View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Centro de Servicios en Gestión de Información

Semantic Support for Scenarios to Improve Communication in Agribusiness

Leandro Antonelli[1], Diego Torres[1,2,3], Mariángeles Hozikian[1], Jorge E. Hernandez[4]

[1] Lifia – Facultad de Informatica, Universidad Nacional de La Plata, Argentina
[2] CICPBA – Comision de Investigaciones Cientificas de la Provincia de BsAs, Argentina
[3] Departamento de Ciencia y Tecnologia, Universidad de Nacional de Quilmes, Argentina
[4] School of Management, University of Liverpool, United Kingdom
{leandro.antonelli, diego.torres, marian.hozikian}@lifia.info.unlp.edu.ar
J.E.Hernandez@liverpool.ac.uk

Abstract. Organizations produce and exchange a lot of critical information to obtain good results. Systems developed in different domains are adapted to be integrated when they need to exchange information. Food and agribusiness are not the exceptions, they are pioneers in the use of massive data and collaboration. One of the biggest challenges to communicate software systems is analyzing their colliding context. Every software system relies on its own context, with its rules, dynamic, and language. It is a big effort to have a complete understanding of the composed domain. Scenarios are well-known tools to describe domains and are commonly described with text. When Scenarios are built by different stakeholders it is extremely important to review them in order to unify their description. Thus, the improved Scenarios with a unified point of view make it possible an analysis to identify the relationship between two different domains. This analysis is the key to design a mechanism to exchange information. This paper proposes a semantic definition of Scenarios and a set of queries to identify issues in the Scenarios and improve their quality. We also provide a wiki platform to implement the semantic support and the queries.

Keywords: Agribusiness, Requirements, Scenarios, Ontologies

1 Introduction

Nowadays, there is a huge level of integration between different software systems. Everyone produces a big amount of data and different organizations share this information to improve their results [5]. Collaboration is needed in every sector. Food and agribusiness are not an exception. Their supply chains are pioneers in the use of massive data, sometimes due to rigorous legislation that force to trace lots of variables along the supply chain [14]. Scenarios are well-known tools to describe situations of the domain [2]. They can be used to capture the context of different applications to identify their relationship. Thus, it is possible to establish a mechanism to make the applications to exchange information.

Nevertheless, it is not an easy task to design a mechanism to interoperate two different applications already developed [5]. Every software system relies on its own

context, with its rules, dynamic, and language. Scenarios should use the language of the stakeholders since the stakeholders are the ones that describe them. Thus, Scenarios need to be described with narrative text [7]. But, it can be hard to identify joints points in Scenarios that are described by two different groups of stakeholders that belong to two different contexts [11].

There are some quality attributes that good specification must satisfy: completeness, consistency, unambiguity, and correctness [6]. Completeness means that no piece of a specification can be missed, because some absence can lead to suppositions. Consistency means that the different points of view should provide a unified description. Unambiguous is related to the use of terms and expressions that should be carefully chosen in order to avoid misunderstanding. Finally, correctness is related to assure that the description satisfy the reality. That is, there is no gap between the intended meaning and the specification.

Scenarios are used to understand the context of the application since they promote communication when there is a great variety of experts [2] [10]. Scenarios should be written carefully in order to satisfy the quality attributes. Nevertheless, it is very difficult to achieve this goal [12]. Scenarios have been historically described by only one person, the requirements engineer who elicited the knowledge, organized it and produced a homogenous specification [7]. This classical view is being replaced by a collaborative model, where every stakeholder contributes directly to the specification [4]. Let's consider the expression "cultural labor". In the agricultural domain, it refers to some task (labor) to take care of the plants (cultures). Nevertheless, the expression can also refers to some artistic (cultural) activity (labor).

A semantic support helps to improve the quality of narrative descriptions [3]. An ontology description is a semantic mechanism that relates every relevant syntactic element (for example nouns and verbs) to a semantic element [13]. For example a homonym could be related to two different ontology elements. Thus consistency and unambiguity can be improved [15][1]. Moreover, ontologies can be described in semantic tools that make possible automatic processing to infer conclusions. For example, considers the following sentences: "A tomato is a vegetable" and "Any vegetable needs irrigation". A semantic query can conclude that "A tomato needs irrigation".

In this paper, we propose a semantic description of the Scenarios, a set of semantic queries, and a tool support for them. This contribution provides an automatic processing of the Scenarios to help to improve their quality regarding consistency, ambiguity, completeness and correctness. The proposal identifies issues in the description of the Scenarios. Thus, the stakeholders can improve their shared knowledge and consolidate it in the Scenarios. This knowledge makes possible the analysis of the colliding areas captured in Scenarios to design an interoperation mechanism. This paper only focuses on identifying issues to improve the quality of the Scenarios. Nevertheless, this is an important step to design an interoperation mechanism.

The rest of the paper is organized in the following way. Section 2 describes the template of the Scenario. Section 3 presents the semantic definition. Section 4 proposes semantic queries to identify issues. Section 5 describes the tool. Section 6 discusses some conclusions.

2 Scenario template

Leite [7] defines a Scenario with the following attributes: (i) a title that identifies the Scenario; (ii) a goal to be reached through the execution of the episodes; (iii) a context that sets the starting point to reach the goal; (iv) the resources, relevant physical objects or information that must be available, (v) the actors, agents that perform the actions, and (vi) the set of episodes, smaller task (that could also be described as a Scenario) to accomplish the goal

Listing 1 and 2 provides examples. The domain used is a farm that grows vegetables, but it also breeds animals in order to be ecologically self-sufficient as well as profitable. The goat milking Scenario (Listing 1), describes some basic steps to obtain milk from the goats. The actors and resources attributes should be used in the episodes, although it is possible, that episodes mention actors and resources not mentioned in these both attributes due to the iterative construction of the Scenarios. That is, in a first step, some stakeholder identifies a Scenario describing its title, then other stakeholders describe the main actors and resources, and finally some other with more knowledge describes the set of episodes. The Cheesemaking Scenario (Listing 2) is related to the Goat milking Scenario because the milk obtained with the first Scenario is used to produce cheese. This relation is showed in the context of the Scenario Cheesemaking and the goal of the Scenario Goat Milking.

Scenario: Goat Milking
Goal: The goat milk is stored in a refrigerated tank.
Context: Goats located at the extraction facility
Resources: goats, refrigerated tanks, milking machine
Actors: farmer
Episodes:
The farmer sets the goat in the milking machine
The farmer extracts milk with the milking machine.
The farmer conducts the milk to a refrigerated tank.

Listing 1. Goat milking Scenario

Scenario: Cheesemaking
Goal: To have cheese to sell and obtain money to run the farm
Context: The goat milk is stored in a refrigerated tank.
Resources: Milk
Actors: Cheesemaker
Episodes:
The cheesmaker curdles the milk with lactic ferments
The cheesmaker adds rennet to the milk
The cheesmaker drains the milk in mussels
The cheesmaker salts the milk
The cheesmaker leaves the milk to refine for 24 hours

Listing 2. Cheesemaking Scenario

3 Semantic definition of the Scenarios

This section describes the ontology designed for providing a semantic description of the Scenarios. Using the proposed ontology, stakeholder can keep using an iterative and incremental approach to describe the Scenarios, but the ontology will provide a support to identify inconsistencies.

The description uses the main principles of the OWL language [8]. We defined six main semantic concepts that are described as classes. The first one is the Scenario. Then, some attributes of the Scenario are also classes: Actor, Resource, and Episodes. Finally, there are two different attributes (Goal and Context) that are described with the same class: Condition. Each of the class concepts has the following intent:

Scenario: It is the core conceptualization in the ontology. It has a title, a data property defined as a string. Additionally, Scenario includes a Goal and a Context, both of them are Conditions. The Context is the pre-condition to perform the Scenario while the Goal is the postcondition. The Scenario also contains actor, resources, and episodes, steps that could be atomic actions represented by Episodes, or more complex ones, described as Scenarios.

Condition: It represents a situation, and it is used to describe goals (the desired situation to achieve) and context (needed situation to allow the execution of the Scenario).

Actor: It represents the subject that is in charge of the Episodes actions and the owner of the scenarios.

Resource: It represents the resources that are used in the episodes by the actors.

Episode: It represents each task that the actor performs with some resource. Thus, the episode is related to an actor, a resource and a verb. Moreover, the episode is related to a prior episode that must be completed.

Action: It represents the main action of an episode. It is important to mention, that the semantic representation of the action not only consider a verb, in fact, it could be a more complex expression that provide an accurate description of the domain.

Figure 1 shows the different classes and the dependencies between them. The figure uses the Scenarios described in Listing 1 and 2. The Scenario Goat Milking (Listing 1) is completely described, while the figure only describes the elements of the Scenario Cheesemaking (Listing 2) that are related to the first one. That's the case of the condition "The goat milk is stored in a refrigerator tank", shared as a goal and a context. Then, it is important to mention that the actions are complex expression, for example: "conducts the milk to", instead of referring only to a verb.

4 Semantic queries to identify issues

This section describes the semantic artifacts that allow the stakeholders to check requirements quality attributes as completeness and consistency. These queries should be checked constantly along the collaborative definition of Scenarios. Thus, when some issue is identified, an alert is shown explaining the issue, so that it can be fixed. The rest of this section describes five semantic queries in a conceptual way and we also present a SPARQL query definition.

Query 1. Consistency between actors and episodes

All the actors included in the attribute actor of the scenario should be mentioned in at least one of the episodes. That is, if an actor a belongs to the scenario s, there should be an episode (or scenario which is an episode of s) that refers the actor a. Because of the iterative and incremental description of the Scenarios, it is not necessary to check that all the actor mentioned in the episodes should be listed in the attribute actor. The SPARQL query detailed in Listing 3 shows the list of actors that are inconsistent for the <scenario>. If the query returns an empty list, it represents the lack of actors and episodes inconsistency.

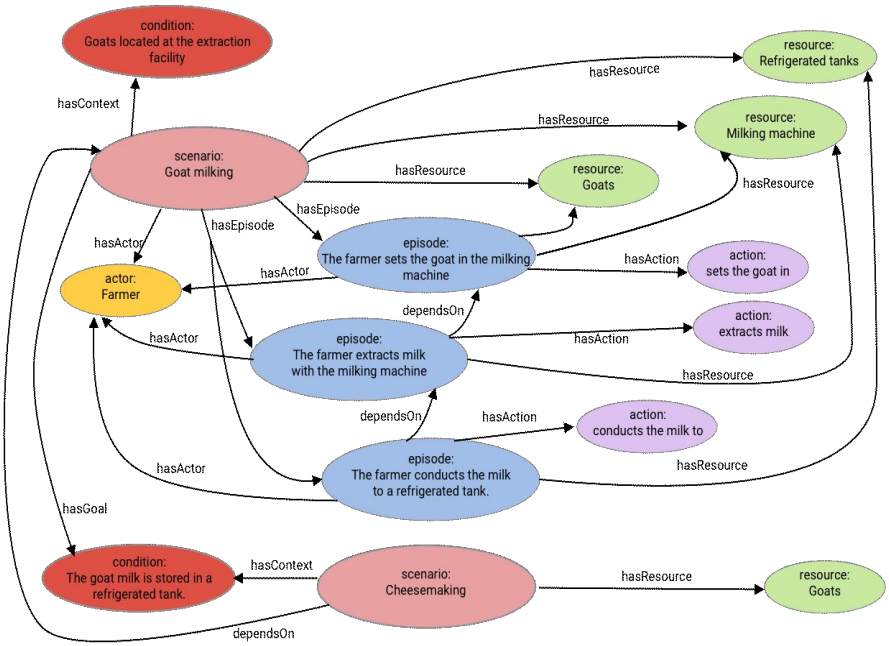

Figure 1. Classes and dependencies

```
1. SELECT ?actors WHERE {
2. {<scenario> hasActor ?actor}
3. MINUS{
4.   <scenario> hasEpisode ?episode.
5.   ?episode hasActor ?actor.}}
```

Listing 3. SPARQL query to detect inconsistency between actors and episodes.

For example, Listing 4 shows a new version of the Cheesemaking Scenario (partially described) that has actors and episodes inconsistency because the actor farmer is not mentioned in any episode. The query applied to the example will return a list with farmer.

Scenario: Cheesemaking
Actors: farmer
Episodes:
The cheesemaker curdles the milk with lactic ferments
The cheesemaker adds rennet to the milk

Listing 4. A scenario with inconsistency between actors and episodes

Query 2. Consistency between resources and episodes

All the resources included in the attribute resource of the scenario should be mentioned in at least one of the episodes. That is, if an resource r belongs to the scenario s, there should be an episode (or scenario which is an episode of s) that refers the resource s. This query is similar to the previous one. The SPARQL query detailed in Listing 5 shows the list of resources that are inconsistent for the <scenario>. If the

query returns an empty list, it represents the lack of resources and episodes inconsistency. For example, Listing 6 shows a new version of the Goat Milking Scenario that has resources and episodes inconsistency because the resource horses is not mentioned in any episode. The query applied to the example will return a list with horses.

```
1. SELECT ?resources WHERE {
2. <scenario> hasResource ?resource}
3. MINUS{
4.   <scenario> hasEpisode ?episode.
5.   ?episode hasResource ?resource.}}
```

Listing 5. SPARQL query to detect inconsistency between resources and episodes.

Scenario: Goat Milking
Resources: goats, refrigerated tanks, milking machine, horses
Episodes:
The farmer sets the goat in the milking machine
The farmer extracts milk with the milking machine.
The farmer conducts the milk to a refrigerated tank.

Listing 6. A scenario with inconsistency with a resource

Query 3. Completeness with the satisfaction of contexts by goals

A scenario s can be performed if all its conditions described in the context attribute are contained in the union of the conditions described in the goal of other scenarios. Goals describe the intended situation (final states, postconditions) while contexts describe the starting point situations (initial states, preconditions). Thus, the context of a Scenario should be satisfied with the goals of other Scenarios, in order to be performed. The SPARQL query detailed in Listing 7 shows the list of conditions (contexts) for the <scenario> that are not satisfied by any other Scenario. If the query returns an empty list, it represents the completeness of satisfaction. For example, Listing 2 describes the Cheesemaking Scenario, where its context is satisfied by the goal of the Goat Milking Scenario described in Listing 1. Nevertheless, the context of the Goat Milking Scenario, is not satisfied with the goal of Cheesemaking Scenario.

```
1. SELECT ?contextCondition WHERE {
2. <scenario> hascontext ?contextCondition.}
3. MINUS{
4.   ?otherScenario a Scenario.
5.   ?otherScenario hasGoal ?contextCondition.}}
```

Listing 7. SPARQL query to detect context and goals completeness.

Query 4. Consistency in the sequence of the Scenarios

A scenario s can be performed if all its conditions described in the context attribute are contained in the union of the conditions described in the goal of the depending on scenarios. This query is a complement of the previous query that only checks if some goal can satisfy a context, while this query checks that a previous Scenario is the one that should satisfy the goal. The SPARQL query detailed in Listing 8 shows the list of

conditions (contexts) for the <scenario> that are not satisfied by any depending on Scenario. For example, Figure 1 shows a dependency between Cheesemaking Scenario on Goat milking Scenario. This dependency is based on some stakeholders who stated that Goat milking should be done first and after that can be done Cheesemaking. Considering this dependency, this query tests if the Cheesemaking Scenario context is satisfied by the goal of the Goat Milking Scenario described.

```
1. SELECT ?contextCondition WHERE {
2. <scenario> hascontext ?contextCondition.}
3. MINUS{
4.   ?otherScenario a Scenario.
6.   <scenario> dependsOn ?otherScenario.
7.   ?otherScenario hasGoal ?contextCondition.}}
```

Listing 8. SPARQL query to detect consistency in the sequence of the Scenarios.

Query 5. Completeness in the redundancy of goals

Some scenarios s1 and s2 has the same goal, thus, they should be refined in order to have different and specifics goals. When a group of stakeholders are collaboratively describing Scenarios, it is difficult that all of them have a complete understanding of the whole domain. Thus, when Scenarios with duplicated goals are identified it means that two overlapping scenarios are described. The SPARQL query detailed in Listing 9 shows the list of Scenarios that has a duplicated goal with the <scenario>. For example, Listing 10 and 11 shows a new version of the Goat milking and Cheesemaking Scenarios. These new version has the same goal because both scenarios are overlapped. The last episode of the Goat Milking Scenario overlaps Cheesemaking Scenario, and the first episode of the Cheesemaking Scenario overlaps with the GoatMilking Scenario.

```
1.SELECT ?scenarios ?goal WHERE {
2.   ?scenario a Scenario.
3.   <current> hasGoal ?goal.
4.   ?scenario hasGoal ?goal.
5.   FILTER(?scenario <> <current>).}
```

Listing 9. SPARQL query to detect redundancy of goals.

Scenario: Goat Milking
Goal: Obtain cheese from the goats
Episodes:
The farmer sets the goat in the milking machine
The farmer extracts milk with the milking machine.
The farmer conducts the milk to a refrigerated tank.
The cheesemaker producer makes cheese.

Listing 10. Goat milking Scenario overlapped with Cheesemaking Scenario

Scenario: Cheesemaking
Goal: Obtain cheese from the goats
Episodes:
The farmer do goat milking.
The cheesmaker curdles the milk with lactic ferments
The cheesmaker adds rennet to the milk
The cheesmaker drains the milk in mussels
The cheesmaker salts the milk
The cheesmaker leaves the milk to refine for 24 hours

Listing 11. Cheesemaking Scenario overlapped with Goat milking Scenario

5 Tool support

We developed a Media Wiki [9] based application to support the semantic representation of the Scenarios and the queries to identify issues. Media Wiki is an open source implementation written in PHP that uses the MySql database engine. Wikipedia and other projects of Wikimedia use Media Wiki.We have added two extensions: (i) an ad-hoc collaborative catalog and editor, and (ii) a semantic Media Wiki. Since it relies on the wikitext format, users with no knowledge of HTML or CSS can easily edit the pages and the result looks like web pages that users are familiar to. Media Wiki stores in a database all the different versions of each page, in a collaborative environment could be necessary to access a previous version. Another advantage of Media Wiki is the management of the links between pages. Although the destination of a link does not exist, the link can also be written and Media Wiki shows it anyway, when the user clicks the link, Media Wiki allow to create the page. It is an useful feature to connect Scenarios while they are being described. Figure 2 shows a screenshot of the Goat Milking Scenario with some report about an inconsistency detected with actors.

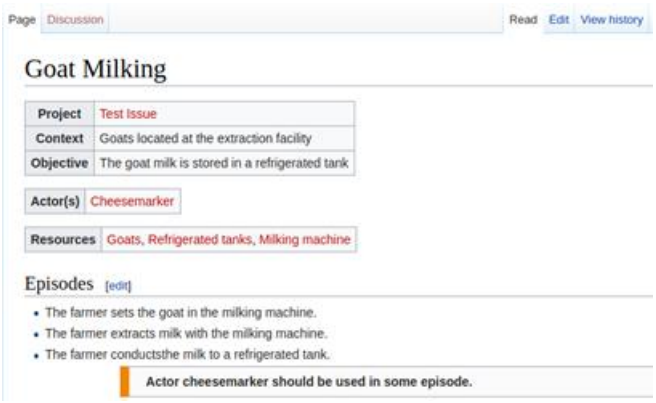

Figure 2. Goat milking Scenario without any reported issue

6 Conclusions

We have presented a semantic description of Scenarios and a set of semantic queries, both things implemented in a semantic Media Wiki in order to support the collaborative description of Scenario. This proposal contributes to identify issues that arise because of the collaborative nature of the construction. Moreover, the agricultural domain is very specific because practices varies between different regions as well as their language. Thus, in order to communicate and interoperate different software systems, it is necessary to unify the knowledge of the different domains. We claim that our proposal provides an approach to capture the knowledge from different stakeholders and obtain a shared knowledge through a iterative and incremental process of checking and improving. This work is supported by the RUC APS project, in which three different groups of teams participate: IT experts, agricultural engineers

and business specialist. We are using Scenarios and preliminary results are satisfactory.

Acknowledgement

This research is supported by Agroknowledge and Ruc-Aps, a H2020 RISE-2015 project, aiming at Enhancing and implementing Knowledge based ICT solutions within high Risk and Uncertain Conditions for Agriculture Production Systems.

References

[1] Bhatia, M.P.S., Kumar, A., Beniwal, R,: Ontology based framework for detecting ambiguities in software requirements specification. In 3rd INDIACom, New Delhi. pp. 3572-3575. (2016)
[2] Carroll, J.M.: Five reasons for scenario-based design. In Interacting with computers 13.1. doi: 10.1016/S0953-5438(00)00023-0. pp 43-60. (2000)
[3] Dzung, D.V., Ohnishi, A.: Improvement of Quality of Software Requirements with Requirements Ontology. In 9th ICQS, Jeju, doi: 10.1109/QSIC.2009.44. pp. 284-289. (2009)
[4] Ge, C., Yu, S., Yang, G., Wang, W,: A collaborative requirements elicitation approach based on scenario. In 10th International Conference on Computer-Aided Industrial Design & Conceptual Design, Wenzhou. doi: 10.1109/CAIDCD.2009.5375171. pp. 2213-2216. (2009)
[5] Ilyas, M., Khan, S.U.: An empirical investigation of the software integration success factors in GSD environment. In 15th SERA, London, pp. 255-262. (2017)
[6] Kummler, P.S., Vernisse, L., Fromm, H.: How Good are My Requirements?: A New Perspective on the Quality Measurement of Textual Requirements. In 11th QUATIC, Coimbra. doi: 10.1109/QUATIC.2018.00031. pp. 156-159. (2018).
[7] Leite, J.C.S.dP., Rossi, G., Balaguer, F., Maiorana, V., Kaplan, G., Hadad, G., Oliveros, A.: Enhancing a requirements baseline with scenarios. In requirements Engineering. Vol 2.4. doi: 10.1109/ISRE.1997.566841. pp 184-198. (1997)
[8] McGuinness, D. L., Van Harmelen, F.: OWL web ontology language overview. W3C Recommendation, 10(10), 2004.
[9] Media Wiki, https://www.mediawiki.org
[10] Potts, C.: Using schematic scenarios to understand user needs. In 1st conference on Designing interactive systems: processes, practices, methods, & techniques. (1995)
[11] Ramasubbu, N., Kemerer, C.F.: Managing Technical Debt in Enterprise Software Packages. In IEEE Transactions on Software Engineering, 40-8. pp. 758-772. (2014).
[12] Sarmiento, E., Leite, J.C.S.d.P., Almentero, E.: Using correctness, consistency, and completeness patterns for automated scenarios verification. In 5th RePa. pp. 47-54. (2015)
[13] Shunxin, L., and S. Leijun, S.: Requirements Engineering Based on Domain Ontology. In 2010 International Conference of Information Science and Management Engineering, Xi'an. doi: 10.1109/ISME.2010.110. pp. 120-122. (2010)
[14] Tan, L., Haley, R., Wortman, R., Zhang, Q.: An extensible and integrated software architecture for data analysis and visualization in precision agriculture. In 13th IRI, Las Vegas, NV. doi: 10.1109/IRI.2012.6303020. pp. 271-278. (2012)
[15] Zait, F., Zarour, N.: Addressing Lexical and Semantic Ambiguity in Natural Language Requirements. In Fifth ISIICT, Amman. doi: 10.1109/ISIICT.2018.8613726. pp. 1-7. (2018)