A Collaborative Approach to Capture the Domain Language

Leandro Antonelli[1], Gustavo Rossi[1], Alejandro Oliveros[2]

[1] Lifia, Fac. de Informática, UNLP, calle 50 esq 120, La Plata, Bs As, Argentina
{lanto, gustavo}@lifia.info.unlp.edu.ar
[2] INTEC-UADE y UNTREF, Bs As, Argentina
aoliveros@gmail.com

Abstract. Software development is a succession of descriptions in different languages where a previous description is necessary for the next one. Thus, it is important to begin software development with requirements that are as correct and as complete as possible. Although some literature holds the belief that correctness and completeness are two attributes that requirements specifications must satisfy, we know that these attributes are very difficult to meet. However, we have to find ways to diminish the level of incompleteness and deal with the possible conflicts that do arise in the requirements context. Defining the domain language before specifying the requirements is a way of coping with this problem. Nevertheless, it is hard to produce a domain language specification when there are many stakeholders involved. We rely on *collaboration* in order to foster the cooperation of the stakeholders, thus they are able to explore the differences constructively and search for solutions that go beyond their own limited views. In this paper, we propose a strategy to capture the domain language in a collaborative way using Language Extended Lexicon and we show a preliminary validation of the proposed strategy.

Keywords. Domain Analysis, Language Extended Lexicon, Collaboration

1. Introduction

The development of software systems is a complex activity since different sorts of actors with different backgrounds are involved. During development they perform various tasks at different moments with the aim of building diverse software products. This means that it is difficult to define a precise schedule to organize the development at the beginning of the project, when the information about it is not very detailed. The result is a coarse-grained schedule whose deadlines become a goal to accomplish instead of a reasonable estimation of the delivery times of the products. As a consequence, overcoming a delay becomes almost impossible. In other engineering disciplines, like civil engineering, it is usual for some tasks to be repetitive, and increasing the number of people can reduce the overall schedule. But Software engineering is different as, in general, tasks are very specific, and if we add more people to overcome the delay, we often lose more time (at least at the beginning, when new people are incorporated). This fact is known as Brooks' Law [5].

Furthermore, the nature of software also makes its development a complex task. To illustrate this, we can list four well-known software characteristics (adapted from [5]) that make software a different kind of artifact altogether, contributing to the complexity of its construction:

(i) Invisibility: software is not visible as a product, like a building, a car or a plane. This characteristic imposes a barrier on human perception of size, form and structure.

(ii) Modifiability / Changeability: software is built by descriptions, which are very malleable since changes can be made abstractly by rewriting. This creates the illusion that software is easy to change.

(iii) Conformity: software has to conform to its infrastructure, which can also be software. It does not exist per se. Changes in the infrastructure imply changes in the software.

(iv) Complexity: software needs reification by other artifacts, including software, to transform itself from an idea into a product. This characteristic corroborates how difficult it is for humans to have a clear idea of its limits, its interaction with the environment and its shape.

Ackoff states that we commonly fail in software construction not because the solution is not technically well-built, but because it does not apply to the problem [1]. Nowadays, this statement is still true, as several surveys confirm [37].

We can represent the software development process as a succession of descriptions in different languages where a previous description is necessary for the next one [36]. So, if changes are incorporated into a description, previous and succeeding descriptions will have to be changed in order to maintain conformity. For instance, Boehm [4] states that if a mistake occurs in a requirements description and it is corrected in code description, the correction cost could be multiplied by up to 200. Moreover, Mizuno developed the "waterfall of errors" [28], in which he states that in each stage of software development the possibility of occurrences of mistakes is stronger than in the previous one, because each stage relies on products from previous states.

Thus, it is important to begin software development with requirements that are as correct and as complete as possible. Although some literature holds the belief that correctness and completeness are two attributes that requirements specifications must satisfy [20], we know that this is unfeasible [15]. However, we have to find ways to diminish incompleteness and deal with the possible conflicts that do arise in the requirements context. Defining the domain language before specifying the requirements is a way of coping with this problem.

The Language Extended Lexicon (LEL) is a technique for specifying an application domain (context) language [23]. LEL is a very convenient tool for stakeholders with no technical skills, although people with such skills will profit more from its use [31]. LEL effectively captures and models the application domain language because it conforms to the mechanism used by the human brain to organize expert knowledge [39]. In particular, the convenience of LEL as a tool arises from three significant characteristics: it is easy to learn, it is easy to use and it has good expressiveness. There are several publications using LEL in complex domains which

validate these claims. Gil et al [16] state that "building a LEL in an application completely unknown to the requirements engineer and with highly complex language can be considered a successful experience, since users stated that requirements engineers have developed a great knowledge about the application". Cysneiros et al [8] state that "the use of LEL was very well accepted and understood by the stakeholders. As these stakeholders were non-technical experts from a specific and complex domain, the authors believe that LEL can be suitable to be applied in many other domains". The characteristics mentioned above contribute significantly to obtaining high quality models, as they allow the actors involved in software development (experts, requirements engineers and developers with different capacities and abilities) to perform the validation of a LEL [21].

Nevertheless, it is very difficult to produce a domain language specification when there are many stakeholders involved [29] [41] [26]. In requirements elicitation, the number of stakeholders is often used to size a project. Cleland-Huang and Mobasher define an ultra-large-scale project to have thousands or even hundreds of thousands of stakeholders [7]. According to Northrop et al. [30] and Cheng and Atlee [6], the human interaction element makes requirements elicitation the most difficult activity to scale in software engineering. We rely on *collaboration* in order to foster the cooperation of the stakeholders, thus they are able to explore the differences constructively and search for solutions that go beyond their own limited views [19] [34] [33] [27] [41]. In a collaborative context, all participants co-construct together even if the task can be divided into several new subtasks. Requirements elicitation, as an interdisciplinary process, requires specific competences from all the users and the stakeholders involved. Collaboration is therefore necessary, as no one person can possess all the competences required for this task. In addition, good collaborative requirements elicitation produces richer, more complete and more consistent requirements [22].

In this paper, we show how to capture the domain language in a collaborative way. The rest of the paper is organized as follows: Section 2 presents some background necessary to understand the strategy. Section 3 describes the strategy. Section 4 shows an experiment validating the strategy. Section 5 discusses some related works. Finally, section 6 presents some conclusions and future works.

2. Background

This section describes the Language Extended Lexicon (LEL), a technique used to capture the language of the application domain.

LEL is a glossary whose goal is to register the definition of terms that belong to a domain. It is tied to a simple idea: "understand the language of a problem without worrying about the problem" [23].

Terms (called symbols within LEL) are defined through two attributes: notion and behavioural responses. Notion describes the symbol denotation and the intrinsic and substantial characteristics of the symbol, while behavioural responses describe connotation, i.e. the relationship between the term being described and other terms.

There are two principles that must be followed while describing symbols: the circularity principle (also called closure principle) and the minimal vocabulary

principle. The circularity principle states that the use of LEL symbols must be maximized when describing a new symbol. The minimal vocabulary principle states that the use of words that are external to the Lexicon must be minimized. These principles are vital in order to obtain a self-contained and highly connected LEL. Connections among symbols determine that LEL can be viewed as a graph.

Each symbol of the LEL belongs to one of four categories: subject, object, verb or state. This categorization guides and assists the requirements engineer during the description of attributes. Table 1 shows each category with its characteristics and how to describe them.

Table 1. LEL categories

Category	Characteristics	Notion	Behavioral responses
Subject	Active elements which perform actions	Characteristics or condition that subject satisfies	Actions that subject performs
Object	Passive elements on which subjects perform actions	Characteristics or attributes that object has	Actions that are performed on object
Verb	Actions that subjects perform on objects	Goal that verb pursues	Steps needed to complete the action
State	Situations in which subjects and objects can be	Situation represented	Actions that must be performed to change into another state

Some examples of LEL symbols are presented here. The classic bank application is used to show symbols from each category. The example consists in a bank which allows its clients to open and close accounts. If the account is activated (open) the client can deposit or withdraw money and consult the balance. The bank can also perform a cash audit.

It is important to mention that we underline the terms which correspond with other defined symbols in order to show the application of the circularity principle. The following examples are: subject *client* in Figure 1; object *account* in Figure 2; verb *withdraw* in Figure 3; and state *activated* in Figure 4.

3. Our approach

This section describes the proposed approach to construct a LEL in a collaborative way. First, we describe its essence and, afterwards, we give details and examples about the approach. The approach consists of two main steps. First, a network of stakeholders must be described using the snowballing technique [26]. After that, the symbols of the LEL must be identified and described.

In order to describe the network of stakeholders with the snowballing technique, some key stakeholders must be identified to begin the process. Then, these key stakeholders nominate more stakeholders who, in turn, appoint some others. Thus, a network description in which nodes describe stakeholders and links describe recommendation is built. The node must also include the role of the stakeholders.

Subject: client
Notion
Person that operates an account.
Behavioral responses
The client can open an account.
The client can deposit money into his account.
The client can withdraw money from his account.
The client can consult his account balance.
The client can close an account.

Figure 1. *Client* symbol's description.

Object: account
Notion
The account has a balance.
Behavioral responses
The client can open an account.
The client can deposit money into his account.
The client can withdraw money from his account.
The client can consult his account balance.
The bank performs a cash audit.
The client can close an account.

Figure 2. *Account* symbol's description.

Verbs: withdraw
Notion
Act of taking money from the account.
Behavioral responses
The bank must check that the account has enough money to perform the withdrawal.
The bank must check that the owner of the account has not withdrawn more times than the limit allows.
The bank must check that the owner of the account does not have any credit card debts.
The bank reduces the balance of the account according to the amount withdrawn.

Figure 3. *Withdraw* symbol's description.

State: Activated
Notion
Situation where the client is ready to use an open account.
Behavioral responses
The client can close the account and he will have a closed account.

Figure 4. *Activated* symbol's description

In order to build the LEL in a collaborative way, the stakeholder involved in the network must identify symbols, describe them, and also vote (indicate they *like*) for descriptions. First, subject symbols must be identified from the network and described. Basically, each role of the network must be considered a subject symbol. In order to describe symbols, different people can add different expressions to the same symbol. And they can also indicate that they *like* the expression. Symbols from other categories must be identified from subject symbols. In general, verb symbols can be identified from behavioural responses of the subjects. Then, while describing verb

symbols, object symbols can be identified. The mechanism to describe symbols of the three categories is similar to the description of subject symbols.

The following figure 5 summarizes the approach. It begins with the identification of key stakeholders. Then, the network of stakeholders is built using the snowballing technique. Finally, the LEL is defined in a collaborative way with the participation of the stakeholder from the network,

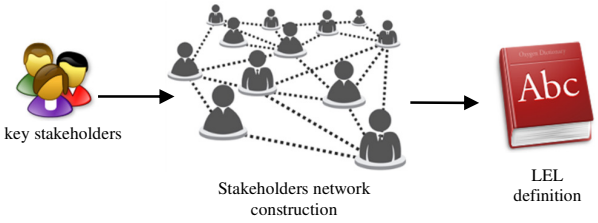

Figure 5. Our approach in a nutshell

3.1. Building a network of stakeholders with snowballing

In social network analysis, the snowballing method is generally used to sample social network data for large networks where the boundary is unknown. Snowball sampling begins with a set of actors. Each of these actors is asked to nominate other actors. Then, new actors who are not part of the original list are similarly asked to nominate other actors. As the process continues, the group of actors builds up like a snowball rolled down a hill. The process continues until no new actors are identified, time or resources have run out, or when the new actors being named are very marginal to the actor set under study [26].

In order to build the network of stakeholders with the snowballing technique, the requirements engineer in charge of constructing the LEL must identify, with the help of the sponsor of the project, some key stakeholders who will initiate the nomination process. It is important to identify the stakeholders and also to define the role they play. This must be done in every nomination.

Let's consider an example from the banking domain. The key stakeholders are John (an accountant) and Walter (a client of the bank). John nominates Arthur (a cashier), and Walter nominates Alice (another client) and also nominates Arthur. Then, both Arthur and Alice nominate Laura (the general manager). Finally, Laura nominates Walter, but since he had been nominated as a key stakeholder, no new stakeholder was nominated so the process ends. Figure 6 shows the network of stakeholders with their roles and nominations.

3.2. Building a LEL in a collaborative way

The traditional process to build a LEL consists of two activities: identify and describe the symbols which are performed exclusively by a requirements engineer. In our collaborative approach we propose both activities and we also add a common social activity: express *like* to an expression that defines a symbol. It is important to

mention that the collaborative approach we propose includes no requirements engineer and the activities are performed directly by the stakeholders. Thus, the identification and the description of symbols also occurs in a collaborative way, and different people cooperate to define a symbol. For example, one person identifies a symbol and another one includes an expression to describe it. Next, another different person includes one more expression to describe the symbol. And, finally, somebody else reviews the symbol and its definition and can indicate that he *likes* some expression. Although we could have used a rating scalefrom 0 to 5 instead of the *like* tool, we wanted to identify the most adequate expression to describe a symbol, and the *like* tool proved to be more effective in this situation [3].

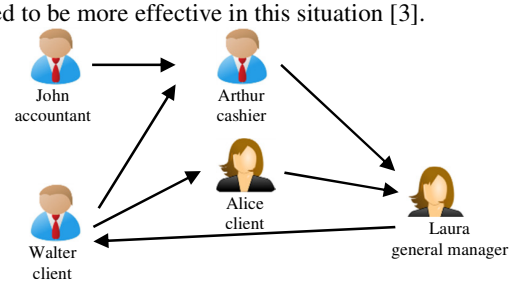

Figure 6. Stakeholder network example

Let's consider the following example. John identifies the symbol *client*. He only identifies the symbol and does not include any definition. Then, Walter adds an expression ("*The client can deposit money into his account.*") into the behavioural responses of the symbol *client.* After that, Arthur adds another expression ("*The client can withdraw money from his account.*") into the behavioural responses of the symbol *client*. Finally, Alice cannot add any new descriptions but she says that she *likes* the description byWalter. The following figure 7 shows this collaboration.

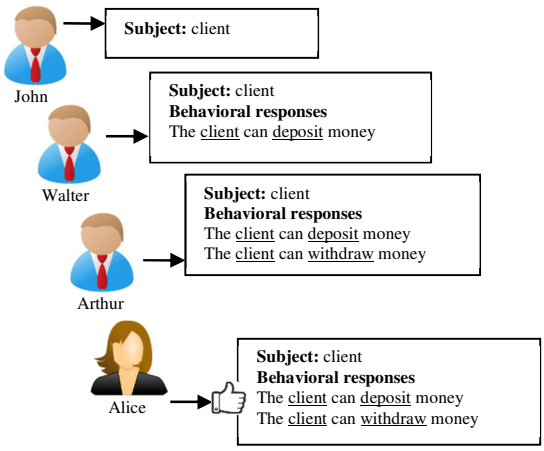

Figure 7. Collaboration in the definition of the symbol.

The identification of symbols can be performed in several steps. Symbols of the subject category must be identified from the roles of the network. Basically, each role of the network must be considered a subject symbol. Then, the description of behavioural responses of subject symbols will provide the source to identify symbols of the verb category. And, finally, the description of behavioural responses of the verb symbols will provide object symbols. Of course, identification and description occur in an iterative and incremental way and the steps mentioned are only suggestions.

Let's consider the *client* symbol of the subject category which refers to the operation (verb) *withdraw* in its behavioural responses. Thus, the verb *withdraw* must be defined and described. Then, the symbol of the object category *account* appears in the behavioural responses of the symbol *withdraw*, so the symbol *account* must be described. Figure 8 shows the example below.

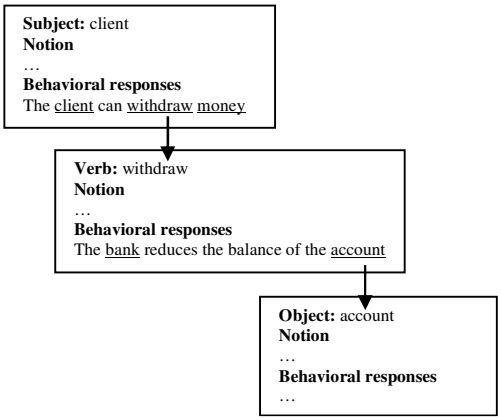

Figure 8. Identification of symbols' example

It is important to mention that the identification and the description of symbols occurs in an iterative and incremental way. There are not rigid steps. It is a recommendation to describe subjects first, then verbs and finally objects. Considering the previous example, it is possible that some stakeholder identify the subject symbol *bank* from the verb symbol *withdraw* after describing this verb. Moreover, in order to describe *bank*, the object symbol *money* (that also appears in *client*) may be identified and described. Thus, people collaborate in an "anarchical" way identifying, defining and signaling *like* to symbols.

4. Case Study

We have conducted a case study in order to verify the applicability of a collaborative approach. That is, we have verified that a group of people can work in a common application domain to produce a richer LEL than the LEL produced only by one person. We have not validated the proposed approach yet, but we are currently running an experiment to verify it and we will present the results in a future work. In

this work we only wanted to show that collaboration of people working in the same application domain can produce a richer specification.

The participants of the case study were 41 people from a Computer Science postgraduate course. The case study was part of the course activities. All the students had a degree in Computer Science and experience in the software development industry: some of them as developers, others as analysts and some others as team leaders. Some participants were also teachers at the University. The majority of the participants were Argentinean, but there were also people from Colombia.

The case study was presented during a class and then, the students had 5 weeks to describe the symbols. At the presentation of the case study, the participants were instructed how to identify symbols and describe them. The participants received 3 hours of training and they also received reference material. Then, 15 groups were formed (some groups worked remotely). Every group had to produce one LEL, thus, at the end of the case study, we had 15 different LEL to analyze. The students had 5 weeks (from November 2013 until December 2013) to produce the LEL. The application domain chosen was a travel site. Participants received a brief introduction and they also received two websites to study. There were two steps during the construction. After 2 weeks, they had to present the list of identified symbols and some symbols described in order to evaluate the progress of the work. After some feedback, participants had 3 more weeks to finish the LEL.

The analysis to verify that a collaborative approach can produce a richer LEL than the LEL produced only by one person consisted in verifying that there is some part of the LEL that is shared by all the people and then, every group enriches the LEL with different symbols and different descriptions of the symbols. It is important that all the groups agree in a shared set of "core" symbols because, although they can add a different point of view to the LEL, they must agree in the essence of the domain.

Considering all the 15 LEL produced by the groups, a total of 595 symbols were identified and described. That means that an average of 40 symbols were identified and described by each group. It is important to mention that there were some replicated symbols within the total number. Thus, the number of symbols without duplication was 281. That means that there were a lot of repeated symbols among the LEL produced by each group. And each group provided, on average, a small percentage of non-duplicated symbols. Thus, the collaboration of different people working in the same application domain allows obtaining a richer LEL than the one obtained by only one person. We have analyzed the percentage of the average of non-repeated symbols grouped by categories. And we have found that every group provides 23% of non-repeated Subject symbols, while it only provides between 13% and 14% of non-repeated Object, Verb and State symbols (Table 2). That could mean that 5 groups would be sufficient to identify and describe subjects, while we need 8 groups for the other symbols. But in order to verify this finding, we will run another experiment. For this case study's goal, the number shows that a collaborative approach has benefits over an individual approach.

Table 2. LEL categories

Category	Average of Non replicated symbols
Subject	23%
Object	13%
Verb	14%
State	13%

We have identified the most repeated symbols, in order to analyze the repetition in the description of the symbols. We identified the symbols defined by more than 11 groups. These are 13 symbols and they are enumerated in Table 3 and, in particular, we focused on symbols *client* and *search* because they were the most representative symbols of the application domain.

Table 3. Symbols most repeated

Category	Average of Non replicated symbols
Subject	Client Company
Object	Flight Hotel Service Car
Verb	Search Cancel Reserve Buy Recommend
State	Reserved Canceled

We have analyzed the number of expressions in notion and behavioural responses descriptions as well as the average by group in order to identify the symbol to use for a deeper analysis. Since *client* is a symbol with more description than *search*, we decided to analyze the *client* symbol. In particular, we analyzed the behavioural responses because they have the most complete description (Table 4).

Table 4. Symbols most repeated

	Client		Search	
Expressions	**Notion**	**Behavioural Responses**	**Notion**	**Behavioural Responses**
Total Number	53	309	16	49
Average	3.78	22.14	1.06	3.26

The groups identified 309 expressions to describe behavioural responses for the symbol *client*. These 309 expressions contain only 117 different expressions and 192 expressions repeat someone of the 117 expressions.

In summary, we have analyzed the repetition of the identification of symbols and the repetition of the expressions to describe a symbol and we have found that a group of different people working in a same application domain agree in a core description

of them and also add more description according to their different points of view. Thus, the LEL produced collaboratively is richer than the LEL produced by only one person.

5. Related Works

Collaboration is used in a wide range of stages in software development. For example in: UI design [18], architectural design [17] and coding [32]. Even in the process of locating (adapting) global systems [14]. Collaboration is also related with iterative process. For example, in [38] the authors propose Iterative cycles of requirements analysis and design exploration to build mutual understanding about users' requirements and the space of possible solutions for those requirements.

There are some similar works in the requirement engineering step. For example, [2] proposes two steps: (i) identify relevant user requirements and (ii) vote for user requirements. Our work begins with identification of users and follows with definition of requirements and voting. The difference lies on the fact that these works produce different products.

Collaborative filtering is a technique for filtering large sets of data for information and patterns. This technique is used in recommender systems to forecast a user's preference. The underlying assumption is that users who have had a similar taste in the past will share a similar taste in the future [26]. We are not interested in predictions because, since we need to capture the language, we must not bias the creativity process.

Dheepa et al. [13] introduce a novel method that uses social networks and collaborative filtering to manage the requirement elicitation process. They use the snowball method to build the user network and prioritize requirements. And they use collaborative filtering to make predictions about requirements. There are other works in the same line. For example, [24] prioritizes stakeholders on three attributes: the power to influence the project, the legitimacy and the urgency of their claims. [26] [25] [12] prioritize their requirements using their ratings weighted by their project influence derived from their position on the social network. We also use snowballing technique, but we do not consider the relevance of the users, because we consider that every vote has the same weight.

Reenadevi et al. [35] proposes a strategy to identify malicious stakeholders, since requirements rated by malicious stakeholders affect the quality of product. We believe that malicious stakeholders will not affect the LEL description, because there is no prioritization similar to requirements prioritization. [9] [10] use social network analysis to study collaboration, communication and awareness among stakeholders. Social network measures, such as degree centrality and betweenness centrality, were used to analyse the collaboration behaviour. We do not perform this analysis, but we have this information, thus, we shall include this analysis in a future work.

6. Conclusions and Future Works

Capturing knowledge from an application domain can be very disappointing if we do not involve all the stakeholders. But involving too many people can be difficult to manage. Thus, in order to cope with this problem, we have presented an approach to capture the language of the application domain in a collaborative way. This approach allows involving as many stakeholders as are needed to produce a richer and more complete LEL than the one produced only by one stakeholder. We have presented a preliminary case study that has shown that in a collaborative construction of the LEL, there is a small subgroup of symbols which are shared by the majority of the participants, and then, every participant enriches the LEL with his own point of view. Nevertheless, we are running a new experiment to apply the method we propose and we also plan to include the role of the moderator [40] [11] in order to focus on the collaborative work. Moreover, we also plan to add the social function *dislike* and the possibility of removing both previously signaled *like* and *dislike*.

Another future work involves dealing with the identification of synonyms. Since many people are identifying and describing symbols, the same symbols could be identified twice through the use of synonyms as identification. Thus, their descriptions (in particular the analysis of the most voted descriptions) could be used to detect synonyms and merge both symbols.

Finally, we believe that it is important to build a specific tool to support the process proposed. Although there are some general tools like wikis that can be used to support this collaborative process, we are developing one for this specific purpose.

References

1. Ackoff, R.: Redesigning The Future, Wiley, 1974.
2. Azadegan, A., Cheng, X., Niederman F., Yin, G.: Collaborative Requirements Elicitation in Facilitated Collaboration: Report from a Case Study, in Proceeding of the 46th Hawaii International Conference on System Sciences, DOI 10.1109/HICSS.2013.136, IEEE, 2012, pp 569-578
3. Bao, J., Sakamoto, Y., Nickerson, J. V.: Evaluating Design Solutions Using Crowds, in proceedings of the Seventeenth Americas Conference on Information Systems, Detroit, Michigan August 4th-7th, 2011.
4. Boehm, B.W.: Software Engineering, Computer society Press, IEEE, 1997.
5. Brooks, F.: The Mythical Man-Month: Essays on Software Engineering, Addison-Wesley Professional, 2 edition, 1995.
6. Cheng, B. H. C., Atlee, J. M.: Research directions in requirements engineering, in proceedings of the Conference on The Future of Software Engineering, 2007, pp. 285-303.
7. Cleland-Huang, J., Mobasher, B.: Using data mining and recommender systems to scale up the requirements process, in proceedings of the 2nd International Workshop on Ultra-Large-Scale Software-Intensive Systems, 2008, pp. 3-6.
8. Cysneiros, L.M., Leite, J.C.S.P.: Using the Language Extended Lexicon to Support Non-Functional Requirements Elicitation, in proceedings of the Workshops de Engenharia de Requisitos, Wer'01, Buenos Aires, Argentina, 2001.

9. Damian, D. , Marczak, S., Kwan, I.:Collaboration patterns and the impact of distance on awareness in requirementscentred social networks, in proceedings of the 15th IEEE International Conference on Requirements Engineering, 2007, pp. 59-68,
10. Damian, D., Kwan, I., Marczak, S.:Requirements-driven collaboration: Leveraging the invisible relationships between requirements and people, Collaborative Software Engineering, Berlin Heidelberg, Springer, 2010.
11. Decker, B., Ras, E., Rech, J., Jaubert, P., Rieth, M.: Wiki-Based Stakeholder Participation in Requirements Engineering, Software, ISSN 0740-7459, DOI: 10.1109/MS.2007.60, IEEE (Volume:24 , Issue: 2), March-April 2007, pp 28 - 35
12. Dheepa, V., Aravindhar, D.J., Vijayalakshmi, C.: A Novel Method for Large Scale Requirement Elicitation, International Journal of Engineering and Innovative Technology (IJEIT) Volume 2, Issue 7, January 2013, pp375-379.
13. Dheepa, V., Vijayalakshmi, C., Naganathan, E. R.: A Novel Method For Large Scale Requirement Elicitation, International Journal of Advanced Computational Engineering and Networking, ISSN (p): 2320-2106, Volume – 1, Issue – 1, Mar-2013, pp 7-12.
14. Exton, C., Wasala, A., Buckley, J., Schaler, R.: Micro Crowdsourcing: a new model for software localization, Localisation focus, the international journal of localisation, issn 1649-2358, special cngl edition, vol 8, issue 1, 2009, pp 81-89
15. Finkelstein, A.C.W., Gabbay, D., Hunter, A., Kramer, J., Nuseibeh, B.: Inconsistency handling in multiperspective specifications, IEEE Transactions on Software Engineering, doi: 10.1109/32.310667, vol.20, no.8, Aug, 1994, pp 569-578.
16. Gil, G.D., Figueroa, D.A., Oliveros, A.: Producción del LEL en un Dominio Técnico. Informe de un caso, in proceedings of the Workshops de Engenharia de Requisitos, Wer'00, Rio de Janeiro, Brazil 2000.
17. Greenwood, P., Rashid, A., Walkerdine, J.: UDesignIt: Towards Social Media for Community-Driven Design, ICSE 2012, Zurich, Switzerland, New Ideas and Emerging Results, 978-1-4673-1067-3/12/ 2012, IEEE, 2012.
18. Heer, J., Bostock, M.: Crowdsourcing Graphical Perception: Using Mechanical Turk to Assess Visualization Design", in proceeding of the CHI 2010, Atlanta, Georgia, USA, ACM 9781605589299, 10/04, April 10–15, 2010.
19. Howe, J.: The Rise of Crowdsourcing, Wired, 14(6), URL (accessed 24 November 2014): http://www.wired.com/wired/archive/14.06/crowds.
20. IEEE, IEEE Recommended Practice for Software Requirements Specifications, IEEE Std 830-1998 (Revision of IEEE Std 830-1993).
21. Kaplan, G., Hadad, G., Doorn, J., Leite, J.C.S.P.: Inspeccion del Lexico Extendido del Lenguaje, In: proceedings of the Workshops de Engenharia de Requisitos, Wer'00, Rio de Janeiro, Brazil, 2000.
22. Konaté, J., Sahraoui, A. E. K., Kolfschoten, G. L.: Collaborative Requirements Elicitation: A Process-Centred Approach, Group Decision and Negotiation, Springer July 2014, Volume 23, Issue 4, pp 847-877.
23. Leite, J.C.S.P., Franco, A.P.M.: A Strategy for Conceptual Model Acquisition, In Proceedings of the First IEEE International Symposium on Requirements Engineering, San Diego, California, IEEE Computer Society Press, 1993, pp 243-246.
24. Lim, S. L., Quercia, D., Finkelstein, A.: StakeNet: using social networks to analyse the stakeholders of largescale software projects. In Proc. of the 32nd Int. Conf. on Soft. Eng. Vol. 1, Cape Town, 2010, pp. 295-304.

25. Lim, S. L., Damian, D., Finkelstein, A.: StakeSource2.0: Using Social Networks of Stakeholders to Identify and Prioritise Requirements, ICSE'11, ACM 978-1-4503-0445-0/11/05, May 21–28, Waikiki, Honolulu, HI, USA, 2011, pp 1022-1024.
26. 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, 2012, pp 707-735.
27. Mistrík, I., Grundy, J., Van Der Hoek, A., Whitehead, J.: Collaborative Software Engineering: Challenges and Prospects, Collaborative Software Engineering, DOI 10.1007/978-3-642-10294-3_19, Springer-Verlag Berlin Heidelberg, 2010, pp 389-403.
28. Mizuno, Y.: Software Quality Improvement, IEEE Computer, Vol. 16, No. 3, March, 1983, pp 66 – 72.
29. Mulla, N., Girase, S.: A New Approach To Requirement Elicitation Based On Stakeholder Recommendation And Collaborative Filtering, International Journal of Software Engineering & Applications (IJSEA), Vol.3, No.3, May 2012, DOI :10.5121/ijsea.2012.3305, pp 51-60.
30. Northrop, L., Feiler, P., Gabriel, R. P., Goodenough, J., Linger, R., Longstaff, T., Kazman, R., Klein, M., Schmidt, D., Sullivan, K.: Ultra-Large-Scale Systems: The Software Challenge of the Future: Software Engineering Institute, 2006.
31. Oliveira, A.d.P.A., Leite, J.C.S.P., Cysneiros, L.M., Cappelli, C.: Eliciting Multi-Agent Systems Intentionality: from Language Extended Lexicon to i* Models, in proceedings of XXVI International Conference of the Chilean Society of Computer Science (SCCC '07), 8-9 Nov, 978-0-7695-3017-8, 2007, pp 40 – 49.
32. Ponzanelli, L., Bacchelli, A., Lanza, M.: Leveraging Crowd Knowledge for Software Comprehension and Development, in Proceedings of CSMR 2013 (17th IEEE European Conference on Software Maintenance and Reengineering), 2013, pp. 59-66.
33. Portillo-Rodríguez, J., Vizcaíno, A., Piattini, M., Beecham, S.: Tools used in Global Software Engineering: A systematic mapping review, Information and Software Technology, Volume 54, Issue 7, July 2012, pp 663-685.
34. Quinn, A. J., Bederson, B. B.: A Taxonomy of Distributed Human Computation, Tech. Rep. HCIL-2009-23, University of Maryland, 2009.
35. Reenadevi, R., Dugalya, P.: Identifying Malicious Stakeholders Using Algorithm For Large Scale Requirement-Elicitation, International Journal of Computer & Communication Technology ISSN (PRINT): 0975 - 7449, Volume-3, Issue-6, 7, 8, 2012, pp 106-108
36. Stahl, T ., Voelter, M., Czarnecki, K.: Model-Driven Software Development: Technology, Engineering, Management , Wiley Software Patterns Series, ISBN: 978-0470025703, 2006.
37. Standish Group, The Chaos Report, http://blog.standishgroup.com/, 2013.
38. Sutcliffe, A.: Collaborative Requirements Engineering: Bridging the Gulfs Between Worlds, Intentional Perspectives on Information Systems Engineering, DOI 10.1007/978-3-642-12544-7_20, Springer-Verlag Berlin Heidelberg 2010, pp 355-376.
39. Wood, L.E.: Semi-structured interviewing for user-centered design, Interactions of the ACM, april-may, 1997, pp 48-61.
40. Vukovie, M.: Crowdsourcing for Enterprises, Congress on Services - I, DOI 10.1109/SERVICES-I.2009.56, IEEE, 2009, pp 686-692.
41. Wu, W., Tsai, W., Li, W.: Creative software crowdsourcing: from components and algorithm development to project concept formations, Inderscience Enterprises Ltd. Int. J. Creative Computing, Vol. 1, No. 1, 2013.