Conflict Management in the Collaborative Description of a Domain Language

Claudia Litvak
DIIT
Universidad Nacional de La Matanza
La Matanza, Bs. As., Argentina
clitvak@unlam.edu.ar

Gustavo Rossi[1], Leandro Antonelli
Lifia, Facultad de Informatica
Universidad Nacional de La Plata
La Plata, Argentina
{gustavo, lanto}@lifia.info.unlp.edu.ar
[1]also CONICET

***Abstract*—The identification and specification of the requirements of a software system is a difficult task that has the goal of obtaining requirements as correct and complete as possible. It is extremely important that Requirements Engineers understand a domain language in order to write high-quality requirements. Moreover, they must describe (and discuss) the language in a collaborative way in order to consider the different points of view of all stakeholders to assure that the resulting requirements will have more chances to meet their needs. However, collaborative construction implies the occurrence of conflicts that are unavoidable because of ambiguity, overlapping and misunderstanding natural language descriptions. This article relies on the Language Extended Lexicon in order to describe the application domain. Although it is a semi-structured glossary and this characteristic helps to reduce the conflicts, our experience shows that conflicts arise anyway. Thus, in order to mitigate this problem, this article presents a catalogue with a set of conflicts that could appear during a collaborative construction of the Language Extended Lexicon and proposes alternatives for their resolution.**

Keywords-requirements engineering, collaboration, conflicts, natural language models

I. INTRODUCTION

Requirements Engineering is one of the initial stages of the Software Development Life Cycle. The goal of this stage is to acquire the knowledge and the requirements needed for the system to be built. Errors made in requirements specifications have a great impact towards the end of software development, since the cost of error correction increases as each stage progresses [1].

Several authors argue that the interaction of different stakeholders working collaboratively on the same problem improves the quality of the system requirements [2] [3]. Since different stakeholders have different concerns and different point of view, all of them working together will produce a richer model.

However, generating models collaboratively implies the emergence of conflicts that must be solved in order to build a consistent high quality model. The existence of a conflict is not a negative situation, in fact it might be positive since it provides the possibility of improving the models, analyzing and discussing the different ideas observed and manifested by the conflict.

In this context, it is even more important, to define a basic language in order to interact and describe the needed models. There are two main kinds of languages: formal and natural language. Despite the introduction of ambiguity, the natural language has the advantage to be understood by all the stakeholders (technical and non technical).

Ambiguity means having two interpretations for the same word. For example, let's consider that the word "label" has two different meanings: (i) "It is the action of putting the brand of the product on the boxes of finished product"; and (ii) "It is the action of marking the price of each finished box of finished product". Imagine a situation where two stakeholders use the same word with different meaning: they would think they understand each other, but in fact, they want to transmit a different idea. An opposite situation could be the use of two different words, which in fact are synonyms and represent the same idea. In this case, both stakeholders can not know that they are talking about the same thing.

Our research is framed by the Language Extended Lexicon (LEL). The LEL is a model that uses Natural Language [4] to describe the vocabulary of the application domain. The LEL is a very convenient tool for stakeholders with no technical skills, although people with such skills will profit more from its use [5]. In particular, the convenience of the LEL as a tool arises from three significant characteristics: it is easy to learn, it is easy to use and it has good expressiveness. Goel [6] states that the LEL is widely used to capture the language to describe requirements. Moreover, it is a useful technique because can be understood by the stakeholders, and this characteristic encourage their active participation which is crucial in first steps of software development.

The LEL captures the terms (they are called symbols) and describes them with the name, the notion, and the behavioral responses. The name identifies the symbol; all synonyms that exist in the domain must be defined in this attribute. The notion describes the meaning (denotation) and the behavioral responses describe the relation of the symbol with other

DOI reference number: 10.18293/SEKE2018-106

symbols (connotation). Every LEL symbol belongs to one of four categories: Subject, Object, Verb, and State.

Antonelli [7] outlines a strategy to describe the LEL in a collaborative way. However, it is very difficult to produce a domain language specification when there are many actors involved [8]. In a collaborative context, all participants build a joint model, and as previously explained conflicts might emerge between the different viewpoints.

This paper presents an approach for the identification and resolution of conflicts that emerge when the LEL is developed collaboratively. The collaborative construction of the LEL means that different stakeholders propose symbols and provides definitions in an iterative way. This means that different people collaborate by making specific contributions: identifying the symbol that must be defined, or adding a definition. Nevertheless, in this context, it is necessary to have a full understanding of all the definitions. Our proposed approach consists in analyzing the whole glossary looking for conflicts and providing a solution for each conflict.

The paper is organized as follows: Section II provides the related work; Section III presents the conflicts, the proposed solutions, and a preliminary evaluation; finally, Section IV sets out the conclusions and future work.

II. RELATED WORK

Different authors have studied the existence of conflicts in Requirements Engineering [9]. Literature covers a wide range of conflict types and stages of the requirement phase where conflicts can appear [10]. Bendjenna [11] states the importance of dealing with conflictive situations during Requirements Engineering, considering the variety of stakeholders with the common objective of obtaining a unique system. Aldekhail [12] presents a literature review related to requirements conflicts. Some publications have presented requirements conflict management in a web-based collaborative environment. The SOP project [13] has developed a wiki using the Volere Requirements Specification Template [14], seeking to pinpoint inconsistencies in requirements documents created with their tool. WikiWinWin [15] is a wiki front-end to the WinWin tool. Urbieta [16] presents an approach for detecting and solving inconsistencies and conflicts in web software requirements and shows a taxonomy for conflicts in Web applications requirements. Lutz [2] developed CREW-Space, a tool to support the co-located collaboration of several users to simultaneously interact through Android-enabled mobile devices. They use role playing to involve different stakeholders in a use case analysis. Azadegan [3] proposes two steps: (i) identifying relevant user requirements and (ii) voting for user requirements.

The problem of conflicts also appears when building domain ontologies collaboratively. Lexons with properties, restrictions and relationships are defined in ontologies. In the LEL, there are symbols with two specific attributes (notion and behavioral responses), and relationships between the symbols are hyperlinks to other symbols used to make the description. Also each symbol has a type. The most important difference between ontologies and our approach is that we analyze these definitions, while approaches with ontologies mainly analyze the relationship between the elements. It was analyzed if there is overlapping in definition of the notion or the behavioral responses, or even if they are similar. If definitions are similar it could imply that synonyms were found. It is important to pay attention to homonyms, which are the same symbol referring to different things. Symbols (concepts) are naturally organized in a hierarchy way. This approach also analyzes how definitions are organized or repeated in such structure. In collaborative ontology engineering there is a great variety of methodologies [17], nevertheless, they do not analyze the definitions. Chen [18] proposes an approach that deals with classes and relations. They detect three kinds of conflicts: hard, soft and latent conflicts between the classes. On the subject of building ontologies collaboratively some studies apply the consensus method [19] [20]. It has been proved to be useful in conflict solution between objects. The most important problem in consensus-based collaboration, is defining when they get an agreement. Consensus quality concept [21] is defined to show, how they get a consensus, in the construction of the Vietnamese language dictionary with WordNet.

III. CONFLICTS IN THE COLLABORATIVELY DEVELOPED LEL MODEL

This section presents the proposed conflict resolution approach and a preliminary validation. Section A describes the process to identify conflicts during the collaborative description of the LEL and presents a set of the conflicts that could arise. It is important to mention that these conflicts were identified from several real-life software systems descriptions. Section B shows each conflict and the actions to solve them. Finally, section C presents a preliminary evaluation.

A. Our Approach in a Nutshell

The LEL is built in an iterative and incremental way, where different Requirements Engineers contribute to its description. With different points of view a conflict may arise. Thus, it must be identified and solved as soon as possible in order to obtain a consistent LEL (see Figure 1).

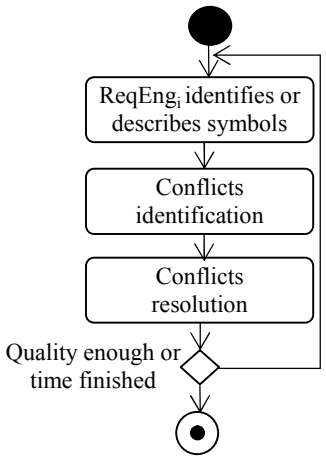

Figure 1. Process for conflicts resolution (Req Eng$_i$: Requirements Engineer$_i$)

The first step represents the action that every Requirements Engineer performs: identifying a symbol or contributing with the description of notion or behavioral response. Every action

can give origin to a conflict. For example, two different Requirements Engineers could define independently the previous Label symbols of Table 1 and 2. Thus, the whole glossary must be analyzed in order to identify conflicts. If a conflict is identified, it should be solved in order to assure the consistency of the LEL.

The list of conflicts was defined from the analysis of several real projects. The conflicts are grouped in categories in order to make the description clearer.

The first category is *Semantic conflicts:* These are conflicts that arise when there are differences in the meaning of the symbols. For example, Label refers to two different actions: the action of putting the brand (Table I) and the action of marking the price (Table II). Subcategories of Semantic conflicts are: (i) the same identification for elements with different meaning and the same syntactic classification; (ii) different identification for elements that refers to the same concept in the same way; (iii) different identification for elements that refers to the same concept in different way; (iv) different identification for elements that refers to the same concept with complementary information.

The second category is *Structural conflicts*: Structural conflicts arise when there is complete or partial repetition in the definitions, considering the description of the behavioral responses or the organization of the description in hierarchies. For example, let's consider that one symbol is a generic concept, and there is a specific term that specializes the previous one, and the last symbol repeats information described in the first one. Subcategories are: (i) different level of detail; (ii) descriptions duplicated in hierarchies.

The last category is *Syntactic conflicts*: These conflicts appear when the same symbol has different syntactic classifications. For example, Label can be an Object or a Verb. There is no subcategory.

B. Catalogue of Conflicts and their Solutions

This section describes the conflicts with more detail together with their proposed resolution. In order to illustrate the proposed approach, we chose "IP Etiquetas S.A.", a company that produces some kinds of sticky labels, either with barcodes, with specific brands or white ones. Underlined words are other LEL symbols.

The study was developed by means of a series of interviews carried out by different Requirements Engineers with several people in the company. A series of conflicts arose during the attempt to define the LEL model collaboratively. The total number of conflicts found was 17. For space reasons we show some of them in detail. The other conflicts refer to behavioral response conflicts and also conflicts generated when part of a description of notion or behavioral response defined by a requirement engineer is contained on the defined by other requirement engineer. Some examples include the Label symbol, which was considered by a Requirements Engineer as the verb meaning "attach a label," whereas another Requirements Engineer considers that Label is the produced label. A third engineer thinks the Label symbol means "attach the price tag," this being also a verb.

1) The same identification for elements with different meaning and the same syntactic classification (Homonym).

This conflict arises when there are two different entries that are identified with the same symbol, but they represent different things. For example, let's consider two different definitions of the symbol "Label" as described in Table I and Table II. The identification of both symbols is the same, since it is "Label". Nevertheless, both LEL entries refer to different things; one represents the action of putting the brand, while the other represents the action of marking the price.

TABLE I. LABEL SYMBOL

Symbol #: 10	**Author: Req. Eng. 3**	**Type[1]: Verb**
Name/s	Label	
Notion	- It is the action of putting the brand of the product on the boxes of finished product.	
Behavioral Response	-The logo of the brand is defined with the client and is previously established.	

TABLE II. LABEL SYMBOL

Symbol #: 10	**Author: Req. Eng. 1**	**Type[1]: Verb**
Name/s	Label	
Notion	- It is the action of marking the price of each finished box of finished product.	
Behavioral Response	-The price per box is previously established according to the total number required.	

Heuristic to detect the conflict: review all the LEL entries, identifying two or more entries with the same identification. Check the notion, in order to determine whether the entry is duplicated or they are different entries.

Solution: If the entry is duplicated merge both definitions. If the entries are different, specialize the identification in order to make clear that there are different entries: Label(1) and Label(2).

2) The same identification for elements with different syntactic classification (Homonym).

This conflict is similar to the previous one, but the difference relies on the type of the entries. For example, let's consider a new symbol "Label" with Verb classification (Table III), while the other "Label" symbols refers to Objects (Table I). The "Label" of object category refers to the end product manufactured by the company.

TABLE III. LABEL SYMBOL

Symbol #: 11	**Author: Req. Eng. 2**	**Type[1]: Object**
Name/s	Label	
Notion	- Product manufactured by the company	
Behavioral Response	-…	

Heuristic to detect the conflict: review all the LEL entries, identifying two or more entries with the same identification and different category.

Solution: Rename the symbols as Label(1) and Label(3).

3) Different identification for elements that refer to the same concept in the same way (Synonym).

This conflict arises when there are two different entries that are identified with different symbols, but they are described in the same way. For example, let's consider two different entries "missing stock" and "insufficient raw material" as described in Table IV and Table V. Both refer to the same situation described identically. That is, "State of raw material stock when it is lower than the minimum stock level."

TABLE IV. MISSING STOCK SYMBOL

Symbol #: 17	**Author: Req. Eng. 2**	**Type[1]: State**
Name/s	Missing stock	
Notion	-State of raw material stock when it is lower than the minimum stock level.	
Behavioral Response	-…	

TABLE V. INSUFFICIENT RAW MATERIAL SYMBOL

Symbol #: 9	**Author: Req. Eng. 3**	**Type[1]: State**
Name/s	Insufficient raw material	
Notion	-State of raw material stock when it is lower than the minimum stock level.	
Behavioral Response	-…	

Heuristic to detect the conflict: Compare all the notions of the different symbols checking for coincidences.

Solution: Define the elements as synonyms. In the example, "Missing Stock / Insufficient Raw Material element" must be defined as synonyms of the same entry.

4) Different identification for elements that refer to the same concept in different way (Overlapping).

This conflict arises when there are two different entries that are identified with different symbols, but they are described in different way. For example, let's consider two different entries "insufficient raw material" as described in Table V and Table VI. Both refer to the same situation described similarly. One symbol is described as "State of raw material stock when it is lower than the minimum stock level." while the other is described as "State of the stock of supplies when it must be changed to replenishment." Both symbols refer to the same concept, and both descriptions are similar.

TABLE VI. INSUFFICIENT RAW MATERIAL SYMBOL

Symbol #: 8	**Author: Req. Eng. 1**	**Type[1]: State**
Name/s	Insufficient raw material	
Notion	-State of the stock of supplies when it must be changed to replenishment.	
Behavioral Response	-…	

Heuristic to detect the conflict: Compare all the notions of the different symbols checking for similarities.

Solution: Since both descriptions are similar, it must be agreed only one description. The other entry must be removed. In Tables V and VI, the same symbol with a different Notion is shown.

5) Different level of detail.

This conflict arises when there are different symbols overlapping concepts in a hierarchy structure not well defined. Let's consider the situation of two different operators: (i) Rewinder Operator and (ii) Flexographic Printing Press Operator. One Requirements Engineer defines only one symbol named "Operator" with a general description considering both roles (i) and (ii). While other Requirements Engineer defines the two specific symbols (i) and (ii). In this situation, there are common characteristics to both roles; it should be described in a generic "operator" symbol, and then, the specific characteristics of both roles (i) and (ii) should be described in them.

TABLE VII. OPERATOR SYMBOL

Symbol #: 22	**Author: Req. Eng. 3**	**Type[1]: Subject**
Name/s	Operator	
Notion	-It is the technician in charge of operating the production machines.	
Behavioral Response	-…	

TABLE VIII. FLEXOGRAPHIC PRINTING PRESS OPERATOR SYMBOL

Symbol #: 9	**Author: Req. Eng. 2**	**Type[1]: Subject**
Name/s	Flexographic printing press operator	
Notion	-Is the technician in charge of operating the flexographic printing press.	
Behavioral Response	-…	

TABLE IX. REWINDER OPERATOR SYMBOL

Symbol #: 20	**Author: Req. Eng. 2**	**Type[1]: Subject**
Name/s	Rewinder operator	
Notion	-It is the person in charge of rewinding the label rolls. -It is the technician in charge of operating the rewinding machine.	
Behavioral Response	-…	

Heuristic to detect the conflict: Compare all the notions of the different symbols looking for possible hierarchy structures.

Solution: Identify the generic and specific terms of the hierarchy structure, and describe the specifics mentioning the generic. For example, in specializes symbols, refer to "Operator", saying that "He is an Operator that ..."

6) Different identification for elements that refer to the same concept with complementary information (Synonym with complementary information).

This conflict arises when there are two different entries that are identified with different symbols, and they are described

with complementary information. For example, let's consider two different entries "Cash Flow" and "Monetary Flow" as described in Table X and Table XI. Both refer to the same situation. In this case "Cash Flow" describes more details in Notion, defining it as "the amount of cash inflows and outflows" and that "it is originated by payments issued or received" while "Monetary Flow" is defined by "the amount of cash inflows and outflows". Moreover, this situation could be observed in Behavioral Response.

TABLE X. CASH FLOW SYMBOL

Symbol #: 3	Author: Req. Eng. 5	Type[1]: Object
Name/s	Cash Flow	
Notion	-It is the amount of cash inflows and outflows. -It is originated by payments issued or received.	
Behavioral Response	-It is daily prepared by the Treasurer.	

TABLE XI. MONETARY FLOW SYMBOL

Symbol #: 13	Author: Req. Eng. 1	Type[1]: Object
Name/s	Monetary flow	
Notion	-It is the amount of cash inflows and outflows.	
Behavioral Response	-It is approved and registered by Treasurer. -It is used as a source of information when preparing the Sales Forecast.	

Heuristic to detect the conflict: Compare all the notions and Behavioral Response looking for common descriptions in different symbols checking for coincidences and differences.

Solution: Define the elements as synonyms; merging all the descriptions, that is, the whole description must be used: the common part, and the particularities of each symbol. In the example, "Cash Flow / Monetary flow" must be defined as synonyms of the same entry with the richer description in each case.

7) Descriptions duplicated in in hierarchies

This conflict arises when descriptions are duplicated in specific elements of the hierarchy instead of putting them in the generic element. For example, two specific elements have the same description in the behavioral responses. Thus, the objective of the hierarchy is to put the common descriptions in the generic element. The same problem could arise in the notion.

TABLE XII. OPERATOR SYMBOL

Symbol #: 22	Author: Req. Eng. 1	Type[1]: Subject
Name/s	Operator	
Notion	-It is the technician in charge of operating the production machines.	
Behavioral Response	- Send the finished order to the Plant Manager	

Let's consider the situation of two different operators: (i) Rewinder Operator and (ii) Flexographic Printing Press Operator. A requirements engineer has placed the same behavioral response on each specialized symbol and another requirements engineer has defined a generic symbol, but the former did not realize that the generic symbol was the right place to put the description. The corresponding behavioral responses "Send the finished order to the Plant Manager" must be eliminated from each specialized, leaving this description only in the generic.

TABLE XIII. FLEXOGRAPHIC PRINTING PRESS OPERATOR SYMBOL

Symbol #: 9	Author: Req. Eng. 5	Type[1]: Subject
Name/s	Flexographic printing press operator	
Notion	-Is the technician in charge of operating the flexographic printing press.	
Behavioral Response	- Send the finished order to the Plant Manager	

TABLE XIV. REWINDER OPERATOR SYMBOL

Symbol #: 21	Author: Req. Eng. 5	Type[1]: Subject
Name/s	Rewinder operator	
Notion	-It is the person in charge of rewinding the label rolls. -It is the technician in charge of operating the rewinding machine.	
Behavioral Response	- Send the finished order to the Plant Manager	

Heuristic to detect the conflict: Compare all the notions and Behavioral Response of the different symbols looking for repetitions in the specific elements.

Solution: Move the repeated description from the specific elements to the generic one.

C. *Preliminary Evaluation*

In order to validate the conflicts proposed in this paper, we analyzed a LEL built collaboratively by 5 Requirements Engineers. We analyze the resulting LEL looking for the conflicts we proposed. Then, we present every report to Requirements Engineers who participated in the construction of the LEL to check whether they agree with the conflicts reported. Requirements Engineers have agreed in almost all the conflict reported. The following Table XV presents some figures for the 5 different participants.

TABLE XV. TOTAL OF CONFLICTS FOUND IN IP ETIQUETAS

Req. Eng.	Total of symbols described	Symbols with conflicts	Percentage
Req. Eng. 1	42	31	74
Req. Eng. 2	35	28	80
Req. Eng. 3	28	21	75
Req. Eng. 4	31	27	87
Req. Eng. 5	47	38	80

Table XV presents for each Requirements Engineers the number of symbols in which he participated in their description, the symbol with conflict identified by our approach and the percentage that it represents. This table shows that conflicts are very common.

IV. Conclusions And Further Work

Requirements definition is one of the initial stages in the software development process and their products are the groundwork for subsequent stages. Thus, errors made in requirements stage will be replicated and deepened in subsequent stages. For this reason, it is extremely important to develop requirements models of the highest quality as possible. When requirements models are developed collaboratively, conflicts unavoidable will arise. Moreover, natural language descriptions are more plausible to give origin to conflicts.

A vast experience in working with a structured glossary, the Language Extended Lexicon (LEL), proves that such structure reduces the occurrence of conflicts. However engineers have observed that while building the LEL collaboratively produces a richer model, it also introduces conflicts. In our research, and by analyzing several application domains of real projects, a classification of conflicts was devised. A process and guides for their resolution has been described in this paper. Our approach with some examples of a real project was also illustrated.

A preliminary evaluation was presented; it showed the importance of identifying conflicts and the solutions for the conflicts proposed. The percentage of conflicts was between 74% and 87%, in the five groups that have been evaluated. It shows the importance of solving those conflicts for arriving to better quality models.

An experiment to validate the conflicts and their resolutions is being designed. This experiment will be conducted in a different country to validate in another context the findings presented in this paper.

A process to identify the conflicts and an automated suggestion of solutions is planned. This implementation will be based on two important modules: (i) a module of natural language processing and (ii) a module of machine learning.

References

[1] B.W. Boehm, Software Engineering Economics. Prentice Hall, 1981.

[2] R. Lutz, S. Schäfer, and S. Diehl, "Using mobile devices for collaborative requirements engineering", 27th IEEE/ACM International Conference on Automated Software Engineering, pp. 298–301. ACM. 2012.

[3] A. Azadegan, X. Cheng, F. Niederman, and G. Yin, "Collaborative requirements elicitation in facilitated collaboration: report from a case study", 46th Hawaii International Conference on System Sciences, pp. 569–578, ISSN 15301605, IEEE, 2013.

[4] J. C. S. D. P. Leite, and A. P. M. Franco, "A strategy for conceptual model acquisition", Requirements Engineering, IEEE International Symposium on , pp. 243–246. IEEE, 1993.

[5] A. d. P. A. Oliveira, J. C. S. d. P. Leite, L. M. Cysneiros and C. Cappelli, "Eliciting Multi-Agent Systems Intentionality: from Language Extended Lexicon to i* Models", Chilean Society of Computer Science, 2007. SCCC '07. XXVI International Conference of the, Iquique, 2007, pp. 40–49, doi: 10.1109/SCCC.2007.20

[6] S. Goel, "Transformation from LEL to UML", International Journal of Computer Applications, vol. 48, no. 12, 2012.

[7] L. Antonelli, G. Rossi, and Oliveros A., "A collaborative approach to describe the domain language through the Language Extended Lexicon", Journal of Object Technology, vol. 15, no. 3, pp. 1–27, 2016.

[8] N. Mulla, S. Girase S, "A new approach to requirement elicitation based on stakeholder recommendation and collaborative filtering", International Journal of Software Engineering and Applications, vol. 3(3), pp. 51–60, 2012, doi:10.5121/ijsea.2012.3305.

[9] S. Easterbrook, "Resolving requirements conflicts with computer-supported negotiation", Requirements engineering: social and technical issues, vol. 1, pp. 41–65, 1994.

[10] W. N. Robinson, S. D. Pawlowski, and V. Volkov, Requirements interaction management. ACM Computer Survey, vol. 35(2), pp. 132–190, 2003.

[11] H. Bendjenna, P. J. Charrel, and N. E. Zarour, "Using AHP Method to Resolve Conflicts Between Non-Functional Concerns", International Conference on Education, Applied Sciences and Management *(ICEASM'2012)*, Dubai, UAE, pp. 26–27, 2012.

[12] M. Aldekhail, A. Chikh, and D. Ziani, "Software Requirements Conflict Identification: Review and Recommendations", International Journal of advanced computer science and applications, vol. 7, no. 10, pp. 326–335, 2016.

[13] B. Decker, E.Ras, J. Rech, P. Jaubert, and M.Rieth, "Wiki-based stakeholder participation in requirements engineering". IEEE Software, vol. 24(2), pp. 28–35, 2007.

[14] J. Robertson, and S. Robertson, "Volere Requirements Specification Template".The Atlantic Systems Guild, 2012.

[15] D. Yang, D. Wu, S. Koolmanojwong, Brown, A. W., and B. W. Boehm, "Wikiwinwin: A wiki based system for collaborative requirements negotiation", Hawaii International Conference on System Sciences, Proceedings of the 41st Annual, pp. 24–24. IEEE, 2008.

[16] M. Urbieta, M. J. Escalona, E. R. Luna, and G. Rossi, G., "Detecting conflicts and inconsistencies in web application requirements", International Conference on Web Engineering, pp. 278–288. Springer, Berlin, Heidelberg, 2011.

[17] E. Simperl, and M. Luczak-Rösch, "Collaborative ontology engineering: a survey", The Knowledge Engineering Review, vol. 29, no. 1, pp. 101–131, 2014.

[18] Y. Chen, X. Peng, and W. Zhao, "An approach to detect collaborative conflicts for ontology development", Advances in Data and Web Management, pp. 442–454.Springer, Berlin, Heidelberg. 2009.

[19] S. Karapiperis, and D. Apostolou, D. "Consensus building in collaborative ontology engineering processes". Journal of Universal Knowledge Management, vol 1(3), pp. 199–216, 2006.

[20] N. T. Nguyen, "Advanced methods for inconsistent knowledge management". Springer, London (2008)

[21] T. H. Duong, M. Q. Tran, and T.P.T. Nguyen, "Collaborative Vietnamese WordNet building using consensus quality", Vietnam J ComputSci 2017, vol 4:85, Springer Berlin Heidelberg, Print ISSN: 2196-8888, Online ISSN: 2196-8896, 2017.