Knowledge management applied to software testing: A systematic mapping

Article · January 2013

CITATIONS
6

READS
147

3 authors, including:

Erica Ferreira de Souza
Federal Technological University of Paraná, Brazil
22 PUBLICATIONS **87** CITATIONS
SEE PROFILE

Ricardo de Almeida Falbo
Universidade Federal do Espírito Santo
172 PUBLICATIONS **1,661** CITATIONS
SEE PROFILE

Knowledge Management Applied to Software Testing: A Systematic Mapping

E. F. Souza
Computação Aplicada
Instituto Nacional de Pesquisas Espaciais, INPE
São José dos Campos/SP, Brasil
erica.souza@lac.inpe.br

R. A. Falbo
Departamento de Informática
Universidade Federal do Espírito Santo, UFES
Vitória/ES, Brasil
falbo@inf.ufes.br

N. L. Vijaykumar
Lab. de Comp. e Matem. Aplicada
Instituto Nacional de Pesquisas Espaciais, INPE
São José dos Campos/SP, Brasil
vijay@lac.inpe.br

***Abstract* - With the growth of data from several different sources of knowledge within an organization, it becomes necessary to provide computerized support for tasks of acquiring, processing, analyzing and disseminating knowledge. In the software process, testing is a critical factor for product quality, and thus there is an increasing concern in how to improve the accomplishment of this task. In software testing, finding relevant knowledge to reuse can be a difficult and complex task, due to the lack of a strategy to represent or to associate semantics to a large volume of test data, including test cases, testing techniques to be applied and so on. This paper aims to investigate, through a Systematic Mapping of the Literature, some aspects associated with applying Knowledge Management to Software Testing.**

Keywords: Software Testing, Knowledge Management, Systematic Mapping

I. INTRODUCTION

Software development is an error prone process. To achieve quality software products, it is essential to perform Verification & Validation (V&V) activities throughout the software development process. Verification determines whether the development products of a given activity conform to the requirements of that activity. Validation refers to whether the software satisfies its intended use and the user needs [1]. V&V activities can be static and dynamic. Dynamic V&V activities require the execution of a program, while static V&V activities do not. Static V&V are typically done by means of technical reviews and inspections. Dynamic V&V are done by means of testing [2]. Thus, Software Testing consists of the dynamic V&V of the behavior of a program on a finite set of test cases, against the expected behavior [3].

Due to advances in technology and the emergence of increasingly critical applications, tests have become more and more complex. Currently, software testing is considered a process consisting of activities, techniques, resources and tools. During software testing, a large number of information is generated. In fact, software testing is a knowledge intensive process, and it becomes necessary to provide computerized support for tasks of acquiring, processing, analyzing and disseminating knowledge for reuse [4].

Finding relevant knowledge in software testing is not an easy task. There is a need to represent and process knowledge in an affordable and manageable manner. In this context, principles of Knowledge Management (KM) are pointed out as an important means to manage software testing knowledge [5].

The main goal of KM is to promote knowledge storage and sharing, as well as the emergence of new knowledge [6]. This paper presents a systematic mapping of the literature in order to identify the primary studies that applied principles of KM to software testing. A systematic mapping provides a broad overview of an area of research, to determine whether there is research evidence on a particular topic. Results of such mapping may identify suitable areas for performing systematic reviews and also areas where a preliminary study is more appropriate. A systematic mapping also helps identifying gaps in order to suggest areas for future research and provides a map that allows appropriately to position new research activities [7].

The systematic mapping presented in this paper investigates the following issues: (i) problems related to knowledge in software testing; (ii) organizations' purposes of applying KM in software testing; (iii) types of knowledge items typically managed in the context of software testing; (iv) benefits and problems reported on the implementation of KM initiatives in software testing; and (v) mechanisms or technologies used to provide KM in software testing. This mapping was structured in five research questions, and 336 studies were selected and analyzed according to a systematic mapping method.

This paper is organized as follows. Section 2 presents the main concepts used in this paper. Section 3 describes the systematic mapping method applied, and discusses the main parts of the mapping protocol used, including research questions, inclusion and exclusion criteria, searched sources and search string. Section 4 presents the main results of the mapping, discussing the selection process, the classification schemas, and presenting data synthesis. Section 5 discusses the findings and the mapping limitations. Finally, Section 6 presents conclusions and future directions for this research.

II. BACKGROUND

In this section, we discuss briefly some of the most important concepts in the research areas studied (namely Software Testing and KM), in order to characterize the scope of our investigation and to support the definition of the research questions that are the subject of the systematic mapping.

Software Testing activities are supported by a well-defined and controlled test process [3]. Testing process concerns to how tests can be conducted and managed. It involves phases, activities, artifacts, techniques, procedures, resources and tools that seek to control and organize tests, in order to achieve high-quality software [2, 3, 8, 9].

As the software development process becomes more complex, testing process also becomes increasingly complex and prone to generate a lot of information. Such information may turn into useful knowledge to potentially benefit future projects from experiences gained from previous projects [4]. However, converting this information into applicable knowledge is not an easy task. There is a need to properly represent and process the knowledge so that it can be accessible and manageable. In this context, Knowledge Management (KM) principles can be applied.

Different KM approaches have been applied in the context of software testing to promote reuse of knowledge generated in the testing process. Given this context, we conducted a systematic mapping of the literature aiming at synthesizing the evidences related to KM in software testing.

III. RESEARCH PROTOCOL

The research method for this systematic mapping was defined based on the guidelines for systematic literature reviews given in [7]. A systematic mapping helps providing a wide overview of a research area and identifying areas suitable for conducting Systematic Literature Reviews and areas where a primary study is more appropriate. It involves three main phases [7]: (i) **Planning**: refers to the pre-review activities, and aims at establishing a review protocol defining the research questions, inclusion and exclusion criteria, sources of studies, search string, and mapping procedures; (ii) **Conducting:** regards searching and selecting the studies, in order to extract and synthesize data from them; (iii) **Reporting:** is the final phase and aims at writing up the results and circulating them to potentially interested parties. Following, the main parts of the mapping protocol used in this work are presented.

A. Research Questions

This mapping aims at answering the following research questions:

RQ1. What are the problems reported by software organizations related to knowledge about software testing?

RQ2. What are the purposes of employing KM in software testing?

RQ3. What are the types of knowledge items typically managed in the context of software testing?

RQ4. What are the main conclusions (benefits and problems) reported on the implementation of KM initiatives in software testing?

RQ5. What are the mechanisms or technologies used to provide KM in software testing?

B. Inclusion and Exclusion Criteria

The selection criteria are organized in one inclusion criterion (IC) and five exclusion criteria (EC). The inclusion criterion is: (IC1) The study discusses KM applied to software testing. The exclusion criteria are: (EC1) The study does not have an abstract; (EC2) The study is just published as an abstract; (EC3) The study is not written in English; (EC4) The study is an older version (less updated) of another study already considered; and (EC5) The study is not a primary study, such as editorials, summaries of keynotes, workshops, and tutorials.

C. Sources

The search was applied in seven electronic databases that were considered the most relevant according to [10]. They are:

IEEE Xplore (*http://ieeexplore.ieee.org*)
ACM Digital Library (*http://dl.acm.org*)
SpringerLink (*http://www.springerlink.com*)
Scopus (*http://www.scopus.com*)
Science Direct (*http://www.sciencedirect.com*)
Compendex (http://www.engineeringvillage2.org)
ISI of Knowledge *(http://www.isiknowledge.com)*

D. Keywords and Search String

The search string considered two areas, Software Testing and KM (Table I), and it was applied in three metadata fields (title, abstract and keywords). The search went through syntactic adaptations according to particularities of each source.

TABLE I. KEYWORDS SEARCH

Areas	***Keywords***
Software Testing	"Software Testing", "Software Test"
KM	"Knowledge Management", "Knowledge Reuse"
Search string: (*"Software Testing" OR "Software Test") AND ("Knowledge Management" OR "Knowledge Reuse")*	

E. Data storage

The publications returned in the searching phase were cataloged and stored appropriately. This catalog helped us in the classification and analysis procedures.

F. Assessments

Before conducting the mapping, we tested the mapping protocol. This test was conducted in order to verify its feasibility and adequacy, based on a pre-selected set of studies considered relevant to our investigation. The review process was conducted by one of the authors and the other two carried out its validation. They analyzed 36% of the studies using two different samples.

IV. CONDUCTING THE MAPPING

In this section, the main steps that we performed in this mapping are discussed, namely: search and selection, data extraction and classification, and synthesis and data analysis.

A. Search and Selection

In the search process, we considered the studies published until January 2013. As a result, a total of 336 publications were returned, out of which 53 from **IEEE Xplore**, 67 from **Compendex**, 70 from **Scopus**, 2 from **Science Direct**, 4 from **ACM Digital Library**, 134 from **SpringerLink** and 6 from **Thomson Reuters Web of Knowledge**.

Then, a selection process, divided into 3 stages, was applied on the returned publications. In the first stage duplicates were eliminated based on examining title and abstract. In this step, the number of publications was reduced to 253 (approximately 25% reduction), since many publications were available in more than one source.

In the second step, inclusion and exclusion criteria were applied considering title and abstract. 219 publications (86.5%) were eliminated. Although the publications cite, in the abstract, the terms contained in the search string, they did not have the principles of KM applied in the area of software testing and thus were eliminated by the inclusion criterion (IC1). Finally, in the third phase, the exclusion criteria were applied considering the entire text, resulting in a reduction of 70.5%. It is worth pointing out that, in the third stage, one publication was eliminated because we did not have access to the full text.

From the three stages of the selection process, 10 studies were considered relevant, from which data were extracted. Table II summarizes the stages and their results. It shows the progressive reduction of the number of studies throughout the selection process. 10 out of 336 was the final number, with a reduction rate of about 97%. Table III lists the 10 studies considered relevant.

TABLE II. RESULTS OF THE SELECTION PROCESS STAGES

Stage	Criteria	Analyzed Content	Initial N. of Studies	Final N. of Studies	Reduction (%)
1st	Eliminating duplication	Title and abstract	336	253	25%
2nd	IC1, EC1, EC2, EC3, EC4 e EC5	Title and abstract	253	34	86.5%
3rd	IC1, EC4, EC5 e EC6	Entire Text	34	10	70.5%

B. Data Extraction and Classification

To answer the research questions from the 10 selected studies, we used a form containing some parameters, including the following: id, bibliographic reference, problems, purpose, types of knowledge, and benefits and problems, related with the implementation of KM in software testing. This form was used to extract the answers. Therefore, before the extraction, categories for classifying the studies were defined according to the research questions. So, depending on the focus of each category item, the study was classified as one or any combination of this. Categories were defined as follows.

Classification schema for problems: this is based on the main problems related to knowledge about software testing. We have identified five main categories of problems, namely: (i) Barriers in transferring testing knowledge, (ii) Loss of testing knowledge, (iii) Low reuse rate of testing knowledge, (iv) Testing knowledge is not properly shared, and (v) Testing knowledge is not properly considered for planning the testing process (including human resource allocation to testing activities).

TABLE III. SELECTED STUDIES

ID	Bibliographic reference
#1	Y. Liu, J. Wu, X. Liu, G. Gu "Investigation of Knowledge Management Methods in Software Testing Process," International Conference on Information Technology and Computer Science, v.2, pp. 90 – 94, 2009.
#2	O. K. Wei, T. M. Ying, "Knowledge Management Approach in Mobile Software System Testing," Industrial Engineering and Engineering Management, pp. 2120 - 2123, 2007.
#3	L. Xu-Xiang, Z. Wen-Ning, "The PDCA-based software testing improvement framework," Apperceiving Computing and Intelligence Analysis (ICACIA), pp. 490 - 494, 2010.
#4	R. Abdullah, Z. D. Eri, A. M. Talib, "A Model of Knowledge Management System in Managing Knowledge of Software Testing Environment," Malaysian Conference in Software Engineering (MySEC), pp. 229 – 233, 2011.
#5	X. Li, W. Zhang, "Ontology-based Testing Platform for Reusing," Sixth International Conference on Internet Computing for Science and Engineering, pp. 86 – 89, 2012.
#6	A. Desai, S. Shah, "Knowledge Management and Software Testing," International Conference and Workshop on Emerging Trends in Technology (ICWET), pp. 767-770, 2011.
#7	E. Collins, A. Dias-Neto, V. F. Lucena, "Strategies for Agile Software Testing Automation: An Industrial Experience," 36th Annual Computer Software and Applications Conference Workshops, pp. 440-445, 2012.
#8	J. Andrade, J. Ares, M. Martínez, J. Pazos, S. Rodríguez, J. Romera, S. Suárez, "An architectural model for software testing lesson learned systems," Information and Software Technology, pp. 18-34, 2013.
#9	K. Karhu, O. Taipale, K. Smolander, "Investigating the relationship between schedules and knowledge transfer in software testing," Information and Software Technology, Vol. 51, pp. 663-677, 2009.
#10	K. Nogeste and D. H.T. Walker, "Using knowledge management to revise software-testing processes," Journal of Workplace Learning, v.18, n.1, pp. 6-27, 2006

Classification schema for purposes: we wanted to know what are the organizations' purposes, when employing KM in software testing. We have identified five main categories of purposes: (i) Reuse of knowledge related to software testing (including lessons learned), (ii) Support for decision making, (iii) Cost reduction, (iv) Competitive advantages, and (v) Organizational learning (including also lessons learned).

Classification schema for types of knowledge: This schema shows what types of knowledge are dealt with by organizations and how they are handled. In this case, we analyzed the explicit and tacit knowledge generated by an organization in the context of software testing. Tacit knowledge comes from individual experiences. It is highly personal, hard to formalize and, therefore, difficult to communicate to others. On the other hand, explicit knowledge is formal and systematic. For this reason, it can be easily communicated and shared. It can be expressed as tables, figures, drawings, sketches, diagrams and requirements [11].

Other classification schemes: In these schemes we collected unstructured data without a predefined classification. We looked at the main findings found as benefits and problems related with the implementation of KM in software testing, and the main mechanisms and technologies reported by the selected studies.

C. Synthesis and Data Analysis

Publications over the years: In order to offer a general view of efforts in the area of KM in Software Testing, a distribution of the 10 selected papers over the years is shown in Figure 1. As this figure suggests, KM in Software Testing is very recent, occurring basically from 2006 to nowadays.

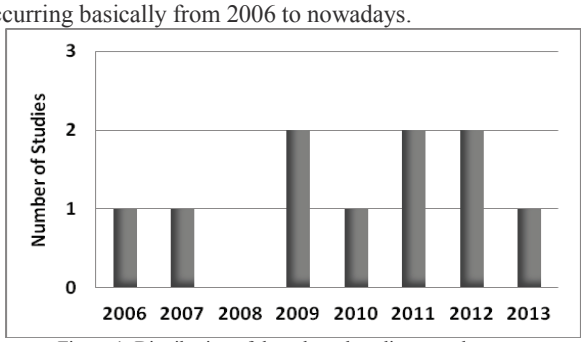

Figure 1. Distribution of the selected studies over the years

Main problems related to knowledge about software testing (RQ1): Figure 2 shows the percentage of studies per category, considering the problems reported by software organizations related to knowledge about software testing. We can notice that "Barriers in Knowledge Transfer" has the largest representativeness (9 studies in 10, corresponding to 90%). It stands out because transfer of organizational knowledge can be quite difficult to achieve. This occurs because most of the knowledge in organizations is tacit, that is, derived from experience, and it becomes difficult to articulate. Another category with a high percentage is "Low Reuse Rate Knowledge" with 60% (6 studies). Software Testing, in general, can involve reusing modules, test cases, components, and experiences. However, testing teams, generally, do not reuse or take advantage on the knowledge acquired or the experience gained. Therefore, the same mistakes are repeated, even though there are individuals in the organization with the knowledge and experience required to stop this [4].

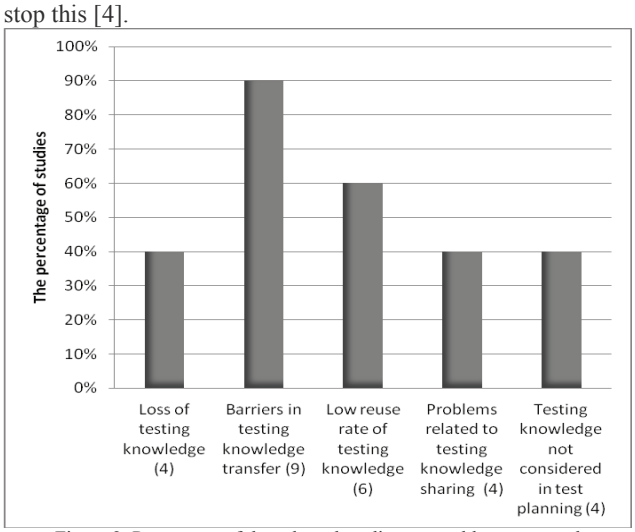

Figure 2. Percentage of the selected studies per problems reported

Main purposes to employ KM in software testing (RQ2): Figure 3 shows the percentage of studies per category, considering the organizations' purposes in managing software testing knowledge. We can notice that "Knowledge Reuse" (10 studies – 100%), "Organizational Learning" (7 studies – 70%) and "Competitive Advantages" (6 studies – 60%) have the largest representativeness. We should highlight that some purposes identified are strongly related. For instance, lessons learned are both a way to promote knowledge reuse and organizational learning. Thus, studies reporting that one of the purposes of applying KM in software testing is registering and disseminating lessons learned (5 studies – 50%) were considered in both categories. Knowledge reuse, in turn, helps increasing test effectiveness and thus leads to competitive advantages and cost reduction.

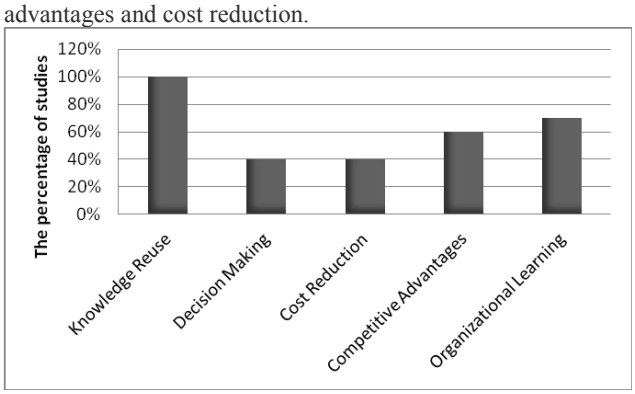

Figure 3. Percentage of the selected studies per purposes reported

Types of knowledge typically managed (RQ3): Knowledge can be of two main types: tacit and explicit knowledge. In the 10 selected studies, both of them are considered. Tacit knowledge is taken into account in all studies (100%), whereas 7 studies (70%) consider also explicit knowledge. Explicit knowledge appears mainly as test artifacts. Some examples of explicit knowledge are: Test Plan, Test Cases, Test Results, Requirements Specification, and Conceptual Models. Out of all these examples, Test Cases are the most common cited artifact in most of the literature evaluated.

Most of the studies identify that tacit knowledge is more difficult to acquire, as part of personal experiences by the members of the test team. They also mention that tacit knowledge can be acquired from discussions, experiences from project members, questionnaires and communications.

Main conclusions (benefits and problems) reported on the implementation of KM initiatives in software testing (RQ4): Regarding this issue, we have to highlight that 4 studies, although discussing some aspects related to KM in software testing, they do not report KM initiatives in software testing. In fact, only six of them (studies #1, #2, #3, #5, #8, and #10 in Table III) discuss KM initiatives in software testing. From these studies, we identified the following conclusions from employing KM in software testing:

- **Major problems found:** (i) Employees are normally reluctant to share their knowledge; (ii) if, on top of this,

knowledge sharing increases the employee workload, KM strategies fail; and (iii) the existing communication systems are not appropriate.

- **Primary benefits found:** (i) Selection and application of better suited techniques, methods and test cases; (ii) Cost reduction; (iii) Increasing test effectiveness; and (iv) Competitive advantages.

Mechanisms or technologies used (RQ5): From the selected studies, 6 of them use a KM system (one using a general purpose one), and 4 present KM models or architectures devoted to KM in software testing. From the mechanisms and technologies applied by them, two highlight: yellow pages (or knowledge maps) are used in 4 studies (40%); and ontologies are used in 3 studies (30%).

V. RESULTS DISCUSSION

In this section we discuss some important findings and limitations of this mapping.

Several studies have reported the problem of knowledge reuse within organizations. The main issue is that knowledge is retained with a single individual and therefore becomes more difficult to raise this knowledge to the organizational level. Even when some knowledge management strategy is applied, it is not always feasible to achieve organizational learning because the employees are reluctant to share their knowledge as they feel that retaining this knowledge is an advantage over their colleagues [4].

Several studies report on the use of a KM system (#1, #2, #3, #5, #8, #10). Others propose knowledge management models or architectures (such as #1, #4, #5, #8). A KM system should support the integration of information from disparate sources, wherein a decision maker manipulates information that someone else conceptualized and represented. So, the KM system must minimize ambiguity and imprecision in interpreting shared information. This can be achieved by representing the shared information using ontologies [12]. Although ontologies have been widely recognized as an important technology for KM, only three studies (#1, #5, and #8) use ontologies. More specifically, only one (#5) uses an ontology of the software testing domain. This seems to be a problem, since, as pointed by Staab et al. [13], ontologies are the glue that binds KM activities together, allowing a content-oriented view of KM. Ontologies define shared vocabulary to be used in the KM systems to facilitate communication, integration, search, storage and knowledge representation [14].

With respect to the types of knowledge, both tacit and explicit have been investigated in the literature. According to [11], as expected, tacit knowledge is more valuable, it is hard to be acquired, and it requires good strategies to acquire and process this knowledge. However, results obtained from it are rich. Therefore, tacit knowledge is important to the test team as it refers to previous experiences and thorough analysis of past projects [15]. In this context, yellow pages or knowledge maps are considered important tools for managing testing knowledge, and the some studies cite their importance, such as #1, #5, and #8.

During the mapping it was possible to infer that much of the explicit knowledge was related to reuse of test cases derived from documents considered complete and correct. According to [16], more detailed information on test cases can provide a greater learning. As test cases evolve in applications, they may be changed for a variety of reasons. Thus an efficient and effective KM process can help in evaluating the impact and in conducting changes of the test cases.

As we can see by means of this mapping, there are many benefits of implementing KM in organizations for managing software testing knowledge:

- **Selection and application of better suited techniques, methods and test cases.** Experience plays a key role in testing, and managing past experience helps to effectively tailor the techniques and methods to the ongoing project. Some of these techniques, such as White-box Testing Techniques, Black-box Testing Techniques or Defect-based Testing Techniques [2], depend on the knowledge, experience and intuition of test analyst.
- **Cost reduction.** In the testing context, cost has a very strong relationship with time. Tester experience is crucial for designing test cases and regression test selection. A good selection of test cases minimizes not only costs but also reduces time [17].
- **Test effectiveness increase.** Knowledge and experience about the domain and the system under test is essential for increasing test effectiveness. This helps testers improve decision making on which techniques to use, selection of test cases or approaches for test input generation [4].
- **Competitive advantages.** In organizations, KM is now seen as a strategic factor and knowledge is also recognized as one of the main sources of cost savings and competitive advantage [18]. The ability to transfer best practices in the organization is a means to build competitive advantage through the appropriation from scarce knowledge [19].

Although KM in software testing brings many benefits, there are also problems, such as:

- **Employees are normally reluctant to share their knowledge:** Many experiences are grasped by only a few people and haven't become public knowledge. This causes many difficulties in knowledge transfer about testing [4].
- **Increased workload:** Shortage of time is a potential risk to incorporate the principles of KM in software testing, because knowledge sharing can imply in increasing the employee workload and costs [4, 18].

- **KM systems are not appropriate yet:** There are many difficulties in implementing knowledge acquisition, coding, storage and searching functionalities effectively in KM system, because it involves all the problems mentioned above as time and interest of the employees.

The mapping conducted in this study also had some limitations. In order to reduce subjectivity, the other two authors made a random validation of 36% of the studies. We did several tests with the search string to try our best not to compromise the return of some preliminary studies. We cannot say what is the best technique applied, but the objective was to map how the principles of KM has evolved in the software testing domain over the years. Furthermore, the results may be different if conducted to another area of application different from the testing software.

VI. CONCLUSIONS

This paper presented a systematic mapping in the context of software testing and KM. Five research questions were defined and addressed investigating the following aspects: (i) main problems found related to knowledge in software testing; (ii) purposes to employ KM in software testing; (iii) types of knowledge typically managed in the context of software testing; (iv) main conclusions (benefits and problems) reported on the implementation of KM initiatives; (v) mechanisms or technologies used in KM in software testing.

The contributions of this work are on making evident some aspects associated to the employment of KM in software testing and research efforts that can drive future research. In this context, we highlight the following conclusions: (i) the major problem in organizations are barriers in knowledge transfer with largest representativeness; (ii) reuse of testing knowledge is the main purpose of applying KM in software testing; (iii) there is a great concern with tacit knowledge.

Implementation of KM strategies in the field of software testing has shown very promising research, since KM helps in handling knowledge within the organization in several respects as shown in this systematic mapping. However, a point seems to be a challenge for KM in software testing. Although recognized as an important instrument by the KM community [12, 13, 14], ontologies are not being widely used in KM initiatives in software testing. Thus, as future work, we intend to explore how ontologies can be used for managing knowledge in the software testing domain.

ACKNOWLEDGMENT

The first author would like to acknowledge FAPESP (Process: 2010/20557-1) for the financial grant. The second author acknowledges FAPES (Process Number 52272362/11) for the financial grant.

REFERENCES

[1] IEEE Std 1012-2004: IEEE Standard for Software Verification and Validation. New York, NY, USA. pp. 120, 2004.

[2] A. P. Mathur, Foundations of Software Testing. 5rd ed. Delhi, India: Dorling Kindersley (India), Pearson Education in South Asia, 2012.

[3] IEEE Computer Society, SWEBOK, A Guide to the Software Engineering Body of Knowledge, 2004.

[4] J. Andrade, J. Ares, M. Martínez, J. Pazos, S. Rodríguez, J. Romera, S. Suárez, "An architectural model for software testing lesson learned systems," Information and Software Technology, pp 18-34, 2013.

[5] A. Desa, S. Shah, "Knowledge Management and Software Testing," in International Conference and Workshop on Emerging Trends in Technology (ICWET 2011) – TCET, Mumbai, India, 2011.

[6] D. O'Leary and R. Studer, "Knowledge management: An interdisciplinary approach," IEEE, vol. 16, No. 1, 2001.

[7] B, Kitchenham, S. Charters, "Guidelines for performing Systematic Literature Reviews in Software Engineering," School of Computer Science and Mathematics Keele University and Departament of Computer Science University of Durham, UK, v. 2.3, 2007.

[8] G. J. Myers, The Art of Software Testing. 2rd ed. Canada: John Wiley and Sons, 2004.

[9] R. Black, J. L. Mitchell. Advanced Software Testing: guide to the ISTQB advanced certification as an advanced technical test analyst. USA:Oreilly & Assoc, 2008.

[10] T. Dyba, T. Dingsoyr, G. Hanssen, "Applying systematic reviews to diverse study types: An experience report," First International Symposium on Empirical Software Engineering and Measurement, Madrid, pp. 225-234, 2007.

[11] I. Nonaka, H. Takeuchi, The Knowledge-Creating Company. How Japanese Companies Create the Dynamics of Innovation, Oxford University Press, Oxford, 1999.

[12] H. M. Kim, "Developing Ontologies to Enable Knowledge Management: Integrating Business Process and Data Driven Approaches," Workshop on Bringing Knowledge to Business Processes, 2000.

[13] S. Staab, R. Studer, H. P. Schurr, and Y. Sure, "Knowledge Processes and Ontologies," IEEE Intelligent Systems, vol. 16, No. 1, 2001.

[14] V. R. Benjamins, D. Fensel, and A. G. Pérez, "Knowledge Management through Ontologies," The 2nd International Conference on Practical Aspects of Knowledge Management (PAKM98), Switzerland, 1998.

[15] K. W. Ong, M.Y. Tang, "Knowledge management approach in mobile software system testing," in Proceedings of the IEEE International Conference on Industrial Engineering and Engineering Management, IEEE IEEM 2007, Singapore, pp. 2120–2123, 2007.

[16] X. Li, W. Zhang, "Ontology-based Testing Platform for Reusing," Sixth International Conference on Internet Computing for Science and Engineering, pp. 86- 89, 2012.

[17] A. Beer, R. Ramler, "The role of experience in software testing practice," in Proceedings of the 34th Euromicro Conference Software Engineering and Advanced Applications, Italy, pp. 258–265, 2008.

[18] O. Taipale, K. Karhu, K. Smolander. "Observing Software Testing Practice from the Viewpoint of Organizations and Knowledge Management," In Empirical Software Engineering and Measurement (ESEM), 2007. pp. 21-30, 2007.

[19] K. Karhu, O. Taipale, K. Smolander, "Investigating the relationship between schedules and knowledge transfer in software testing," Information and Software Technology, Vol. 51, pp. 663-677, 2009.