An Iterative and Collaborative Approach to Specify Scenarios using Natural Language

Leandro Antonelli[1], Juliana Delle Ville[1], Felipe Dioguardi[1], Alejandro Fernandez[1 2], Luciana Tanevitch[1], Diego Torres[1 2 3]

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

{leandro.antonelli, juliana.delleville, felipe.dioguardi, diego.torres, alejandro.fernandez, luciana.tanevitch}@lifia.info.unlp.edu.ar

Abstract—Requirements engineering is one of the most important stages in software development. Errors committed at this stage require a lot of effort to fix in further stages. There are two main strategies to deal with software development: classic and agile. Classic software development relies on an extensive and very detailed specification, while agile development minimizes the effort on requirements specifications, relying on brief descriptions and a running prototype as a means of communication. Before the specification of requirements, there are some preliminary meetings held between the stakeholders and the IT team with the objective of discovering the goal and scope of the application to be developed. The knowledge transferred during these meetings is crucial to provide a context to requirements, no matters whether the approach is classic or agile. We argue that scenarios written in natural language are adequate artifacts to capture this knowledge. This paper proposes an iterative and collaborative approach to describe scenarios. The approach has two main activities: one of them is concerned with the description that stakeholders should perform, while the other provides guidelines to verify and possible improve the scenarios. This paper also presents a prototype tool that helps enforcing these guidelines automatically. This prototype relies on natural language processing. Finally, the paper shows the result of a preliminary evaluation of the proposed approach that indicates its results are promising

Keywords: requirements; specifications; scenarios; natural language

1 Introduction

Use Cases capture the domain and software application knowledge that is spread among many people. Some key characteristics that should be considered while describing Use Cases are: (i) they should clearly state the limit between the system and the real world, (ii) they should describe the conversation, that is, the interaction between the actor (located outside the boundaries of the software system) and the system, without providing any description of the User Interface of the software

system, and (iii) a single Use Case should describe multiple scenarios (happy path, alternative, exceptional) [13]. Workshops are the most frequent techniques to describe Use Cases [2] [26]. In a workshop, a large number of stakeholders provide their knowledge for the software professionals to understand, evaluate and organize into Use Cases. Hence, the professionals should cope with an important gap between the stakeholders and the Use Cases. It is necessary to provide some technique to deal with the amount of information incrementally, to produce preliminary artifacts until Uses Cases are obtained.

In both classic and agile development, the client should have a clear idea of the role of technology and a vision about the software artifact they need. Thus, the client should express their needs, wishes and expectations in meetings held before the definition of requirements [17]. In these meetings, the goals and boundaries of the software application must be determined. Following these definitions, the requirements can be specified using either Use Cases or User Stories.

It is indispensable to have artifacts that capture the information presented on the early meetings and allow its use as a means of communication. Clients and IT teams belong to different worlds [22]. The clients are experts in the domain, and use a specific language, while the IT teams need to understand the domain and its terminology. This situation creates a great gap of communication, and even more when knowledge is spread among several experts [16]. Thus, it is crucial to utilize an artifact that can be used by both clients and IT teams in order to consolidate the knowledge and use it as a communication means.

Scenarios are widely used artifacts. Although scenarios have many conceptions, they generally describe the dynamics (activities, tasks) to be carried out in some specific situation, which should be different to the situation and dynamics described in other scenarios. Nevertheless, multiple scenarios might still depict the same objective. This definition applies for scenarios in software engineering as well as in finance, catastrophic events, etc. [2].

Scenarios are suitable to capture knowledge because they simply tell a story, and people know how to tell stories (funny anecdotes, stories for children, etc.). This story telling approach is effective because it is a way to incorporate details that are essential to provide a rich consolidation of knowledge. Because scenarios use natural language, experts can use them without the need to learn complex formalisms. Moreover, scenarios also promote communication and cooperation when there is a wide variety of experts [8], as many of them can describe different scenarios, improve them if necessary, while learning from each other. Today software development crosses the boundaries of several application domains, hence it is important the use of an artifact that enables the collaborative sharing and building of knowledge. Nevertheless, scenarios need some structure that organizes their information to avoid redundancy, inconsistency, and ambiguity [17]. This makes the knowledge they capture more valuable to the IT team.

This article proposes an iterative and collaborative approach to describe scenarios. This approach uses a simple template consisting of 6 attributes [14] where descriptions are written in natural language. The proposed approach has two main activities. The first one consists of a set of steps that help the experts describing scenarios incrementally, while the second one presents some guidelines to verify the quality (level of detail) of the scenarios. As a result of applying these guidelines,

some improvements to the scenarios can arise. It is important to mention that these guidelines can be applied manually, but this paper also proposes a prototype tool that applies them using natural language processing. Finally, the paper also describes some preliminary evaluation using the SUS survey [6] [7] that shows the applicability and usability of the approach.

The rest of the paper is organized in the following way. Section 2 describes some background about the scenarios. Section 3 details our contribution, that is, the proposed approach. Section 4 describes the tool to support the automated application of the guidelines to the scenarios. Section 5 presents the preliminary evaluation. Section 6 reviews some related work. Finally, Section 7 discusses some conclusions.

2 Scenarios

A scenario [8] is an artifact that describes situations in a domain using natural language. It describes a specific situation that arises in a certain context in order to achieve some goal. There is a set of steps to reach that goal (episodes). In the episodes, active agents (actors) use materials, tools, and data (resources) to perform some specific action.

For example, let's consider the agricultural domain, in which it is necessary to choose a technique to water (irrigate) the plants. There can be different scenarios. One scenario can describe a situation in a rainy area, where artificial irrigation is not necessary because it will occur naturally. Another scenario can describe a situation in a dry area where rainfall is not sufficient, due to the likelihood of drought. In order to face this circumstance, and considering that there are rivers in the vicinity of the region, water should be channeled from them to irrigate. Another scenario could describe a situation where there is not enough rain and there are no rivers to channel, making it necessary to plan and implement some artificial technique to irrigate.

Thus, there are different scenarios related to irrigation, considering three different contexts: enough rain, irrigation through channelization, and artificial irrigation. All the scenarios pursue the same objective: to supply water to the crops. Nevertheless, the three scenarios are different because their contexts (the weather and the hydrology situation) are different. The work that the farmer must perform is specific for each scenario. In the first scenario (enough rain) the farmer only needs to check the humidity of the ground (as a preventive measure). In the second scenario (possible insufficiency of rainfall) the farmer needs to check the humidity of the ground and, if necessary, channelize water from the river (which implies designing and building the channel in advance) . Finally, in the last scenario (not enough rain and no river to channelize) the farmer needs to plan, implement and activate the irrigation mechanism, as he cannot rely on the rain alone.

The three scenarios describe real world situations where some software systems might be incorporated, and scenarios could be used to understand the business rules and define its scope. For example, the software system could measure the humidity or control the gate of the channel to allow the water flows. Thus, the scenarios can be used as a means of communication and as a starting point to describe the requirements (Use Cases or User Stories) of the software system to be developed [22].

Leite [14] defines a scenario with the following attributes: (i) a title; (ii) a goal or aim to be reached through the execution of the scenario; (iii) a context that sets the starting point to reach the goal; (iv) the resources, which are relevant physical objects or information that must be available; (v) the actors, who are agents that perform the actions; and (vi) the set of episodes. This template is a simple and intuitive way to organize the knowledge. In our experience, this template has the minimum set of attributes to describe a situation of the domain and a set of action to perform from it to reach a goal. At the same time, this template provides the information to describe others artifacts (more detailed) in the software development life cycle. Figure 1 summarizes the template.

Scenario title: id
Goal: objective
Context: starting point (time, place, activities previously achieved).
Actors: active agents
Resources: passive elements (tools, materials, data)
Episodes: List of actions, simple breakdown with no conditions, no iterations.

Fig. 1. Template to describe a scenario

3 Approach

The approach proposed to write scenarios is a collaborative and iterative one. It is collaborative because many people can describe scenarios at the same time. They can contribute with different scenarios, or describe the same one. Different stakeholders would have different points of view about the domain, and this complementary vision is very important to produce an integrated description of it. It is important to mention that the approach proposed considers collaboration where participants contribute adding description with no conflicts. That's mean that participants should add information to the information provided by other stakeholders with no contradiction for example. The process is iterative because although one stakeholder can entirely depict a scenario, the proposed approach organizes the description into two main activities: (i) description of the scenario and (ii) verification of the scenario. The description of the scenario (i) is performed in several steps, where every step focuses on different attributes. After the scenario is completely described, the verification (ii) activity should be performed. This verification consists in the revision of the scenarios by applying some proposed guidelines to determine whether the scenario can be improved with more detail. Therefore, the same scenario can be described and improved iteratively. The guidelines to review the scenario can be applied manually, but this paper also describe a prototype that can deal with the revision automatically. Figure 2 summarizes the approach.

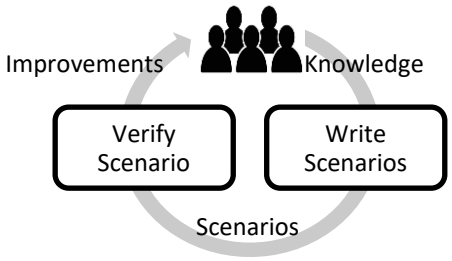

Fig. 2. The proposed approach

The rest of this section is organized in two subsections. The first subsection describes the activity of writing scenarios, while the second subsection describes the guidelines to verify the scenarios.

3.1 Process to Write Scenarios

The process proposed to write scenarios consists of an incremental approach where the title, the context and the goal of the scenario should be described first, incrementally followed by the rest of attributes until the whole scenario is completely described. The process begins with the title, the goal, and more important the context to identify the scenario. The title and the goal could be enough, but different context (starting points) can arise to obtain the same goal. Thus, it is important to start with these three attributes, and then continue with the description of the episodes as a second step. Then, third step consist in identifying actors and resources. This activity can be done independently of the episodes, but by analyzing the episodes, the identification of actors and resources will be richer. Then, a following revision of the episodes should be done considering the actors and resources looking for missing episodes related with them.
This description can be made by a single stakeholder or by different ones. The rest of this subsection describes and exemplifies the four steps proposed to describe scenarios.

Description of the title, context, and goal. The first step consists of defining the title of the scenario. Sometimes, the title is enough to define the scope of the scenario, but some other times, scenarios can be similar so that the context and the goal help to define their scope. For instance, let's consider a scenario to describe the fertilization task. This task can be carried out in different ways: fertilizer can be applied directly into the soil or it can be dissolved in water to pour the mixture into the soil. Moreover, the pouring of the mixture can be done through an irrigation pipe or manually using a backpack. Figure 3 provides the example of one scenario about fertilization using an irrigation pipe.

Scenario: fertilize using the irrigation pipe
Goal: add nutrients to the plant
Context: the cistern has enough water to activate the irrigation pipe

Fig. 3. Description of the title, context, and goal.

Description of the episodes. The episodes should be a set of steps for achieving the goal considering the context as a starting point. Let's consider the situation where fertilization is done through an irrigation pipe. The basic flow of steps consists in preparing the mixture, pouring the mixture and making the mixture flow. It is important to mention that episodes should be straightforward because if alternatives were to be considered, a different scenario should be written to describe each one. People (domain experts) generally do not express alternatives systematically. The IT team is the one that looks for all the variations and alternatives to include in the software system as business rules and nested conditional sequences. Figure 4 provides the example of the fertilizing scenario with the incorporation of the episodes.

Scenario: fertilize using the irrigation pipe
Goal: add nutrients to the plant
Context: the cistern has enough water to activate the irrigation pipe
Episodes:
The farmer dilutes the minerals in water
The farmer pours the mixture into the irrigation pipe
The farmer pours fresh water into the irrigation pipe

Fig. 4. Description of the episodes.

Identification of actors and resources. The following step consists in identifying the actors and resources from the episodes. These two attributes can be identified before describing the episodes, but oftentimes they help spotting them because they provide more information about how the scenario is performed. The example involves only one actor: the farmer. Regarding resources, there are many: the irrigation pipe, the water and the minerals. All of them are mentioned explicitly in the episodes. There is however one resource that is implicitly used in the episode: "The farmer dilutes the mineral". A chart is required to calculate the amount of minerals. Figure 5 provides the complete description of the scenario.

Scenario: fertilize using the irrigation pipe
Goal: add nutrients to the plant
Context: the cistern has enough water to activate the irrigation pipe
Actor: farmer
Resource: irrigation pipe, water, minerals, chart to calculate the amount of minerals
Episodes:
The farmer dilutes the minerals in water
The farmer pours the mixture into the irrigation pipe
The farmer pours fresh water into the irrigation pipe

Fig. 5. Identification of the actors and the resources

Improvements of the episodes. Finally, the episodes should be reviewed considering the lists of actors and resources, in order to add any missing episode. The iteration between the description of the episodes (step 3) and actors / resources (step 4) could be repeated several times, but in our experience one iteration is enough. With the identification of the new resource "chart to calculate the amount of minerals", it proves necessary to write a new first episode with the task "the farmer calculates the amount of minerals". Figure 6 describes the final scenario.

Scenario: fertilize using the irrigation pipe
Goal: add nutrients to the plant
Context: the cistern has enough water to activate the irrigation pipe
Actor: farmer
Resource: irrigation pipe, water, minerals, chart to calculate the amount of minerals
Episodes:
The farmer calculates the amount of minerals
The farmer dilutes the minerals in water
The farmer pours the mixture into the irrigation pipe
The farmer pours fresh water into the irrigation pipe

Fig. 6. Improvement of the episodes

3.2 Guidelines to Verify Scenarios

This subsection describes the guidelines proposed to verify scenarios in order to improve their detail. Sometimes, during the description of a scenario, clues appear that indicate that more information should be included in it. New information added in one scenario can trigger that experts include more information into other scenarios. This is the reason we particularly propose the following guidelines. We believe that these guidelines help to obtain in an explicit and detailed way more information about the scenarios in order to obtain a richer set of scenarios. This is not explicitly stated in our proposed approach, but it is a desired effect of the iterative and collaborative approach proposed that we evidence in our experience. The rest of the subsection describes the guidelines.

Adverbs in episodes suggest new scenarios. Adverbs describe the way in which some action is carried out. It could be obvious to the experts, but the reason to use scenarios is to explicitly capture the knowledge of the domain for non-experts to learn. Let's consider for example the episode "The farmer carefully waters the tomatoes". "Carefully" is an adverb that describes the way the tomatoes must be watered. The action can be performed in different ways for it to be rightfully qualified as "careful": (i) avoiding pouring the water directly on the plant, (ii) pouring and waiting until the soil drains it, (iii) or even by doing both things at the same time. Thus, when an adverb is identified, an analysis should be conducted to decide if a new scenario describing the whole activity ("carefully pouring") should be formed.

Conditions in episodes suggest splitting the scenario into several scenarios with different contexts. A scenario should describe in a straightforward way a set of episodes that lead from the situation described in the context to the one described in the goal. If some episode includes some expression checking some condition, it means that the context does not express the whole conditions needed for its scenario. For example, let's consider the episode: "The farmer pours the mixture into the irrigation pipe if there is enough water". The condition "There is enough water" should be added to the context (as shown in Figure 6). Moreover, it should be analyzed what happens if there is not enough water. That is why the scenario "fertilize using the irrigation pipe" should be split into two different scenarios. One including the context "The cistern has enough water", and another with the context "The cistern

has not enough water". The word "if" clearly identifies a condition as do other expressions such as "verifies", "according to", "in case of", etc.

Alternatives in episodes suggest splitting the scenario into several scenarios with different contexts. This situation is similar and complementary to the previous guideline. Some episodes could describe different alternatives to perform some step (using for example "or"), despite not having stated any condition. If there are alternatives to perform some step, it means there are some conditions (or criteria) to follow one alternative or the other. Therefore, different scenarios should be described for each alternative. For example: "The farmer pours the mixture into the irrigation pipe or into the backpack". The reason behind these alternatives relies on whether there is enough water in the irrigation pipe or not. If there is enough water, the irrigation pipe can be used. If there is not enough water, the mixture should be sprayed with a backpack in order to save water. The word "or" suggests the presence of two different scenarios, as do other words that introduce alternatives: "alternative", "choice", etc.

Speculative Expressions in Episodes suggest improving the Context of the Scenario. Speculative expressions are expressions where there is no certainty about some affirmation, and which could be bound to some condition. For example, the expression "The farmer could water the tomatoes" is speculative since there is a chance the farmer waters the tomatoes, but it is not completely ensured. There are some other modal verbs and words to express speculation: might, may, should, ought, likely, probably, etc. In order to acquire a deeper understanding behind a speculative expression, it is necessary to remove the speculation altogether and determine the conditions that state when the action will be performed. An example could be "The farmer waters the tomatoes if the humidity is under 60%", although the "if" condition is not desirable and should be added as part of the context of the scenario, in accordance with the second guideline.

4 Tool support

A software tool was prototyped to help with the application of the guidelines that verify the quality of the scenarios. The prototype[1] is a web application implemented in Python using the SpaCy library [29] to deal with natural language processing. It is important to mention that the prototype is not yet ready to use, it is only a proof of concept to show the applicability of the approach. The prototype receives scenarios in JSON format as input. The JSON is parsed with SpaCy to identify the role of every word. Thus, entities are identified, mainly from the nouns that represent actors and resources in the scenarios. Then, the relationships inside the episodes that connect these entities together are extracted and analyzed. These relationships can be obtained through verbs, because an actor (noun) performs an action over a resource (another noun). For example, "The farmer waters the tomatoes" has two entities: "farmer" and "tomatoes", and "water" describes the relationship between them. Finally, a graph is created with the nodes "farmer" and "tomatoes" linked by "water". Since the tool also has semantic support, the graph is in fact a knowledge graph, that is stored in RDF

[1] Source code available at `https://github.com/cientopolis/specguru`

format [25]. Finally, relying on the knowledge graph, the guidelines proposed can be applied using SPARQL queries [27].

Guidelines to verify scenarios are easy to implement because they consist of checking the function of some word and / or checking the word with some dictionary of specific word. Rule 1 consists in checking the presence of adverbs. Rule 2 consists in checking some expressions from a list (if, verify, according to, in case of, etc.). Rule 3 consists in checking for more than two entities in an episode. And Rule 4 consists in checking some expressions from a list (could, might, may, should, ought, etc.)

Although the tool is a prototype, it has some features that make it possible to implement new guidelines (whose relevance is currently being evaluated). The graph can be analyzed to make certain measurements. For example, the amount of relationships of a specific node. If an actor is related to many different actions, it can either mean that the actor really is a central element or of the system, or that it was incorrect to define it as only one actor, and it should in fact be split into different ones. Moreover, the graph is exported to a format commonly used by semantic platforms (like protégé [23]) that allow complex reasoning to be carried out. That is, the semantic platform can infer some conclusion by following a trace of dependencies. For example, a scenario could have its goal stated by the context of another scenario, while this second scenario has its own goal stated by the context of the first scenario. This sets a cyclic relationship between the two scenarios. That should not be right, unless some other scenario had provided the context to start this loop.

5 Evaluation

The whole process of writing and verifying scenarios using the guidelines was evaluated. The tool that helps with the application of the guidelines was not used in this evaluation. The prototype is only considered in this paper to show that the automation of the verification of scenarios is possible.

The participants of the evaluation were 14 students of a graduate course on requirements engineering. All the participants had experience in the industry of software development, and, more importantly, with the topic of the case study. All the participants played the role of domain experts, and they had to specify scenarios following the proposed approach. It is important to mention that participants have no experience in specifying scenarios. They received the training on the proposed approach during the course. All of the participants collaborate to specify a unique set of scenarios. They had to write and verify scenarios. Sometimes, they wrote their own scenarios and verified them, while some other times they collaborate in writing scenarios and providing feedback regarding the verification. A collaborative document was provided and participants had two weeks to complete their tasks.

The application domain used in the evaluation was the market place. All the participants had experience in the domain as users with different roles (buyers and sellers), and some of them as developers of some domain specific application. To

solve some ambiguities, one specific web site of the marketplace domain was selected to have its functionality followed.

The System Usability Scale (SUS) [6] [7] was used to evaluate the usability and applicability of the approach. Although SUS is mainly used to assess usability of software systems, it proved to be effective to assess products and processes as well [3].

The System Usability Scale (SUS) consists of a 10-item questionnaire; every item must be answered in a five-options scale, ranging from "1" ("Strongly Disagree") to "5" (" Strongly Agree"). Although there are 10 items, they are paired into 5 groups regarding 5 different concepts. Each question of the group focuses on a particular perspective of the concept they treat, in order to obtain highly reliable results. The items are the following:

1.- I like to use this approach more often.
2.- I find this approach to be more complicated than it should be
3.- I think the approach is simple and easy to use.
4.- I need technical support to use this approach.
5.- I find the approach functioning smoothly and is well integrated.
6.- I think there are a lot of irregularities in the approach.
7.- I think most people can learn this approach quickly.
8.- I find this approach to be time-consuming.
9.- I feel confident while using this approach.
10.- I think there are a lot of things to learn before I can start using this approach.

The calculation of the SUS score is performed as follows. 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 [19]. The score obtained by the proposed approach was 70,53. Thus, it can be considered as "acceptable".

In addition to the evaluation of the applicability and usability of the approach, the quality of the scenarios described was evaluated. The professor of the course assessed the scenarios to assure that they were correctly used regarding the philosophy of the artifact as well as the content described in accordance with the website provided. It is important to mention that is was a preliminary evaluation to assess applicability and usability of the approach. We are planning to develop a further evaluation to assess also effectiveness of the proposed approach. Moreover, we plan to include an application to assess the activity and test also de performance of the tool.

6 Related works

Several other approaches are proposed to define a process to capture knowledge and describe scenarios in natural language, emphasizing completeness. Some approaches rely on providing a visual representation of the knowledge captured by the scenarios to help experts identify any missing information and complete the scenarios. Alawairdhi et al. [1] propose an approach to derive Business Process Modeling Notation BPMN models from the scenarios, while Hassaine et al. [10] propose an approach to generate high level analysis tasks. However, they attribute the responsibility of identifying how scenarios can be improved to the experts, while in our proposed approach we provide clues about what information should be added.

Other approaches suggest what information is missing in the scenarios by generating combinations of new scenarios based on existing ones. Nevertheless, the experts still need to analyze the suggestions to identify if they are useful and realistic or not, which is critical and demands a lot of effort [12]. Makino et al. [18] present a method to generate alternative/exceptional scenarios using differential information of existing scenarios. Wu et al. [30] use a technique called “random forest” to explore alternatives systematically. Yan et al. [31] deal with a very specific situation: causal scenarios; and they also perform an automatic analysis combining the technical characteristics of the fully automatic operation (FAO) system based on the System-Theoretic Process Analysis (STPA) method. We believe that our proposed approach balances the effort and benefits because it suggests what information is missing, relying on the experts to complete it.

Other approaches are concerned with combining scenarios with some support structure to improve the description of the scenarios, and they use this extra information to analyze (manual or automatically) and suggest ways to enrich the scenarios. Bock et al. [5] use Domain Specific Language (DSL) to handle ambiguity, facilitating the application of the approach they recommend in multilingual environments. Shi et al. [26] work in a specific domain: fire management; and they define a set of primitive descriptions. Qu et al. [24] define resources used in the scenarios and relate them through these elements. Particularly, they work in a very critical and sensible domain: combat simulations, where scenarios are intertwined. Ponn et al. [21] propose a data driven approach that is also based on relating resources to each other . Torii et al. [28] relies on the rationale behind the scenarios to relate them. Our proposed approach only relies on scenarios, and we think that the extra information captured in the other proposal also demands much effort.

Finally, there are some other approaches that use scenarios, only at a different stage. We propose to use scenarios from the early meetings of software development process, nevertheless some approaches use them after requirements have been captured, or even after the software has already been coded, to document it. Biswal et al. [4] generate scenarios from UML diagrams to obtain Test Cases. Hussain et al. [11] propose a similar approach. We believe that if scenarios begin to be used from the early stages of software development, their description and the test cases will be richer. Dong et al. [9] use scenarios for decision-making support; Li et al. [15] for documenting services, as well as Kim et al. [13], who also provide a complete traceability scheme from requirements to source code. We think that these approaches

use scenarios to consolidate knowledge that can be managed directly from the source code, when they should have used scenarios from earliest stages.

7 Conclusions and future work

This paper proposed an approach to specify scenarios in an iterative and collaborative way. The approach consists of two main activities. The first activity proposes a set of steps to describe scenarios, while the second activity proposes a set of guidelines to verify the quality of the description. An evaluation was performed to show the usability and applicability of the approach. This paper also presents a software prototype that can be used to automate the application of guidelines to check the quality of the scenarios.

The results are promising, although there is still much work to be done. Capturing the knowledge from a group of people is complex, since different people can provide different point of view from the same fact or activity. The contribution proposed in this paper makes it possible to obtain a uniform and consistent description of scenarios. Nevertheless, this proposal does not deal with the difficulties of obtaining a coherent and unified description of the real world. That is, we have to work on identifying similar scenarios to merge them. This issue is even more complex because different people can use different words to express the same thing. For example, some experts can refer to tomatoes while some others can refer to vegetables. These two expressions are hyperonyms, since a tomato is a kind of vegetable (but not all vegetables are tomatoes). Semantic support is useful to deal with these problems, which is why we have included this feature in the software prototype.

Acknowledgment

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

8 References

1 Alawairdhi, M. and Aleisa, E.: "A Scenario-Based Approach for Requirements Elicitation for Software Systems Complying with the Utilization of Ubiquitous Computing Technologies," 2011 IEEE 35th Annual Computer Software and Applications Conference Workshops, doi: 10.1109/COMPSACW.2011.63, pp. 341-344, 2011

2 Alexander, I. and Maiden, N.:Scenarios, Stories, Use Cases, through the system development life cycle, West Sussex: John Wiley & Sons, 2004.

3 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.

4 Biswal, B. N., Nanda, P. and Mohapatra, D. P.: "A Novel Approach for Scenario-Based Test Case Generation," 2008 International Conference on Information Technology, pp. 244-247, 2008.

5 Bock, F., Sippl, C., Heinz, A., Lauer, C. and German, R.: "Advantageous Usage of Textual Domain-Specific Languages for Scenario-Driven Development of Automated Driving Functions," 2019 IEEE International Systems Conference (SysCon), doi: 10.1109/SYSCON.2019.8836912, pp 1-8, 2019.

6 Brooke, J.: “SUS-A quick and dirty usability scale” Usability evaluation in industry, 189(194), pp. 4-7, 1996.

7 Brooke, J: "SUS: a retrospective", Journal of usability studies 8.2, pp.29-40, 2013.

8 Carrol, J. M.: "Five reasons for scenario-based design," in Proceedings of the 32nd Annual Hawaii International Conference on Systems Sciences, 1999.

9 Dong, C. and van de Giesen, N.: "Scenario development for decision-making in water resources planning and management," 2011 International Symposium on Water Resource and Environmental Protection, pp. 928-931, 2011.

10 Hassaine, S., Dhambri, K., Sahraoui, H. and Poulin, P.: "Generating visualization-based analysis scenarios from maintenance task descriptions," 2009 5th IEEE International Workshop on Visualizing Software for Understanding and Analysis, doi: 10.1109/VISSOF.2009.5336423, pp. 41-44, 2009.

11 Hussain, A., Nadeem, A. and Ikram, M. T.: "Review on formalizing use cases and scenarios: Scenario based testing," 2015 International Conference on Emerging Technologies (ICET), pp. 1-6, 2015.

12 Jahan, M., Shakeri Hossein Abad, Z., and Far, B.: "Detecting Emergent Behaviors and Implied Scenarios in Scenario-Based Specifications: A Machine Learning Approach," 2019 IEEE/ACM 11th International Workshop on Modelling in Software Engineering (MiSE), pp. 8-14, 2019.

13 Kim, S., Kim, M., Sugumaran, V. and Park, S.: "A Scenario Based Approach for Service Identification," 2010 IEEE 34th Annual Computer Software and Applications Conference Workshops, pp. 237-238, 2010.

14 Leite, J. C. S. d. P., Rossi, G., Balaguer, F., Maiorana, V., Kaplan, G., Hadad, G. and Oliveros, A.:"Enhancing a requirements baseline with scenarios," Requirements Engineering, vol. 2, no. 4, pp. 184-198, 1997.

15 Li, Z., Tu, Z., Liu, B., Chu, D. and Li, C.: "Multi-view Scenario-based Service Resource Description Modeling and Application Method," 2021 IEEE World Congress on Services (SERVICES), doi: 10.1109/SERVICES51467.2021.00043, pp. 96-101, 2021.

16 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

17 Loucopoulos, P., Karakostas, V.: System Requirements Engineering, McGraw-Hill, Inc., New York, 1995

18 Makino, M. and Ohnishi, A.: "A Method of Scenario Generation with Differential Scenario," 2008 16th IEEE International Requirements Engineering Conference, doi: 10.1109/RE.2008.17, pp. 337-338, 2008.

19 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.

20 Nuseibeh, B.: "Requirements We Live By," 2019 IEEE 27th International Requirements Engineering Conference (RE), doi: 10.1109/RE.2019.00009, pp. 1-1, 2019.

21 Ponn, T., Breitfuß, M., Yu, X., and Diermeyer, F.: "Identification of Challenging Highway-Scenarios for the Safety Validation of Automated Vehicles Based on Real Driving Data," 2020 Fifteenth International Conference on Ecological Vehicles and Renewable Energies (EVER), pp. 1-10, 2020

22 Potts, C.: "Using schematic scenarios to understand user needs," in Proceedings of the 1st conference on Designing interactive systems: processes, practices, methods, & techniques, 1995
23 Protégé. https://protege.stanford.edu/, accessed: 2022-02-26
24 Qu, Q., Wang, Y., Yao, Y. and Qiao, S.: "Research on four problems in the scenario specification of intelligent combat simulation," 2020 IEEE International Conference on Power, Intelligent Computing and Systems (ICPICS), doi: 10.1109/ICPICS50287.2020.9202327, pp. 637-641, 2020.
25 RDF. https://www.w3.org/RDF/, accessed: 2022-02-26
26 Shi, Y., Xu, J., Zhang, H. and Yuan, Z.: "Research on Fire Scenario Primitives in Subway Station based on Object-oriented thinking," 2020 Chinese Control And Decision Conference (CCDC), pp. 2300-2304, 2020.
27 SPARQL. https://www.w3.org/TR/sparql11-query/, accessed: 2022-02-26
28 Torii, T., Umeda, Y. and Kondoh, S.: "Development of Life Cycle Scenario Description Support Tool," 2005 4th International Symposium on Environmentally Conscious Design and Inverse Manufacturing, doi: 10.1109/ECODIM.2005.1619306, pp 596-597, 2005.
29 Vasiliev, Y.: Natural Language Processing with Python and SpaCy: A Practical Introduction. No Starch Press (2020)
30 Wu, K. and Zhang, Y.: "Study on the Construction of a Structured scenario system for Conventional Emergencies based on random forest," 2020 International Conference on Robots & Intelligent System (ICRIS), doi: 10.1109/ICRIS52159.2020.00096, pp. 365-369, 2020.
31 Yan, F., Ma, J., Li, M., Niu, R. and Tang, T.: "An Automated Accident Causal Scenario Identification Method for Fully Automatic Operation System Based on STPA," in IEEE Access, vol. 9, doi: 10.1109/ACCESS.2021.3050472. pp.11051-11064, 2021.