domain” [7]. This knowledge needs to be captured in a complementary artifact to Use Cases and User Stories, for example in business rules [17] or given-then-when scenarios [22].

While goals and requirements for the software application can be elicited from a small group of people (the client or the sponsor) the knowledge of the domain relies in a wider group of stakeholder, the domain experts, who generally have a different and complementary point of view of the domain. Thus, it is important to involve as many experts as possible to collaborative [14] acquire their knowledge.

Experts and development team belong to different worlds and use different languages [20]. The experts use the language of the domain while development team uses a computer science language. In order to cope with this communication gap it is important to use artifacts in natural language that are readable by both parties [14].

The concept of the kernel sentence was introduced in 1957 by linguist Z.S. Harris [12] and featured in the early work of linguist Noam Chomsky [8]. Kernel sentences are also known as basic sentences. They are declarative constructions, in active voice, always affirmative with only one verb. Boyd [4] suggests the use of Kernel Sentences to describe models in Software development.

This article proposes a collaborative process to specify Kernel Sentences. It consists of three main activities. The first activity consists in the specification of kernel sentences by the experts. The second activity consists in the validation of the kernel sentences specified in the first activity by the other experts. Finally, the third activity consists in assessing the behavior of the experts to determine how reliable their contributions are. This article also describes a prototype that can be used to support the proposed process. Finally, the paper presents some preliminary evaluation using the SUS survey [5] [6] that shows the applicability of the process.

We believe that this approach can be used in both philosophies of software development: agile and classic. The effort for performing the process is more related to classic development cycle and it can be integrated with early stages of requirements engineering, before defining the scope of the software system as well as the requirements. Nevertheless, we think that the proposed approach can also be used in agile because the prototype tool proposed can deal with the complexity and reduce the effort. Moreover, the knowledge consolidated is a good complement to User Stories.

The rest of the paper is organized in the following way. Section 2 describes some background about kernel sentences. Section 3 details our contribution namely, the proposed collaborative process. Section 4 describes the tool to support the process. Section 5 presents the preliminary evaluation. Section 6 reviews some related work. Finally, Section 7 discusses some conclusions.

2. Kernel Sentences

A kernel sentence is a simple construction with only one verb. It is also active, positive and declarative. This basic sentence does not contain any mood. It is termed as “kernel” since it is the basis upon which other more complex sentences are formed. For example, Figure 1 describes two kernel sentences. The first sentence states that the subject “farmer” performs an action (“fertilizes”) on a certain object (“tomatoes”). The second sentence has the same structure while it describes a different action (“water”) and it also adds the description about “when” the action is performed. It is important to mention that the verb “to be” does not have a semantic meaning. That is why the second example has two verbs: “to water” and “to be”. Figure 2 shows two sentences that are not kernel, since both sentences has two verbs. First sentence uses the verbs “to fertilize” and “to add”, while second sentence uses the verbs “to water” and “to prevent”. The first sentence can be rewritten into two kernel sentences (Figure 3). The farmer is the subject of the first verb, that is, “the farmer fertilizes…”. And the fertilization activity is the subject of the second action that is “the fertilization adds nutrient”. This example shows how the original sentence (Figure 2) with two verbs draws a conclusion about the role of the farmer who fertilizes and adds nutrient. Nevertheless the correct responsibilities are stated in Figure 3, that describes that the farmer fertilizes and because of this activity, the nutrients are added. This precision in the description is very important to understand the domain.

The farmer fertilizes the tomatoes
The farmer waters the tomatoes when it is hot

Figure 1. Kernel Sentences

The farmer fertilizes the tomatoes to add nutrients that are not present in the soil.
The farmer waters the tomatoes to prevent them of drying out.

Figure 2. No Kernel Sentences

The farmer fertilizes the tomatoes
The fertilization adds nutrient to the soil

Figure 3. Sentences rewritten as kernel sentences

3. The Proposed Approach

The proposed process has the objective of collaboratively obtaining kernel sentences from the domain experts in order to consolidate the knowledge of the application domain. The process is collaborative because many experts can participate at the same time. People contribute with different kernel sentences, since different stakeholders may have different point of view about the domain. And this complementary vision is very important to produce an integrated and complete description of the domain.

The process considers two different roles: experts of the domain and analysts. The experts of the domain provide the kernel sentences and validate the kernel sentences proposed by other experts. Thus, the collaborative characteristic is

reinforced. Analysts participate as moderators of the process. They have a complete vision of the set of kernel sentences provided by all the experts and the scope of the software system. Thus, analysts can identify kernel sentences that do not belong to the domain and remove them from the process. Analysts can also identify duplicated kernel sentences proposed by different experts. Finally, analysts monitor the activity of the expert by identifying experts with some particular biased behavior. For example, an expert who accepts as valid every kernel sentence, or that rejects as invalid all of them is not concerned with the activity and his contribution should be omitted since it is not reliable. The analyst not necessarily must be an expert of the domain. Moreover, we believe that he should not be an expert in order to avoid biases based on his subjectivity.

Thus, the process proposed considers three different activities: (i) kernel sentences specification, (ii) kernel sentences validation, and (iii) experts assessment. These three activities are conducted in parallel. Thus, while sentences are specified, other sentences can be validated and at the same time experts can be assessed.

The first activity (kernel sentences specification) relies on the definition of kernel sentences by some expert (who becomes its author). Then, some revision should be made to verify that the kernel sentence satisfies the conditions to be considered a kernel sentence (from the grammar perspective). Finally, some analyst should review the knowledge stated by the kernel sentence to determine if it is valuable or not for the description of the domain.

The second activity (kernel sentences validation) relies on collecting the opinion (agreement) of the experts (different from the author) about the kernel sentences of the first activity. Finally, the analyst decides whether to accept a kernel sentence as valid, based on the opinions of the experts.

The third activity (experts assessment) relies on monitoring the contributions of the experts, to identify how reliable some participant is. If a person is not reliable the analyst should exclude the person from the activity as well as his contributions. Figure 4 summarizes the whole process.

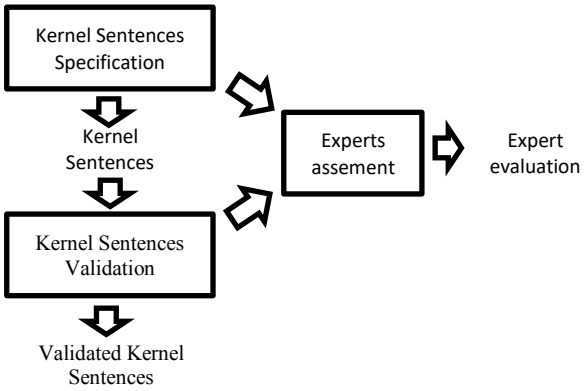

Figure 4. The proposed approach

3.1. Kernel Sentences Specification

The activity of kernel sentences specification is composed of three steps: (i) the description of the kernel sentence, (ii) the verification of its kernel sentence quality, and (iii) the revision of the kernel sentence, to confirm that it is significant for the scope of the software system.

The first step, the description of kernel sentence, is performed by some expert. The expert becomes the author of the sentence. It is important to trace every kernel sentence to its author because the other experts (different from the author) must indicate if they agree or not with the contribution. Moreover, identifying the author is also important to monitor his participation during the expert assessment activity.

The second step, the verification step, checks whether the kernel sentence is really a kernel sentence. We also propose some extra verification to make sure that the contribution of the expert is as simple and clear as possible. The following paragraphs describe the three verification steps we propose.

The first verification consists in checking that the sentence has the structure subject + verb + object to verify that it has only one verb, and that it is written in active voice. Figure 5 provides some examples of this verification.

The tomatoes are fertilized by the farmer (passive voice, not correct)
It is necessary to fertilize the tomatoes (null subject, not correct)
The farmer fertilizes and waters the tomatoes (two verbs, not correct)
The farmer fertilizes the tomatoes (correct)

Figure 5. Revision of structure subject + verb + object

The second verification consists in checking the presence of conjunctions. There are different types of conjuntions: (i) coordinating conjunctions such as ‘and’, ‘or’, ‘for’, ‘but’, etc., (ii) correlative conjunctions such as ‘not only’, ‘but also’, ‘either’, ‘neither’, etc, (iii) and subordinating conjunction such as ‘after’, ‘as long as’, ‘if only’, ‘where’, ‘according to’, etc. The presence of conjunctions does not determine that the contribution is not a kernel sentence. Nevertheless, it could provide a clue that too much information is contained in only one sentence. Figure 6 provides some examples of this revision. It is important to mention that the third sentence in Figure 6 (“The farmer assesses the humidity of the soil”) provides knowledge that is implicit in the second sentence of the same figure (“The farmer waters the tomatoes according to the humidity of the soil”). That is, “according to” means that the farmer should “assess the humidity”. Moreover, the farmer has some criteria to decide when humidity is enough and no watering is necessary.

The farmer fertilizes and waters the tomatoes (conjunction “and” to express two verbs)
The farmer waters the tomatoes according to the humidity of the soil (correct, but “according to” suggests the presence of more knowledge).
The farmer assesses the humidity of the soil (correct, it is inferred from the previous one)

Figure 6. Revision of presence of conjuctions

The third verification consists in checking the presence of adjectives and adverbs. Although their presence does not confirm that the contribution is not a kernel sentences, these types of words characterize nouns and verbs, and their presence could provide a clue that more information could be added in another kernel sentence. Figure 7 provides some examples of this revision. The first sentence uses the word "carefully" and the farmer knows what "carefully" means. Nevertheless, it is important to state explicitly its meaning. Thus, the second sentence describes that "carefully" means "waters the soil of the tomatoes" (avoiding pouring directly on the plant).

> The farmer carefully waters the tomatoes (it is a kernel sentence, but it uses the adverb "carefully", that should be describe)
> The farmer waters the soil of the tomatoes ("carefully" means avoiding pouring the water directly to the tomatoes)

Figure 7. Revision of presence of adjectives and adverbs

The third (and last) step of the activity of the kernel sentence specification consist in analyzing whether the sentence is significant for the domain or not. This analysis is performed by the analyst and since he is not an expert, his criteria could not be accurate. Nevertheless, it is important to perform some preliminary analysis of the suitability of the sentence before involving the rest of the experts in the validation. Figure 8 provides an example of this revision. The analysts know that safety of the workers is priority, that is why "the farmer uses sun protection" is important. Nevertheless it is outside of the scope of the software system. Thus, this sentence should be rejected.

> The farmer uses sun protection (this is important for the farmer health, but it is out of the boundary of the software application to develop)

Figure 8. Revision of suitability for the domain

3.2. Kernel Sentences validation

The activity of kernel sentences validation has the goal of deciding whether to accept or not the kernel sentences specified in the first activity. Thus, the accepted ones will integrate the knowledge about the domain. This activity is composed of two steps: (i) experts give their opinion about the kernel sentences specified by another author, and (ii) the analyst takes a decision about the kernel sentence (accept it or not) regarding the opinions of the experts.

The first step, the opinion of the experts, consists in providing one of three possible alternatives: "accept", if the expert considers that the kernel sentence should be accepted (because the expert agree with the author), "reject", if the expert considers that the kernel sentence should be rejected (because the expert does not agree with the author), (iii) and "don't have opinion", if the expert does not have knowledge about the statement of the kernel sentence.

The second step, deciding about the kernel sentence, consists in analyzing the opinions and, if they are conclusive, make a decision: accept or reject. If no, the analyst can wait until more opinions are collected to decide. As a suggestion, more

than half of the experts should have provided their opinion, and more than half of the opinions should agree on accept or reject.

3.3. Experts assessment

The main reason for the experts assessment activity is to monitor the contribution of the experts to identify people that are not concerned with the activity and whose contributions are not reliable. For example (i) people to contribute with junk information (because it is not finally accepted), (ii) people with a systematic tendency to contradict (that is, providing always false information), (iii) people that do not think thoroughly and automatically accept or reject everything. In our experience, we have identified two extreme behaviors. One represented by experts that specified a lot of kernel sentences, but only few of them were accepted by their colleagues. Another behavior corresponded to people that gave their opinion about accepting most of the kernel sentences, but many of them were finally rejected by their colleagues.

It is important to identify these unreliable contributors and take some actions to avoid biasing the result of the activity. The simplest measure consists in excluding the person from the activity since he is not committed with it. Considering the amount of people excluded from the activity, it could be necessary to review the kernel sentences where they were involved, since they would have been biased by unreliable opinions.

Beyond this proposed process, the assessment of the experts can be used in later stages of the requirements engineering process. For example, the information collected about the behavior of the expert can identify experts that prefer writing or validating, or people that have a lot of knowledge and their contribution are mainly accepted. For example, the ratio regarding kernel sentences validated and kernel sentences written can identify if the expert is a “writer” or a “validator”. Then, the ratio considering the kernel sentences written and kernel sentences accepted suggests that the person is an expert among the other experts. Thus, this information can be used to plan furthers stages in the requirements engineering process. For example, to discuss some particular topic with one some specific person, or to provide some information to the “validator” to obtain his feedback.

4. Tool Support

A software prototype was implemented that can be used to support the application of the proposed approach. The prototype is a web application implemented following a service-oriented architecture. The services are implemented in PHP [19] using the Symphony framework [23]. Moreover, Python [21] is also used to communicate to the SpaCy library [25] used to deal with natural language processing.

The application is responsive, that means, the interface of the application adapts itself to the device used: a computer (desktop or laptop) or mobile device (phone or tablet). Thus, the application provides a variety of platforms to be used and experts will have a wide range of possibility to contribute with knowledge acquisition.

The prototype implements two roles of users, (i) experts and (ii) analysts. All participants can work asynchronously. The experts can add contributions (kernel sentences) and validate other contributions. Figure 9 shows the interface to provide the opinion about kernel sentences written by the other experts. The analysts can verify the contributions of experts, and finally accept or reject them regarding the opinion of the experts. Figure 10 shows the interface where the kernel sentences are shown with the opinion provided by the experts about them. The percentage of "accepts" and "rejects" are displayed as well as the number of opinions provided. The analysts can also monitor the activity of the experts. Figure 11 depicts the interface that shows the contribution of some expert. It shows the kernel sentences provided as well as the kernel sentences that he provides his opinion. The interface shows the opinion of the expert and the final decision. And there are some stats about this activity. Thus, the analyst can assess the activity of the experts and identifying unreliable contributors.

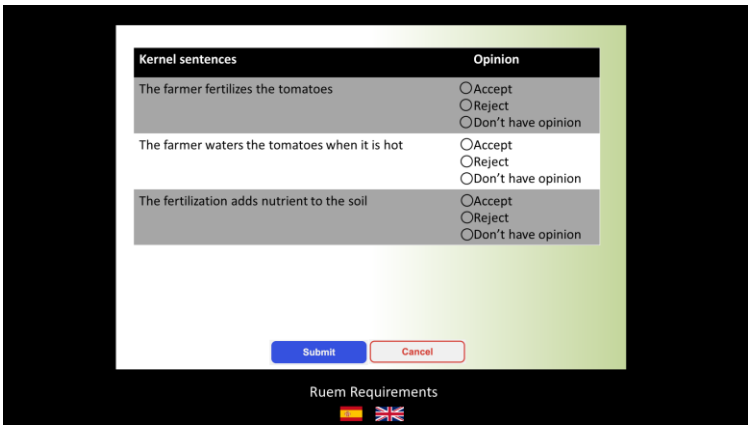

Figure 9. Interface of expert role

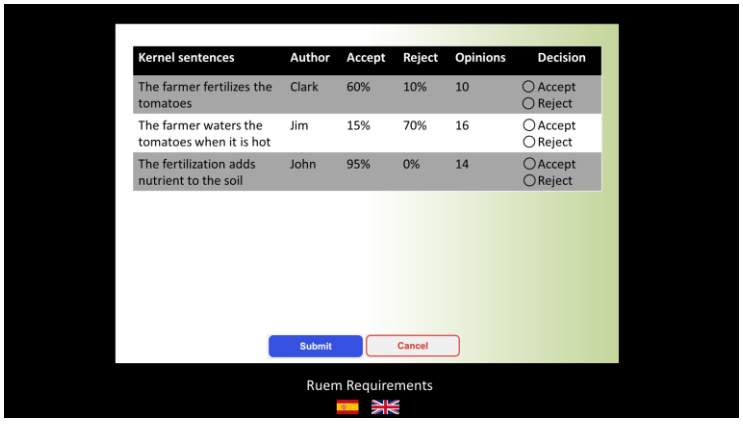

Figure 10. Interface of the analyst role

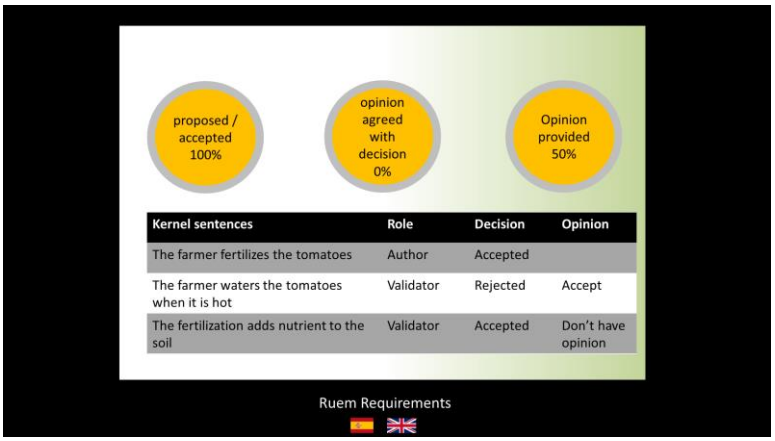

Figure 11. Interface of the experts assesment.

Thus, the complete workflow for a kernel sentence from its contribution to its finally inclusion in the data base of consolidate knowledge is the following. Some expert writes a contribution and becomes his author. The tool verifies whether the contribution is a kernel sentence or not (that is, verifies the rules described in the section of the approach).The analyst checks if the kernel sentence describes knowledge within the scope of the system. He can discard (reject) or consider that the kernel sentence is eligible (accept) to show the experts to express their opinion.

The application shows the kernel sentences accepted by analysts to the experts to ask their opinion about the correctness of the fact the kernel sentence state. Thus, the experts can answer "accept", "reject" or "don't have opinion". When the majority of experts provided their opinion, the analysts evaluate the percentages of "accept / reject / don't have opinion" to finally accept or discard the kernel sentence.

5. Evaluation

The collaborative process proposed was evaluated. The evaluation was performed using general collaborative tools like google spreadsheet instead of using the prototype tool presented in this paper, because the goal of the evaluation was to assess the applicability of the process instead of evaluating the usability of the tool. Moreover, only the main two activities were evaluated: specification and evaluation, because we trusted in the commitment of the participants, thus it was not necessary to assess the reliability of their contributions.

The participants of the evaluation were 14 students of a graduate course on requirements engineering. All of them have experience in industry, in software development. Nevertheless, the most important characteristic of them is the experience with the topic of the case study. They played the role of experts of the domain, and they had to specify and validate kernel sentences following the proposed approach. All the participants contributed to specify and validate kernel sentences for the same knowledge base. They were asked to contribute with a range between 10 and 20 kernel sentences, and they have one week to perform the task. In general, all the

kernel sentences were correctly defined, although there were many repeated contributions, because when experts contribute, they do not know the contributions of the others (unless they receive the contribuation to validate. One of the authors of this paper was the lecturer of the course and played the role of the analyst. He checked that the kernel sentences satisfied the conditions to be considered in the first activity (kernel sentences specification) and he also assigned the kernel sentences to the participants to obtain their opinion and finally accept or reject them.

The application domain used in the evaluation was the market-place application domain. All the participants had experience, as user with different roles (buyers and sellers) and some of them as developer of some application in the domain. In order to solve some ambiguities, it was defined one specific web site of the market place domain to follow their functionality.

The Systems Usability Scale (SUS) was used to evaluate the results of the case study [5] [6] in terms of the applicability of the proposed approach. Although SUS is mainly used to assess usability of software systems, it was probe to be effective to assess products and processes [2].

The System Usability Scale (SUS) consists of a 10-item questionnaire; every question must be answered in a five-options scale, ranging from ”1” (”Strongly Disagree”) to ”5” (”Strongly Agree”). Although there are 10 questions, they are related by pairs, asking the same question but in a complementary point of view in order to obtain a result of high confidence.

The calculation of the SUS score is performed in the following way. First, items 1, 3, 5, 7, and 9 are scored considering the value ranked minus 1. Then, items 2, 4, 6, 8 and 10, are scored considering 5 minus the value ranked. After that, every participant’s scores are summed up and then multiplied by 2.5 to obtain a new value ranging from 0 to 100. Finally, the average is calculated. The approach can have one of the following results: “Non acceptable” 0-64, “Acceptable” 65-84, and “Excellent” 85-100 [15]. The score obtained was 71,07. Thus, the approach can be considered as “acceptable”.

6. Related works

Garner et al. [9] state how important the human interaction is, and the associate sharing in problem solving activities as software development. Thus, our proposed approach relies on a collaborative construction and a validation of the knowledge.

Giraldo et al. [10] propose an approach to transform BPMN models to a model with more precision called CIAM. They emphasize in the importance of capturing the knowledge as early as possible, for example in early meeting as we propose.

Vijayan et al. [26] agree in the importance of eliciting domain knowledge and they propose a tool based on StakeRare [14]. Since they work in a very early phase, their method needs as input a definition of the scope of the system. Then, it builds a network of stakeholders based on recommendations. Finally, it elicits the knowledge from them. It is interesting the idea of building the network using a snowball rolling technique. Our proposed approach does not suggest how to involve the stakeholder. Nevertheless, Meng et al. [16] describes their finding in the identification of key user

for extracting knowledge in a crowdsourcing environment. They assess some characteristics that we agree: user knowledge value and willingness of knowledge exchange. Zhang et al. [28] performed a literature review about the characteristics of participants in order to perform the best selection of them. They agree with the characteristic defined by Meng et al. [16] and add more characteristics: interest, skills, expertise, willingness to achieve, and reputation. It is a very interesting contribution although it will increase the effort of applying our approach if we would like to add some of them.

Unkelos et al. [24] propose a gamified collaborative requirements engineering process. Particularly, they developed a tool to support their approach. They defined three roles that are related with our proposed roles: the creator (it is the expert in our approach), the reviewer (it is the expert in our approach since he review the knowledge) and the costumer (it is the analyst in our approach since he is going to consume the knowledge obtained). They also evaluate the participants but they do with a different objective from our evaluation. They evaluate participants in order to foster and reward their participants according to the philosophy of the gamified techniques. Kifetew et al. [13] also agree in the need for gamify collaborative requirements practices. In particular they propose a tool to prioritize requirements.

Nejad et al. [18] present a collaborative method for knowledge management in architectural design. Although the goal of their method is different from our goal, the two proposals agree in collecting, verifying and validating knowledge in collaborative way. They use a "trust" computation for validation purpose. Gonçalves et al. [11] also agree with the overall method and they also consider the importance of business rules to consolidate the knowledge of the domain. Although we use kernel sentences, some authors state their similarities [4]. Wen et al. [27] make an interesting proposal of a platform for eliciting requirements defined through 4 attributes: stakeholder, context, functional and non-functional requirements. It is similar to our approach the linking between the stakeholder and the functional requirements. And it is interesting the definition of the context. We believe that it can help to improve the description.

7. Conclusions and future work

This paper proposes a process to consolidate the knowledge of the application domain by capturing, in a collaborative way, kernel sentences directly from the experts. The proposed process consists of three activities: (i) kernel sentences specification, (ii) kernel sentences validation, and (iii) experts assessments. Two roles participate in this process, (i) experts who describe and validate the kernel sentences, and (ii) analysts who perform additional validations and take the final decision on the kernel sentences.

Although the process proposed has the objective of consolidating the knowledge of the domain, the process and the kernel sentences can be considered as a first step in a bigger strategy to manage requirements. Kernel sentences can be used in more complex artifacts of knowledge specification as well as requirements. Moreover, the assessment of the participants is useful to draw a profile of the people involved in order to plan further stages in the requirements engineering process.

The results of the case study are promising, although there is too much work to do. A full implementation of the tool with a complete assessment of both, the tool and the process, is necessary. Moreover, more experiments about range of values and percentages should be done in order to obtain precise definition of the values to accept or reject kernel sentences and define a profile of the participants. We also consider that it is very interesting to include some kind of glossary or ontology to the approach in order to deal with ambiguities for providing a more precise definition.

Acknowledgment

This paper is partially supported by funding provided by the STIC AmSud program, Project 22STIC-01.

References

1. Alexander, I. and Maiden, N.: Scenarios, Stories, Use Cases, through the system development life cycle, West Sussex: John Wiley & Sons, 2004.
2. Bangor, A., Kortum, P. T., Miller, J. T.: "An empirical evaluation of the system usability scale." Intl. Journal of Human–Computer Interaction 24.6, pp. 574-594, 2008.
3. Boehm, B.W.: Software Engineering, Computer society Press, IEEE, 1997.
4. Boyd, N. S.: “Using Natural Language in Software Development.” In: Journal of Object-Oriented Programming - Report on Object Analysis and Design, 11-9, 1999
5. Brooke, J.: “SUS-A quick and dirty usability scale” Usability evaluation in industry, 189(194), pp. 4-7, 1996.
6. Brooke, J: "SUS: a retrospective", Journal of usability studies 8.2, pp.29-40, 2013.
7. Brooks, F., The Mythical Man-Month: Essays on Software Engineering, Addison-Wesley Professional, 2 edition 1995.
8. Chomsky, N.: The Logical Structure of Linguistic Theory. Plenum Press, New York, 1975.
9. Garner, B. J.: "Collaborative knowledge management requirements for experiential learning (CKM)," Proceedings IEEE International Conference on Advanced Learning Technologies, doi: 10.1109/ICALT.2001.943989, pp. 488-489, 2001.
10. Giraldo, F., Alzate, A., Duarte, L., Tobón, M. and Hoyos, B.: "Deriving collaborative models from business process models," 2011 6th Colombian Computing Congress (CCC), doi: 10.1109/COLOMCC.2011.5936278, pp. 1-5, 2011.
11. Gonçalves, J. C. de A.R., Santoro, F. M. and Baião, F. A.: "Collaborative narratives for business rule elicitation," 2011 IEEE International Conference on Systems, Man, and Cybernetics, doi: 10.1109/ICSMC.2011.6083954, pp. 1926-1931, 2011.
12. Harris, Z. S.: Co-occurrence and transformation in linguistic structure. (Linguistic Society of America) pp. 390- 457, 1957.
13. Kifetew, F., Munante, D., Perini, A., Susi, A., Siena, A. and Busetta, P.: "DMGame: A Gamified Collaborative Requirements Prioritisation Tool," 2017 IEEE 25th International Requirements Engineering Conference (RE), doi: 10.1109/RE.2017.46, pp. 468-469, 2017.
14. Lim, S. L., Finkelstein, A.: “StakeRare: Using Social Networks and Collaborative Filtering for Large-Scale Requirements Elicitation”, IEEE transactions on software

engineering, Volume 38, Issue 3, May-Jun 2012, DOI 10.1109/TSE.2011.36, pp 707-735, 2012
15. McLellan, S., Muddimer, A., Peres, S. C.: "The effect of experience on System Usability Scale ratings." Journal of usability studies 7.2, pp. 56-67, 2012.
16. Meng, Q,. and Guo, X.: "Identification of key user knowledge source in crowdsourcing innovation mode," 2015 12th International Conference on Service Systems and Service Management (ICSSSM), doi: 10.1109/ICSSSM.2015.7170263, pp. 1-6, 2015.
17. Meservy, T. O., Zhang, C., Lee, E. T. and Dhaliwal, J.: "The Business Rules Approach and Its Effect on Software Testing," in IEEE Software, vol. 29, no. 4, doi: 10.1109/MS.2011.120, pp. 60-66, July-Aug, 2012.
18. Nejad, M. S., Moaven, S., Habibi, J. and Alidousti, R.: "Toward a collaborative method for knowledge management of software architectural decisions based on trust," 2015 12th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), doi: 10.1109/FSKD.2015.7382050, pp. 828-834, 2015.
19. PHP, https://www.php.net/, accessed: 2022-03-05
20. Potts, C.: "Using schematic scenarios to understand user needs," in Proceedings of the 1st conference on Designing interactive systems: processes, practices, methods, & techniques, 1995
21. Python, https://www.python.org/, accessed: 2022-03-05
22. Rose, S., Nagy, G.: Formulation: Document examples with Given/When/Then, Independently published, 979-8723395015, 2021.
23. Symfony, https://symfony.com/, accessed: 2022-03-05
24. Unkelos-Shpigel, N. and Hadar, I.: "Inviting everyone to play: Gamifying collaborative requirements engineering," 2015 IEEE Fifth International Workshop on Empirical Requirements Engineering (EmpiRE), doi: 10.1109/EmpiRE.2015.7431301, pp. 13-16, 2015.
25. Vasiliev, Y.: Natural Language Processing with Python and SpaCy: A Practical Introduction. No Starch Press, 2020.
26. Vijayan, J., Raju, G. and Joseph, M.: "Collaborative requirements elicitation using elicitation tool for small projects," 2016 International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES), doi: 10.1109/SCOPES.2016.7955848, pp. 340-344, 2016.
27. Wen, B., Luo, Z. and Liang, P.: "Distributed and Collaborative Requirements Elicitation Based on Social Intelligence," 2012 Ninth Web Information Systems and Applications Conference, doi: 10.1109/WISA.2012.14, pp. 127-130, 2012.
28. Zhang, X., Gong, B., Ni, H., Liang, Z. and Su, J.: "Identifying Participants' Characteristics Influencing Participant Estimation in Knowledge-Intensive Crowdsourcing," 2019 8th International Conference on Industrial Technology and Management (ICITM), doi: 10.1109/ICITM.2019.8710681, pp. 358-363, 2019.