redin Revista Facultad de Ingeniería, Universidad de Antioquia, No. 77, pp. 88-94, 2015

Requirements engineering based on knowledge: a comparative case study of the KMoS-RE strategy and the DMS process

Ingeniería de requisitos basada en conocimiento: un estudio comparativo de la estrategia KMoS-RE y el proceso DMS

Karla Olmos-Sánchez*, Jorge Rodas Osollo[1], Luis Fernández Martínez, Victor Morales Rocha

Centro de Ingeniería del Conocimiento e Ingeniería de Software, Universidad Autónoma de Ciudad Juárez. Av. del Charro 450 Norte. C. P. 32310. Ciudad Juárez, México.

ARTICLE INFO

Received February 27, 2015
Accepted July 8, 2015

KEYWORDS

Requirements engineering, knowledge management, tacit knowledge, empirical software engineering

Ingeniería de requisitos, gestión de conocimiento, conocimiento tácito, ingeniería de software empírica

ABSTRACT: Nowadays, a difficult problem that software development companies are facing in the elicitation and discovering requirement process is the management of tacit knowledge, which is valuable information that for some reason remains hidden to the developers. The Knowledge Management on a Strategy for Requirements Engineering (KMoS-RE) is especially designed to face that problem and obtain a set of requirements that fulfill the clients' needs and expectations. This paper presents the design and preliminary results of an empirical study that compares the KMoS-RE strategy with the requirements elicitation process proposed by MoProSoft; a Mexican software process model oriented to the specific needs of the software industry in Mexico. Preliminary results show that KMoS-RE seems to be more suitable than the before mentioned process proposed by MoProSoft.

RESUMEN: Actualmente la gestión de conocimiento tácito, información valiosa que por alguna razón permanece oculta a los desarrolladores, es un problema que enfrentan las compañías de desarrollo de software en la elicitación y descubrimiento de requisitos. Este problema ocasiona que los productos finales no cumplan con las expectativas y necesidades de los clientes y/o usuarios. La Estrategia de Ingeniería de Requisitos Basada en Conocimiento (KMoS-RE por sus siglas en inglés) fue diseñada especialmente para encarar este problema y obtener un documento de especificación lo más cercano a las necesidades de los clientes y/o usuarios. Este artículo presenta el diseño y los resultados preliminares de un estudio empírico que compara la estrategia KMoS-RE con el proceso de elicitación de requisitos propuesto por MoProSost: Modelo de Procesos de Software orientado a la industria de software en México. Los resultados preliminares muestran que la estrategia KMoS-RE es más adecuada que el proceso de elicitación de requisitos propuesto por MoProSoft.

1. Introduction

In any software development project, errors in the requirements process can be costly in terms of waste of time, loss of reputation, and even company survival [1]. Requirements Engineering (RE) arises as a discipline with the aim of eliciting, analyzing, evaluating, consolidating and managing the requirements of a product or solution [2]. However, there are a great number of serious and inherent difficulties in the process of discovering the correct and appropriate requirements because of the complexity of the requirement task, the intricate interaction between designers and the intended users, and the limits of human information processing [3].

Nowadays, an open problem in RE is the management of tacit knowledge. Any RE process involves a knowledge transfer and transformation process [4], which is not a trivial task: the knowledge of a person must be transformed into natural language and non-verbal channels of human communication in order to be transferred to another person, who then decodes this knowledge according to their own interpretation. Every knowledge transfer involves an explicit and a tacit component [5]. The former is related to theoretical knowledge such as facts and other elements, of which people are aware when thinking. The latter, tacit knowledge refers to personal and context-specific knowledge, which is hard to formalize and communicate. Tacit knowledge affects, at a greater or lesser degree the development of any software project. According to [6], tacit knowledge can cause critical expectations, knowledge and

* Corresponding author: Karla Olmos Sánchez
E-mail: kolmos@uacj.mx
ISSN 0120-6230
e-ISSN 2422-2844

DOI: 10.17533/udea.redin.n77a11

needs of the stakeholders to remain hidden. In addition, the presence of multiple stakeholders increases the problem because the backgrounds, perspectives, interests and expectations (tacit knowledge) of the stakeholders differ depending on their experience and their role in the application domain.

Our main motivation is to provide a methodological frame to the development software companies that can be used in order to confront the problem of achieving a solution or a product closer to the needs of the clients or users. As a result, we have developed the Knowledge Management on a Strategy to Requirements Engineering (KMoS-RE), which is a pattern in a stream of decisions, oriented to the elicitation, transformation and transference of knowledge, with the aim of incorporating this knowledge into a specification document that will minimize the percentage of ambiguous, incomplete and inappropriate requirements [7].

The KMoS-RE strategy is supported by three fundamental ideas: a) It uses a mechanism to represent and understand the application domain, b) it incorporates the SECI (Socialization, Externalization, Combination and Internalization) knowledge management model and c) it manages the tacit knowledge by incorporating methods to identify, capture, index and make explicit the largest possible amount of tacit knowledge. Those characteristics give theoretical arguments about the functionality of the strategy. However, it is necessary to corroborate its usefulness and effectiveness through empirical studies. Thus, we conducted a case study to compare the KMoS-RE strategy with the Software Development and Maintenance Process (DMS, for its acronym in Spanish). Although there are many other requirements methodologies or process, such as TROPOS [8], KAOS [9] or Thecne [10], we selected DMS because it is a well-established process proposed by MoProSoft, which is a Mexican software process model oriented to the specific needs of the software industry in Mexico [11]. The goal of this paper is to present the results of the comparative empirical study.

The remainder of this paper is organized as follows: section 2 introduces the KMoS-RE strategy. In section 3, the DMS process is explained. Section 4 describes the design, data collection, data comparison and results of the case study. Finally, in section 5 the conclusions and future work are presented.

2. KMoS-RE Strategy

The KMoS-RE strategy is designed to provide a systematic way to elicit, structure and create knowledge, both tacit and explicit, which can be incorporated into a product or solution. For this purpose, the strategy is based on the following premise: "To develop a software system, it is necessary to understand the requirements, and to educe the requirements, it is necessary to understand the domain" [12]. The strategy is composed by three sequential phases: domain modeling, system modeling and specification developing [13]:

- *Domain Modeling Phase* (DM). The first phase of the strategy aims to formalize the domain properties by making explicit concepts, attributes, relationships between concepts and basic integrity restrictions. A lexicon is used to identify, classify and define the terms of the domain. Once the lexicon is developed, it is used to build the conceptual model.

- *System Modeling Phase* (SM). In this paper, the word system denotes a set of components interacting with each other to satisfy some global objectives. The requirements engineers should model two versions of the system: the system as it exists before the deployment of a solution (current system), and the system as it should be when the solution is will be operated in it (future system). The aim of this phase is to formalize the current and future system processes. The Use Cases Model is used to model both the current system and the future one, and the information used to develop this model is derived from the lexicon and the Conceptual Model.

- *Specification Development Phase* (SD). In this phase, the requirements engineers derive the set of requirements of the future system from the Uses Cases. These requirements will be used to build the specification document.

In order to manage knowledge, the KMoS-RE strategy includes transversal activities to identify tacit knowledge using discourse analysis techniques, such as analysis of presuppositions [14] and Bloom's Taxonomy [15]. In addition, the strategy incorporates two new artifacts: a record of wrong beliefs and a piece of knowledge matrix to keep track of the tacitness grade of concepts, relationships and behaviors for every participant in the project; i.e. domain specialists and requirements engineers. The Figure 1 depicts the general perspective of the KMoS-RE strategy.

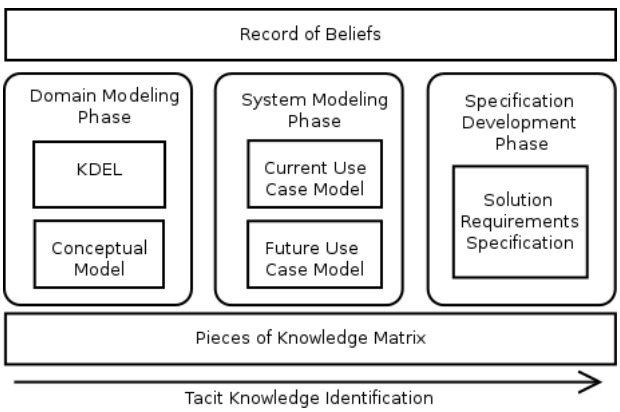

Figure 1 General perspective of the KMoS-RE strategy

The KMoS-RE strategy is structured according to the Knowledge Evolution Model for Requirements Engineering (KEM-RE), which is an iterative cycle based on the SECI model. The SECI model postulates four iterative conversion modes: *Socialization*, the process of transferring tacit

knowledge between individuals by sharing mental models and technical skills; *externalization*, the process of converting tacit knowledge to explicit through the development of models, protocols and guidelines; *combination*, the process of recombining or reconfiguring existing bodies of explicit knowledge to create new explicit knowledge; and *internalization* the process of learning by task repetition and wherein individuals will absorb the knowledge as tacit knowledge again [16].

The KEM-RE is an iterative cycle that consists of four stages (Figure 2): 1) Knowledge Elicitation & Creation (KE&C), where the requirements engineers elicit knowledge from domain specialists; 2) Knowledge Integration and Application (KI&A), where the requirements engineers generate models using the acquired knowledge and their solution space knowledge; 3) Knowledge Sharing and Exchange (KS&E), where the models developed by requirements engineers will be discussed with the domain specialists, and 3) Knowledge Validation (KV), where the domain specialists validate the models. This iterative process leads to the elicitation and creation of new knowledge.

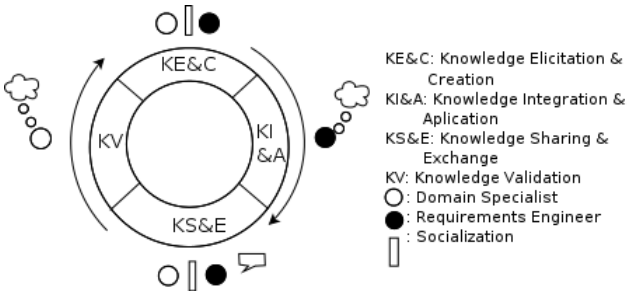

Figure 2 Knowledge Evolution Model for Requirements Engineering (KEM-RE)

Figure 3 shows a partial view of the KMoS-RE in an UML activity diagram in order to emphasize how the activities are structured according the KEM-RE. The details of the KMoS-RE strategy can be consulted in [7].

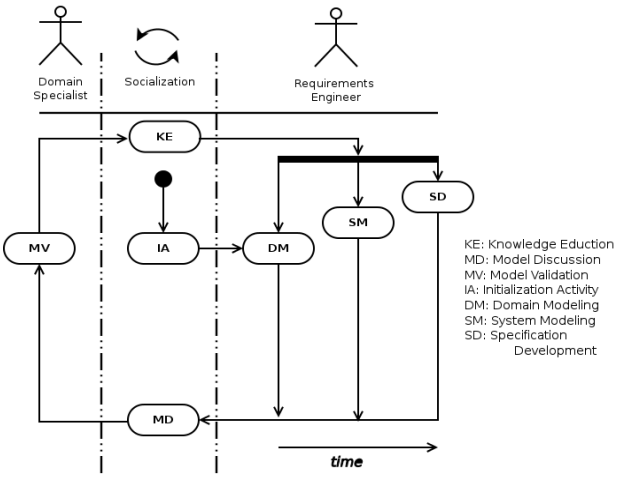

Figure 3 UML activity diagram of the KMoS-RE strategy

3. DMS Process

MoProSoft is a Mexican software process model oriented to the specific needs of the software industry in Mexico [11]. MoProSoft groups their processes into three categories: the *Top Management*, which includes practices related to business management; the *Management Category*, which includes process, project, and resource management practices; and the *Operations Category*, which addresses the practices of software development and maintenance projects. The last category includes the *Administration of Specific Project Process* and the *Software Development and Maintenance (DMS for its acronym in Spanish)* process.

The purpose of the DMS process is to develop, in a systematic way, the activities of requirements elicitation, analysis, design, codification, integration and test of the software product that fulfills the specified requirements. DMS is composed of one or more development cycles and every cycle is composed of the following phases of beginning, requirements, analysis and design, construction, testing and integration, and closing.

The KMoS-RE strategy is focused on the generation of the Requirements Specification. However, in the process, some artifacts of analysis and design are generated, such as the Case Use Model and the Conceptual Model. Therefore, there is a correspondence between the products generated by the KMoS-RE strategy and those generated by the requirements and analysis and design phases of the DMS process.

4. Case Study

Case studies are particularly important in software development area. However, to avoid bias and assure reliability, a systematic process is necessary to follow. The research in this paper is based on the recommendations of [17].

The objective of the case study was to compare the KMoS-RE strategy with the requirements and analysis and design phases of the DMS process in order to obtain evidence of 1) the elicitation of unambiguous functional requirements, 2) the effectiveness of managing tacit knowledge and 3) the improvement of the negotiation process. Thus, the research questions are the followings:

a) Does the KMoS-RE strategy obtain more unambiguous functional requirements in comparison with DMS Process?

b) Does the KMoS-RE strategy elicit more domain knowledge in comparison with DMS Process?

c) Does the KMoS-RE strategy improve the negotiation process in comparison with DMS Process?

d) Does the KMoS-RE strategy generate a more complete specification document in comparison with DMS Process?

4.1. Case Study Design

For the purpose of the research, three software development projects were selected as units of analysis:

- *Cognitive Rehabilitation System.* The goal of this project was to develop a software system that automates the diagnostic evaluation and rehabilitation of cognitive impairment in patients with multiple sclerosis.
- *Virtual Classroom.* This project aimed to develop a software system that manages the processes of a classroom. This system will be used by the teachers of the courses to manage information about the students, such as attendance, grades, assessment...
- *Remote Monitoring of Diabetes Patients.* The goal of this project is to develop a software system that will be used by specialized physicians to receive medical information about their patients through mobile devices.

For each project, two software development teams were selected. One team used the KMoS-RE strategy and the other used the DMS process. The team that used the KMoS-RE strategy was composed of graduate students guided by us. The other team was composed of engineers of four years experience who guide their activities according to MoProSoft. In order to facilitate the comparison, the standard specification IEEE SA-830, the Use Cases Model and the Conceptual Model were requested to both teams. Furthermore, the teams recorded for each activity: the kind of activity, the participants, the time spent on each activity, and the observations.

The case study design consisted of a set of propositions taken *a priori*. The way to measure them and the way of data collection is shown in Table 1.

Table 1 Case study design

Proposition	Measure	Data Collection
The KMoS-RE strategy closes the symmetry of ignorance. Thus, it minimizes the negotiation time.	– Negotiation time. – Better understanding.	– Record of work sessions. – Direct observation.
The KMoS-RE strategy forces the identification and definition of terms. Thus, it facilitates the conceptual model development.	– Number of entities, relationships and attributes in the conceptual model.	– Conceptual model.
The KMoS-RE strategy is based on a knowledge management theory. Thus, it accelerates the knowledge transference.	– Negotiation time. – Better understanding.	– Record of work sessions. – Direct observation.
The KMoS-RE strategy incorporates tacit knowledge identification techniques. Thus, it produces requirements more appropriate, correct and unambiguous.	– Numbers of uses cases. – Relationships between use cases and actors. – Functional requirements And unambiguous requirements.	– Specification document. – Use cases model.
The KMoS-RE strategy obtains the functional requirements from the future use cases model. Thus, it increases the completeness of specification document and minimizes the time of subsequent modifications.	– Number of functional and non-functional requirements. – Number of ambiguous requirements.	– Record of specification document development time. – Specification document.

4.2. Data collection

After the teams delivered the artifacts, they were analyzed in order to obtain a summary of the major elements of each one. For the Use Case Model, it was considered the number of use cases, actors and relationships as the meaningful elements. For the Conceptual Model, the number of entities and the relation of entities were considered as the main ones. Finally, for the Specification Document, the number of functional requirements and the ambiguous functional requirements were considered. In order to identify ambiguous requirements the Bloom's Taxonomy, a

Table 2 Comparison of artifacts

	Cognitive Rehabilitation		Virtual Classroom		Remote Monitoring	
Use Cases Model						
	DMS	**KMoS**	**DMS**	**KMoS**	**DMS**	**KMoS**
Number of use cases	7	37	11	15	9	8
Number of actors	3	3	3	2	3	4
Number of relationships	5	8	17	6	13	12
Conceptual Model						
	DMS	**KMoS**	**DMS**	**KMoS**	**DMS**	**KMoS**
Number of entities	9	18	10	15	7	9
Number of relationships between entities	9	23	10	15	10	12
Specification Document						
	DMS	**KMoS**	**DMS**	**KMoS**	**DMS**	**KMoS**
Number of functional requirements	9	85	46	54	21	39
Number of ambiguous functional requirements	1	0	2	1	1	0
Work Session Time						
	DMS	**KMoS**	**DMS**	**KMoS**	**DMS**	**KMoS**
Time of work sessions	0:50	1:20	3:40	3:20	1:10	2:05

framework for classifying statements of what it is expected or intended for students to learn as a result of instruction, was used. In Bloom's Taxonomy the categories are ordered from simple to complex and from concrete to abstract, and the higher levels subsume lower levels. The use of Bloom's Taxonomy to identify tacit knowledge has been explored in the development of Knowledge Management Systems [15]. If a verb is in the higher order of the taxonomy, then the domain specialists are referencing a critical thinking and the statement could be ambiguous and abstract. Table 2 summarizes that information and the accumulate time of the work sessions for each project.

4.3. Data comparison

The data comparison was made according to the perspective of the domain knowledge, the system projection, the specification document features and the negotiation time, as explained below:

- *Domain knowledge.* A domain is composed of entities, attributes, relationships, functions and states. Although the conceptual model represents only the structural view, it is a partial indicator of the elicited domain knowledge. As can be seen in Table 2, in all projects, the number of entities and relationships was greater when the KMoS-RE strategy was used. In particular in the case of the Cognitive Rehabilitation System, the number of entities and relationships was doubled. This case had a great quantity of informality because the client did not have a clear idea of the software solution. In the other two projects, the clients had a better understanding about the domain and the software functionality requested. Even so, the number of concepts and relationships was greater.

- *System projection.* The future use cases model was evaluated according the number of actors, the number of use cases and the descriptions of the use cases. As can be seen in Table 2, the number of use cases identified in the projects of Cognitive Rehabilitation and Virtual Classroom was higher when the KMoS-RE strategy was used. In the case of the Remote Monitoring project, the number of use cases was very similar in the two teams. However, it should be emphasized that the use cases identified by the team who used the DMS process were more related to the use of technology than to the application domain. For example, in the Virtual Classroom project, people using DMS process did not detect the basic and evident use case "Calculate grade". In relation with the actors, in the project of Virtual Classroom, the team who followed the KMoS-RE strategy did not identify an additional actor: the "super administrator". The interviews were reviewed and this actor was not explicitly required by the client. So, the actor was introduced in the model by the requirements engineers who used DMS process. Conversely, in the Remote Monitoring project, the team that used the KMoS-RE strategy identified an additional actor. The team that used DMS process did not detect that the "Monitoring Center Staff" and the "System Manager" were in fact, two different roles that required different actors. Finally, the number of relationships between the use cases in the projects of Remote Monitoring and Cognitive Rehabilitation

were similar. In the project of Virtual Classroom the number of relationships was increased significantly. However, the use case model was vague and poorly structured.

- *Specification document.* As well as the number of functional requirements is an estimate of the fulfillment of the projection of the system; the number of ambiguous requirements is an estimate of the depth of the elicited domain knowledge. In all the projects was evident that with the use of the KMoS-RE strategy, the number of functional requirements was increased. As a result, the specification document was more complete and appropriate. The number of ambiguous requirements according to Bloom's Taxonomy was very small. In the case of the team who used the DMS Process, this could be due to the small quantity of requirements in the specification.

- *Negotiation time.* The time wasted in the work session is an estimate of the understanding between the domain specialists and requirements engineers. The time spent on the working sessions were very similar in the three projects, which indicates that the use of the KMoS-RE strategy did not increase significantly the development time.

4.4. Case study results and discussion

The DMS process is an iterative cycle based on a waterfall process of software development, which starts with the requirements specification and finalizes with the product delivery, as it is explained in section 3. The case study showed that performing the requirements specification document in first place causes incorrect and ambiguous requirements. In addition, as a whole, the document is incomplete. These errors could be inherited to later stages of the development project causing a product that not fulfills the needs and expectations of clients or users or a extend development time.

The KMoS-RE strategy's fundamental ideas (Section 1) close the symmetry of ignorance between clients and users, and developers. This fact facilitates the transference and transformation of knowledge and improves the negotiation. On the other hand, the KMoS-RE strategy requires generating the current system model, for which it is necessary to understand the domain. Thus, the conceptualization of the future system is more direct because of the understanding of the domain. Finally, the requirements specification document is obtained directly of the future system model. As the case study showed, this characteristic causes that the requirements specification document contains more unambiguous functional requirements.

In summary, in comparison with the DMS process, the KMoS-RE strategy improved the negotiation process, obtained more and unambiguous functional requirements and elicited more tacit knowledge. In addition, using the KMoS-RE strategy did not increase the development time.

5. Conclusions and future work

We recognize the value of MoProSoft to give order and improve the software development companies' processes. However, the case study showed that applying new perspectives and proposals of requirements engineering could enhance it. Especially, to minimize the risk and uncertainty that occur when developers select the specific activities of each phase of the DMS process, in particular, of the requirements engineering process.

We are aware of the limitations of case study as a research method. In the particular case of this research, the most important limitations are: 1) the case study is limited to the requirements elicitation process. 2) The case study only evaluates the products. It is necessary to conduce another case study that considers the stakeholders' opinions. And, 3) we guided the teams that used the KMoS-RE strategy. Although, we did not intend to prove the usability of the strategy, but its functionality and efficiency, it is necessary to conduce another case study that use the KMoS-RE strategy without our guide.

However, the application of the strategy in different projects provides empirical insights about the usefulness and the value of the KMoS-RE strategy. As a future work, it will be necessary to apply the KMoS-RE strategy to more projects in order to obtain more evidence of its utility. In addition, it is necessary to work in how to integrate KMoS-RE to MoProSoft.

6. References

1. L. McLeod and S. MacDonell, "Factors that affect software systems development project outcomes: A survey of research", *ACM Computing Surveys (CSUR)*, vol. 43, no. 4, pp. 24-56, 2011.
2. S. Hansen, N. Berente and K. Lyytinen, "Requirements in the 21st century: Current practice and emerging trends", in *Design requirements engineering: A ten-year perspective*, 1st ed., K. Lyytinen, P. Loucopoulos, J. Mylopoulos and W. Robinson (eds). Cleveland, USA: Springer, 2009, pp. 44-87.
3. L. Pilat and H. Kaindl, "A Knowledge Management Perspective of Requirements Engineering", in *5th International Conference on Research Challenges in Information Science* (RCIS), Guadeloupe, France, 2011, pp. 1-12.
4. A. Distanont, H. Haapasalo, M. Vaananen and J. Lehto, "The engagement between knowledge transfer and requirements engineering", *International Journal of Management, Knowledge and Learning*, vol. 1, no. 2, pp.

131-156, 2012.
5. M. Polanyi, *The tacit dimension*, 1st ed. Garden City, USA: Doubleday, 1966.
6. R. Gacitua *et al.*, "Making Tacit Requirements Explicit", in *2nd International Workshop on Managing Requirements Knowledge* (MARK), Atlanta, USA, 2009, pp. 40-44.
7. K. Olmos and J. Rodas, "KMoS-RE: knowledge management on a strategy to requirements engineering", *Requirements Engineering Journal*, vol. 19, no. 4, pp. 421-440, 2014.
8. P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia and J. Mylopoulos, "Tropos: An Agent-Oriented Software Development Methodology", *Autonomous Agents and Multi-Agent Systems*, vol. 8, no. 3, pp. 203-236, 2004.
9. R. Darimont and A. Lamsweerde, "Formal Refinement Patterns for Goal-Driven Requirements Elaboration", *ACM SIGSOFT Software Engineering Notes*, vol. 21, no. 6, pp. 179-190, 1996.
10. I. Jureta, A. Borgida, N. Ernst and J. Mylopoulos, "Towards a New Generation of Requirements Modeling Languages with Goals, Preferences, and Inconsistency Handling", in *18th IEEE International on Requirements Engineering Conference*, Sydney, Australia, 2010, pp. 115-124.
11. H. Oktaba, "MoProSoft®: A Software Process Model for Small Enterprises", in *1st International Research Workshop for Process Improvement in Small Settings*, Pittsburgh, USA, 2006, pp. 93-101.
12. Dines Bjørner, "Rôle of domain engineering in software development. Why current requirements engineering is flawed!", in *7th International Andrei Ershov Memorial Conference on Perspectives of Systems Informatics* (PSI), Novosibirsk, Russia, 2009, pp. 2-34.
13. G. Hadad, "Uso de escenarios en la derivación de software", Ph.D. dissertation, National University of La Plata, La Plata, Argentina, 2008.
14. L. Ma, B. Nuseibeh, P. Piwek, A. Roeck and A. Willis, "On presuppositions in requirements", in *2nd International Workshop on Managing Requirements Knowledge* (MARK), Atlanta, USA, 2009, pp. 27-31.
15. M. Mitri, "Applying tacit knowledge management techniques for performance assessment", *Computers & Education*, vol. 41, no. 2, pp. 173-189, 2003.
16. I. Nonaka and H. Takeuchi, "The knowledge-creating company: how japanese companies create the dynamics of innovation", *Hardvard Business Review,* vol. 69, no. 6, pp. 96-104, 1995.
17. P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering", *Empirical Software Engineering*, vol. 14, no. 2, pp. 131-164, 2009.