Using the Findings of a Mapping Study to Conduct a Research Project: A Case in Knowledge Management in Software Testing

Conference Paper · August 2015

DOI: 10.1109/SEAA.2015.10

CITATIONS
2

READS
102

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

2015 41st Euromicro Conference on Software Engineering and Advanced Applications

Using the findings of a mapping study to conduct a research project: a case in knowledge management in software testing

Érica F. Souza
Department of Computer
Federal Technological University
of Paraná, UTFPR
Cornélio Procópio/PR, Brazil
ericasouza@utfpr.edu.br

Ricardo A. Falbo
Department of Computer Science
Federal University of Espírito
Santo, UFES
Vitória/ES, Brazil
falbo@inf.ufes.br

Nandamudi L. Vijaykumar
Lab. of Comp. and Applied Math.
National Institute for Space
Research, INPE
São José dos Campos/SP, Brazil
vijay.nl@inpe.br

***Abstract*—A mapping study provides a broad overview of a research area in order to determine whether there is research evidence on a particular topic. Results of a systematic mapping may identify suitable areas for performing future research. In this paper, we discuss our experience in using the findings of a mapping study on Knowledge Management (KM) in Software Testing for performing a real research project, which also applied other empirical approaches. The main goals of this paper are: (i) to reinforce the importance of a systematic mapping in the conduction of a research project by discussing a real case of such application, and (ii) to present the results of our survey on the most important aspects of KM when applied to software testing.**

Keywords*—*Mapping Study, Systematic Mapping, Survey, Knowledge Management, Software Testing

I. Introduction

An important criticism that can be directed to researchers in Software Engineering is that they make little or no use of the methods and experiences available from other studies [1], [2]. In order to avoid such shortcoming, Systematic Literature Review (SLR) methodology should be used for constructively supporting software engineering research [3].

A mapping study (also known as systematic mapping) is a kind of SLR whose purpose is to identify and categorize the available research on a broad software engineering topic [3]. Mapping studies can be of significant benefit to researchers in establishing baselines for further research activities. They provide a body of knowledge that researchers can use in a variety of ways, such as [3]: (i) the starting point for undertaking conventional SLRs; (ii) a baseline against what research trends can be tracked over time; (iii) a justification for further primary studies when there are few (or no) relevant empirical studies; (iv) a baseline for empirical research of various kinds; and (v) an education resource.

In 2010, we saw the opportunity to conduct a research on Knowledge Management (KM) in Software Testing. Software testing importance is widely recognized, and there is a growing concern in how to improve the accomplishment of this process. In this context, software testing knowledge can be systematically collected, stored in an organizational repository, and shared across the organization [4]. Thus, KM emerges as an important means to manage software testing knowledge, and to improve the software testing process.

After informally studying this research topic, in late 2011, we decided to perform a mapping study on KM in Software Testing to identify the state of the art in the area, and thus to guide our research efforts. The results of this mapping study were published in [5]. From the findings of the mapping study, we noticed that the application of KM strategies in the field of software testing were a very promising research area, since KM helps in handling testing knowledge within software organization in several aspects. On the other hand, KM in software testing is still incipient, given that the number of primary studies addressing this topic is low. Moreover, the mapping also showed us that ontology-based KM solutions are even rarer in the software testing domain. This finding attracted our attention, since ontologies are recognized as an important instrument for supporting KM in general [6]. This fact motivated us to undertake this research project in ontology-based KM in software testing.

In this paper, we discuss how the findings of the mapping study guided our research efforts, which also applied other empirical approaches. From the mapping study, we performed two SLRs, developed a Reference Ontology on Software Testing (ROoST) [7], and performed a survey to define a scenario to apply KM in software testing. The survey aimed at identifying, among others, the testing activities in which KM is more useful, testing artifacts more appropriate for reuse, and types of testing knowledge more important for performing the testing process. From the results of the suvey, we established the most appropriate scenario in the software testing domain, from the point of view of testing stakeholders, for applying KM. Finally, considering the survey results, and based on ROoST, we developed a KM system, called Testing Knowledge Management Portal (TKMP), for supporting managing testing knowledge repository as a proof of concept for the project.

It is worthwhile to point out that the main contribution of this paper is to present a real case in which a systematic mapping had a great importance in the conduction of a research project. Although several authors advocate in favor of using mapping studies as a baseline for further research activities, such as [3] and [8], there are few works describing real cases of

978-1-4673-7585-6/15 $31.00 © 2015 IEEE
DOI 10.1109/SEAA.2015.10

such a successful application. Since our goal in this paper is to discuss how the mapping study guided our efforts, we discuss: (i) results of our research that have already been published ([5], [7], [9], [10]); and (ii) results that have not been published yet as a survey. With respect to the latter, we discuss in details these results of the survey on the most important aspects of KM applied to software testing.

The remainder of this paper is organized as follows. Section II presents the main concepts related to the background of this paper. Section III presents the main findings of the mapping study, and discusses how they were used to guide our research project. In this section we also discuss the survey performed to define a scenario to apply KM in software testing. Section IV discusses related work. In Section V, we present our final considerations.

II. BACKGROUND

Software testing comprises a set of activities with the main objective to contribute to the quality of the software products. Software Testing should be supported by a well defined and controlled testing process and the main activities of this process are: Test Planning, Test Case Design, Test Coding, Test Execution and Test Result Analysis [11], [12]. First, testing should be planned. Key aspects of test planning include, among others, planning and scheduling testing activities, and defining the test environment for the project. Test planning is documented in a Test Plan. Test case design aims at designing the test cases to be run. Test cases are documented and implemented. During test execution, test cases are run, producing results, which are then analyzed to determine the result (passed or failed). Software testing involves also levels, artifacts, techniques, procedures, resources and tools that seek to control and organize tests, in order to achieve high-quality software [12], [13], [14].

During the testing process, a variety of knowledge is necessary, ranging from knowledge on the application domain to knowledge on testing techniques. This makes software testing a knowledge intensive process, and it is useful to provide automated support for tasks of acquiring, processing, analyzing and disseminating knowledge for reuse [4]. In this context, Knowledge Management (KM) emerges as an important means to manage software testing knowledge, and to improve the software testing process. The main goal of KM is to promote knowledge storage and sharing, as well as the emergence of new knowledge [15]. KM has an objective in developing and leveraging organizational knowledge, by making knowledge created by organization members available.

In this context, one of the main problems is how to represent knowledge. A KM system should minimize ambiguity and imprecision. This can be achieved by using ontologies [16]. Ontologies are particularly important to support KM. They bind KM activities together, allowing a content-oriented view of KM [17]. Ontologies define the shared vocabulary to be used in order to facilitate knowledge communication, representation and storage, integration, and search [6].

The Software Engineering community has recognized the need for managing knowledge and that it could learn from the KM community [18]. Software development is a quickly changing, knowledge-intensive business, involving many people working in different phases and activities [19]. Knowledge in Software Engineering is diverse and organizations have problems in identifying its content, location, and use. An improved use of this knowledge is the basic motivation and driver for KM in Software Engineering. As a consequence, KM in Software Engineering has been subject of deeper analyses, such as those conducted by Rus and Lindvall [19], and by Bjørnson and Dingsøyr [18].

As a sub-area of Software Engineering, Software Testing also presents the same features, and knowledge can be applied to different testing tasks and purposes [20]. Given the applicability of KM to improve the software testing process, we decided to conduct a research project in this topic. However, for advancing the research on a sound basis, we needed to systematically inspect the literature through a mapping study.

Systematic Literature Review (SLR) is an essential methodology to be used constructively to provide support to software engineering research. In general, SLRs are secondary studies used to find, critically evaluate and aggregate all relevant research studies in a software engineering topic. A mapping study is a form of SLR. Mapping studies use the same basic methodology as SLRs, but they identify and classify all research related to a broader topic. They are intended to provide an overview of a topic area and identify whether there are sub-topics with sufficient primary studies to conduct conventional SLRs and also to identify sub-topics where more primary studies are needed [3].

III. FROM A MAPPING STUDY TO A KM SYSTEM

The efforts of our research project in KM applied to Software Testing started with a systematic mapping of the literature. The main goals of this mapping were to make evident some aspects associated to the employment of KM in software testing, and to identify challenges and gaps that could drive our research [5].

We considered studies published until December 2013, and we used the following search string, which was applied in seven electronic databases: *("software testing" OR "software test" OR ("software project" AND ("test" OR "testing")))* AND *("knowledge management" OR "knowledge reuse" OR "knowledge sharing" OR "knowledge transfer")*. We selected 562 publications, out of which 77 from **IEEE Xplore**, 86 from **Compendex**, 95 from **Scopus**, 4 from **Science Direct**, 19 from **ACM Digital Library**, 270 from **SpringerLink**, and 11 from **Thomson Reuters Web of Knowledge**. Then we followed a selection process comprising five stages, as shown in Figure 1.

In the 1^{st} stage, we eliminated publications that appear in more than one source, achieving 440 studies. In the 2^{nd} stage, we applied the selection criteria (inclusion and exclusion criteria) over title, abstract and keywords, resulting in 43 papers. In the 3^{rd} stage, the selection criteria were applied considering the full text, resulting in a set of 13 studies. Over these 13 studies considered relevant, we moved to the 4^{th} stage to perform snowballing. Snowballing is a process that checks if the selected studies cite other relevant studies, retrieve those studies, and continue this process until no more relevant studies are found [21]. Snowballing resulted in 8

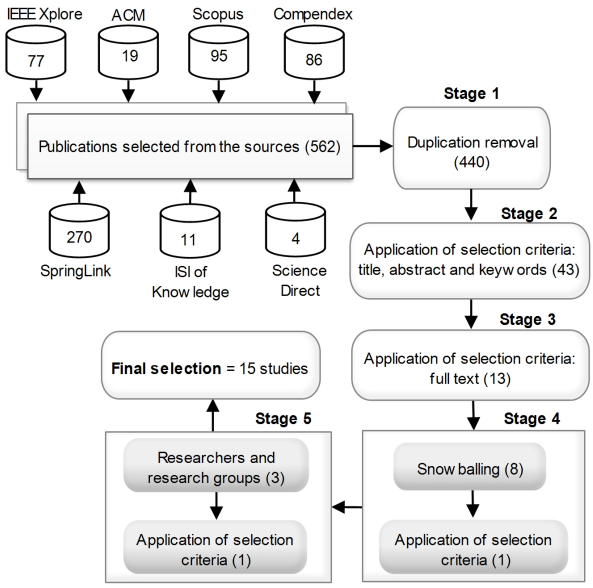

Fig. 1. Steps of research project from a mapping study.

papers. After applying the selection criteria, only 1 papers remained. Finally, from the 14 papers selected until then, in the 5^{th} stage, we looked for publications authored by the researchers and research groups involved in these studies. In this stage, we searched for their personal pages, their entries in the DBLP Computer Science Bibliography, as well as other publications authored by them in the digital libraries that we used as sources for this mapping. 3 papers were selected from the same research group, but 1 was eliminated by the exclusion criteria. As the final result, we ended up with 15 studies to be analyzed. For details regarding the mapping, see [5].

The main conclusions achieved by the mapping were the following [5]: (i) the major problem in software organizations related to software testing are low knowledge reuse rate and barriers in knowledge transfer; (ii) reuse of test cases is the perspective that has received more attention; and (iii) there is a great concern with both explicit and tacit knowledge. In particular, one finding drew our attention: only three studies discuss ontologies in the context of KM applied to software testing. Moreover, only two actually use ontologies in a KM initiative applied to software testing ([22], [23]). This seems to be a contradiction, since in the KM literature, ontologies have been widely recognized as an important instrument for knowledge communication, representation and storage, integration, and search [24], [6], [17]. This finding motivated us to further investigate ontology-based approaches for KM in software testing. Thus, as the next step, we investigated ontology-based approaches for KM in software testing by means of a Systematic Literature Review (SLR). The same search string and electronic databases were used. This approach greatly facilitated our work. As a result, we retrieved only two studies, namely [22] and [23]. This SRL showed us that only one study uses ontology about the software testing domain [22] (the other [23] uses ontology about the "knowledge" domain). In both the studies, ontologies are used at development time, mainly for structuring knowledge repositories.

Based on the findings of the SLR, we decided to look for ontologies on the software testing domain in the literature, to select one to be used in our research project. We looked for ontologies in the software testing domain and the context that it was applied. For systematically inspecting the literature, another SLR was performed, which is published in [10]. 12 ontologies addressing this domain were identified. To analyze these ontologies, we considered some of the characteristics pointed out by D'Aquin and Gangemi [25] as characteristics that are presented in "beautiful ontologies". In our analysis, we considered the following characteristics: (i) having a good domain coverage; (ii) implementing an international standard; (iii) being formally rigorous; (iv) implementing also non-taxonomic relations; (v) following an evaluation method; and (vi) reusing foundational ontologies.

As the main findings obtained from this SLR, we can highlight [10]: most ontologies have limited coverage; the studies do not discuss how the ontologies were evaluated; none of the analyzed testing ontologies is truly a reference ontology; and, finally, although foundational ontologies have been recognized as an important instrument for improving the quality of conceptual models in general, and more specifically of domain ontologies [25], none of the ontologies analyzed in our SLR is grounded in foundational ontologies. In summary, we concluded that the software testing community should invest more efforts to get a well-established reference software testing ontology [10]. This motivated us to build ROoST - a Reference Ontology on Software Testing [7].

ROoST has been developed in a modular way. Currently, ROoST covers aspects related to the Software Testing Process and its Activities, Artifacts that are used and produced by those activities, Testing Techniques for test case design, and the Software Testing Environment, including hardware, software and human resources. In order to evaluate ROoST, we performed ontology verification & validation activities. ROoST evaluation started with a verification activity, where we checked if the defined concepts, relations and axioms were able to answer the competency questions posed to the ontology. To validate ROoST, we instantiated its concepts and relations with individuals extracted from an actual project, to check if the ontology was able to represent concrete situations of the real world.

The mapping study showed us that one of the major challenges in managing testing knowledge is how to effectively integrate KM with software testing so that knowledge items can be shared and reused. Furthermore, managing testing knowledge is not an easy task, and thus it is better to start with a small-scale initiative. So, we decided to perform a survey in order to define a scenario for applying KM in software testing. The purpose of our survey is to identify which is the most appropriate scenario in the software testing domain, from the point of view of testing stakeholders, for starting a KM initiative. As the data generation method, we decided to use a questionnaire, with 9 questions, to be applied on the Web. To establish the survey sample, groups interested in software testing registered in the *LinkedIn*[1] network were invited to answer the survey. 86 experts answered the survey. Out of these 86 participants, 50 work directly with software testing (tester,

[1] http://www.linkedin.com/

test analyst, test designer, among others); 19 perform roles related to software development (system analyst, programmer, etc.); and 17 perform other functions, such as professor, researcher, and consultant. Participants have an average of 5 years of experience. However, if we consider only the experts in software testing, 50% of them have more than five years of experience. Table I shows the number of experts by experience time in years.

TABLE I. NUMBER OF EXPERT BY EXPERIENCE TIME IN YEARS

Time (t) in years	Number of experts
$t < 3$ years	12 (24%)
3 years $\leq t < 5$ years	13 (26%)
t $\geq$ 5 years	25 (50%)

The questions defined in the survey are related to the mapping study, as well as to the conceptualization described by ROoST, as shown in Table II. Following, we present the main results of the survey.

TABLE II. RELATIONSHIPS BETWEEN THE SURVEY QUESTIONS (SQ) WITH THE MAPPING RESEARCH QUESTIONS (RQ) AND ROOST

Survey Questions	*Based on*
SQ1. In which activities of a Testing Process is KM more useful? ***SQ2.*** In which activities of Testing Planning is KM more useful?	**ROoST**: Testing Process and Activities sub-ontology
SQ3. A test environment consists of, among others, human resources, hardware and software. About which of these resources are more important to have available knowledge at the moment of defining the test environment?	**ROoST**: Testing Environment sub-ontology
SQ4. In which testing level is KM more useful?	**ROoST**: Testing Techniques sub-ontology
SQ5. What is the type of knowledge you consider to be more important during the software testing process?	**Mapping Study**: *RQ7* (What are the types of knowledge items typically managed in software testing?)
SQ6. Regarding the types of knowledge items listed below, indicate the importance of generating explicit knowledge from tacit knowledge.	**Mapping Study:***RQ7*
SQ7. Regarding testing artifacts, which are the ones you judge to be more appropriate for reuse?	**Mapping Study:** *RQ7* **ROoST:** Testing Artifacts sub-ontology
SQ8. What is the purpose of applying KM in Software Testing?	**Mapping Study:** *RQ6*. (What are the purposes of employing KM in software testing?)
SQ9. What benefits KM can bring to software testing?	**Mapping Study:** *RQ9*. (What are the main benefits and problems reported regarding applying KM in software testing?)

Concerning *SQ1*, as Figure 2 shows, in general, "Test Case Design" (98.8%) and "Test Planning" (96.5%) were considered the testing activities in which KM is considered most useful. The activities listed as possible answers for this question (see Figure 2) were extracted from ROoST. We also performed an analysis taking the position into account, i.e. we analyzed the answers from experts who work directly with software testing (testers). 90% of the 50 testers considered also the "Test Result Analysis" a testing activity in which KM is very useful.

Regarding *SQ2*, as Figure 3 shows, "Testing Technique Selection" (41%) and "Test Environment Structuring" (36%) were considered the testing planning activities in which KM is most useful. With respect to *SQ3*, as Figure 4 shows, "Human resource" and "Software Resource" are considered the resources from which it is most important to have knowledge

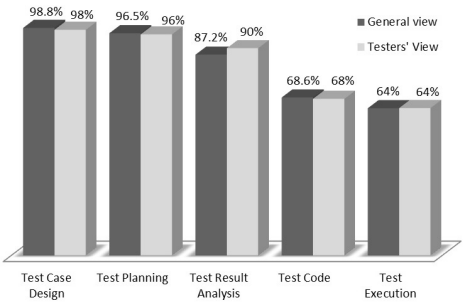

Fig. 2. Importance of KM to Software Testing Process Activities.

available at the time of setting the test environment. Like SQ1, the possible answers for this question (see Figure 4) were extracted from ROoST.

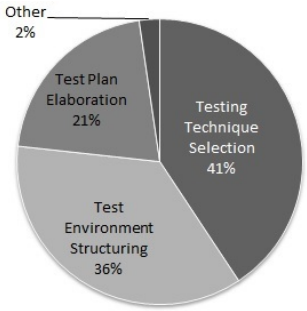

Fig. 3. Useful of KM in activities of Testing Planning Sub-activities.

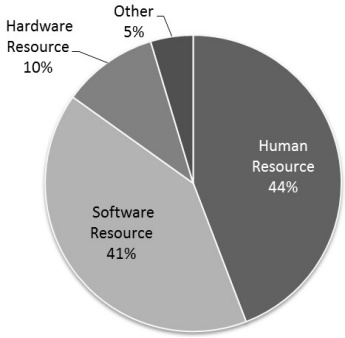

Fig. 4. Importance of KM to Testing Environment Definition.

With respect to the test level (*SQ4*), in general "System Testing" (49%) was considered the one in which KM is most useful. The options listed as possible answers for this question (see Figure 5) were also extracted from ROoST. This view is also corroborated by the opinion of the testers, which considered system testing even more important (58%).

SQ5, SQ6 and *SQ7* were based on the research question RQ7 of the mapping study that investigated which types of knowledge items are typically managed in the context of software testing. We tried to reproduce the same aspects investigated in the mapping to see if the survey participants confirm or deny the findings of the mapping. In the survey, the participants considered explicit knowledge (69.8%) more important than tacit knowledge (30.2%). The same occurs

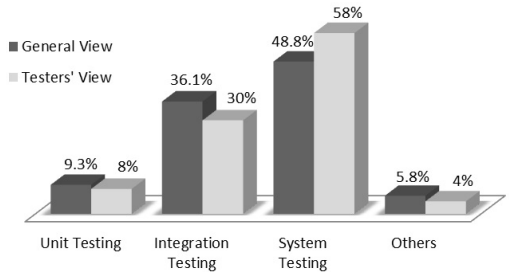

Fig. 5. Importance of KM to Test Levels.

when analyzing only the responses of testers (explicit knowledge (68%) and tacit knowledge (32%)). Among the types of tacit knowledge items that seem to be most important to be made explicit (*SQ6*), two stand out, as Figure 6 shows: "Individual Experiences" (95.3%) and "Communications between the members of the test team" (92%). However, for testers, "Discussions" (94%) are considered more important than "Communications between the members of the test team" (82%). Regarding testing artifacts, "Test Plan" (91.9%) and "Test Case" (90.7%) were considered the most reusable ones, as Figure 7 shows. For testers, the most reusable artifacts are "Test Cases" (94%).

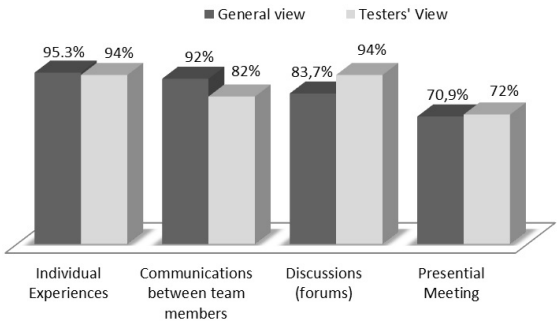

Fig. 6. Making tacit knowledge explicit.

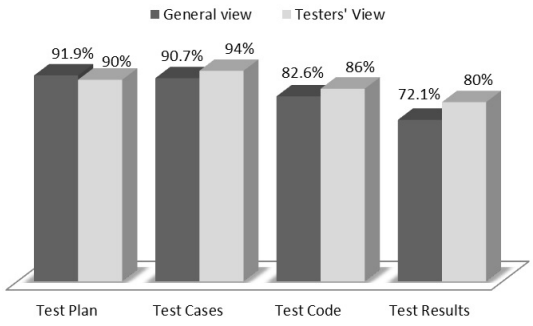

Fig. 7. Artifacts considered more appropriate for reuse.

Finally, *SQ8* and *SQ9* were based on *RQ6* and *RQ9* of the mapping study, and intend to investigate how the purposes (RQ6) and the benefits of applying KM in software testing (*RQ9*) pointed out in the mapping study were perceived as important by the survey participants. Considering the opinion of all the participants, "Improving the quality of results" (28%) and "Reducing costs, time and effort" (26%) are the main purposes of applying KM in software testing; for testing experts "Improving the quality of results" (30,5%) and "Supporting decision-making process" (25,3%) are the main purposes of applying KM in software testing (see Figure 8). Regarding the benefits, according both to the opinion of all the participants and of the testing experts, "Increasing the testing process efficiency" (41%; 44%) and "Selecting and applying more suitable testing techniques" (33%; 34%) are the main benefits of applying KM in software testing (see Figure 9).

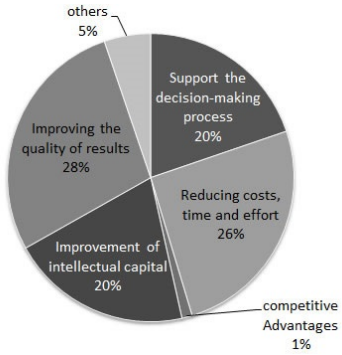

Fig. 8. Purpose of applying KM in Software Testing.

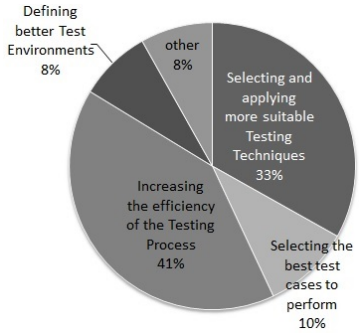

Fig. 9. Expected benefits of applying KM in Software Testing.

From the survey results, we achieved the following conclusions:

- The participants identified test case design and test planning as being the activities in which KM would be most useful. As a consequence, test cases and test plans are considered the most useful artifacts to be reused. These results are in line with the mapping findings, which points out that "Test Case" is the testing artifact that is managed in most cases.
- Explicit knowledge (69.8%) was considered more important than tacit knowledge (30.2%). This corroborates what we raised in the systematic mapping: there is a difficulty in working with tacit knowledge. Moreover, it explains the great interest in explicit knowledge in the studies analyzed in the mapping. In the mapping study, all the 12 selected studies have discussed explicit knowledge, while tacit knowledge is discussed in only 8 studies.

The results of the survey and the results of the mapping were then combined to define the scope of an initial testing KM

initiative. Considering the main findings of the survey and the mapping, test case design was considered the software testing activity to be first supported, and test cases the main knowledge item to be managed. All relevant information for designing test cases has also to be considered in the scope of a testing KM initiative, such as test case input, expected result, actual result, test code and testing technique. As a proof of concept of the ideas developed in our research project, we decided to develop a KM system supporting the scope defined, called Testing Knowledge Management Portal (TKMP). TKMP is a web application that was developed using ROoST for structuring its knowledge repository.

TKMP's knowledge repository was populated with 1568 test cases extracted from an actual project, called Amazon Integration and Cooperation for Modernization of Hydrological Monitoring Project (ICAMMH Project). Other test cases from another actual project called On-Board Data Handling (OBDH), inside Inertial Systems for Aerospace Application (SIA) Project, were also inserted in TKMP's knowledge repository. TKMP was then evaluated by the testing leaders of these two projects. Both stressed the importance of such a system to support the software testing process, in particular, to critical systems such as the ones from which the test cases were extracted.

With respect to the comments of the ICAMMH project testing leader, it was pointed out that TKMP would be extremely useful for the project. As the project dealt with a significant number of human resources, there was a loss of knowledge due to the high turnover rate of the testing team members. Therefore, according to the leader, a KM system such as TKMP would be beneficial for supporting test case reuse in similar situations in different modules and even in other projects.

With respect to the OBDH project, the leader's evaluation about TKMP was that such a portal would be very useful while dealing with critical systems. However, it was pointed out that the main problem is that the organizations that deal with software testing must be open minded in order to change their culture and be ready not only to accept new concepts and tools, but also to implement such new ideas. Moreover, in the development of critical systems, organizations generally tend to be more conservative, and the adoption of new tools and methodologies is made more slowly than in other software organizations.

It is important to notice that this evaluation is quite preliminary, as the ICAMMH project had already finished when the evaluation was done, and the testing activities of the OBDH project were only in an initial phase.

IV. Related work

In [3], in order to assess the value of mapping studies, Kitchenham et al. use a multi-case, participant-observer case study using five studies that were preceded by mapping studies. The research question addressed by this case study is: "How do mapping studies contribute to further research?" As a result, Kitchenham et al. have identified some benefits that can accrue from basing research on a preceding mapping study, and associated problems. In this sense, our work followed a similar approach to the studies analyzed in [3]. As in our case, these studies used the results of the mapping as basis for follow-on research activities. However, two of them start with tertiary studies, and thus are not quite similar to ours. Another study (the one referring to a mapping study on visualization techniques) has not yet been published, and thus it was not possible to compare it with ours. Following, we discuss the other two studies.

In [26], Pretorius and Budgen performed a mapping study of UML empirical studies. The results they got encouraged them to extend the original study into a SLR. Thus, as a follow on research activity, Budgen et al. [27] extended the study to include papers published up to the end of 2008, and, for the categories that had sufficient material, they have also analyzed the relevant studies in detail, making a more extensive assessment of the available knowledge about the effectiveness of the UML, as determined by empirical studies. Analogously to this study, the results of our mapping led us to perform a SLR to investigate in details ontology-based KM in software testing. Moreover, this extension led us to investigate, by means of another SLR, ontologies in the software testing domain.

In [28], Zhang and Budgen conducted a mapping study to investigate how extensively the use of software design patterns has been subjected to empirical study and what evidence is available about how and when their use can provide an effective mechanism for knowledge transfer about design. The follow-on activity was to review observational studies on design patterns. Subsequent review found little additional information, so the researchers undertook an online survey of pattern users. Potential respondents were identified from the authors of the primary studies found in the mapping study. Analogously to this study, we had also undertaken an online survey based on the mapping. However, instead of identifying potential respondents from the authors of the primary studies, we used insights from the mapping to formulate some of the survey questions.

It is worth pointing out that several advantages, raised by the researchers who answered the questionnaire in [3], were also perceived by us, notably the following:

- The follow-on SLR was less time consuming. Since the studies were already selected from the mapping, the first SLR we performed for investigating in details the ontology-based approaches to KM in software testing was greatly facilitated.
- A mapping makes it easier for researchers to understand the literature and construct research questions. In our case, the mapping was essential for guiding the research efforts, in special in defining research questions to be investigated.
- A mapping study may indicate that more primary studies are needed. In our case, gaps were identified, as well as ways forward. Especially, the mapping study pointed us that there are very few studies using ontology-based approaches for KM in software testing.
- Previous study provides a set of known studies against which to assess search strings. In our case, a retrieved

study that used a software testing ontology for structuring a testing knowledge repository was added to the control group used to assess the search string of the second SLR we performed to investigate existing ontologies in the software testing domain.

- Procedures, forms and experiences can be reused. In our case, the experience gained with the mapping on KM in software testing was essential for the two SLRs carried out in the sequel. Procedures and forms were reused and adapted in these SLRs.
- Results from the original study can act as a baseline for comparison with the results of the follow-on study. In our case, we used the results of the mapping to select studies to compare to our approach, as discussed below.

Kitchenham et al. [3] have also identified some problems of basing follow-on research activities on mapping studies, namely: (P1) If there is a time lag between the follow-on activities, another search may be necessary adding to the time and effort required for the follow-on activity; (P2) If the search was restricted in any way, a broader search will be required if the follow-on activity is an SLR; (P3) If the individual papers are not fully referenced and there is no clear link between each paper and its classification, the results of the mapping study may be of limited importance to new researchers to have an overview of how much literature needs to be read, and who are the important researchers in their field of interest; (P4) Basing mapping studies on papers, not individual studies (as required for SLRs), means that papers reporting the same study and papers reporting multiple studies will not be clearly identified. This means the number of papers may be a poor indication of the number of primary studies; (P5) Detailed procedures may change as a result of problems experienced in the initial mapping study. This may make comparisons between the original study and the follow-on study difficult to report accurately and may make the results difficult to interpret; (P6) Flaws with the original study may also affect the follow-on research; (P7) If the original mapping study was of poor quality, the entire search process may need to be redone.

In general, we got around these problems. For instance, to avoid P1, follow-on research activities were done as soon as we finished the mapping. P2 did not occur, since our string was general enough for providing a basis for the follow-on SLR. To avoid P3, we clearly identify each paper and how it is classified. P4 did not occur in our mapping. However, in the SLR on software testing ontologies, where there were different papers presenting parts of the same ontology, we considered each ontology as a study, as required by a SLR. P5 did not occur in our case. To avoid P6, we performed a survey to confirm or deny some of the conclusions we got in the mapping. In general, the results from the survey corroborate the mapping results.

It is important to notice, however, that several difficulties experienced by the participants of the five case studies reported in [3] were also faced by us. For instance, we have problems with some classifications, which had to be reviewed. The first version of the mapping missed papers found by snowballing, which were included later. Moreover, we decided to search for studies from researchers and research groups, as suggested in [3], increasing the coverage of our mapping study.

As aforementioned, results from the mapping study were used as a baseline for comparison with the results of our KM initiative in software testing. These results point out that test case reuse has been the main focus of the recent research. For instance, in [23] and [29], test case reuse is the main focus.

In [23], Li and Zhang propose a reusable test case KM model for supporting test case reuse. This model is based on an ontology of reusable test cases. However, this ontology has a poor coverage when compared with the Reference Ontology on Software Testing (ROoST) we have developed. This study neither mentions the use of international standards as basis for their ontology, nor which references were used as basis for developing the ontology. Moreover, nothing is said about how the ontology was evaluated. ROoST, in contrast, is a heavyweight modular ontology that was built considering several references, including international standards. ROoST covers several aspects related to software testing. In order to evaluate ROoST, we performed ontology verification & validation activities. For more details regarding a comparison between ROoST and the ontology of reusable test cases proposed in [23], see [10].

In the other study [29], Janjic and Atkinson present an automated test recommendation approach that proactively makes test case suggestions while an engineer is developing tests. They developed a prototype of an automated, non-intrusive test recommendation system called Test Tenderer. A test case search engine, called SENTRE, drives the test recommendation tool. The search engine uses the current test case to perform a search for reusable, semantically matching components. Name-based searches are used, trying to match the name of the test to the name of a class included in the same package as the test. Analogously to [29], test case design was considered the software testing activity to be supported by TKMP. However, Test Tenderer addresses unit testing, while TKMP is more general. Although Janjic and Atkinson say that SENTRE performs "a search for reusable, semantically matching components", the heuristics applied are based on name-based searches. In TKMP, in turn, the knowledge repository is structured based on ROoST, which is also used as basis for the search functionality. Finally, Test Tenderer works non-intrusively in the background and smoothly integrates into normal working environments. Thus, the developers' normal working practices are not disturbed and they only need to break away from the task of writing new test cases to consider already existing tests suggested by the recommendation engine. TKMP, in the other hand, does not proactively suggest test cases. Testers have to make a query for retrieving similar test cases.

V. Conclusions

In this paper, we discussed the use of the findings of a mapping study for guiding a research project in Knowledge Management (KM) applied to software testing. The mapping helped us to identify and categorize the available research related to initiatives of KM in Software Testing, and showed us a gap in ontology-based approaches for managing testing knowledge. A Reference Ontology on Software Testing (ROoST) was then developed. Based on ROoST and the results

of the mapping, we carried out a survey for defining the most appropriate scenario for an initial testing KM initiative. Test case design was considered the software testing activity to be supported. A KM system, the Testing KM Portal (TKMP), was then developed to support this scenario. Actual test cases were included in TKMP, which was evaluated by two experts.

The main contribution of this paper is to present a real case in which a systematic mapping had a great importance in the conduction of a research project. Our experience corroborates what is said by authors that advocate in favor of using mapping studies as a baseline for further research activities, such as [3] and [8]. Among the benefits of this use, we can highlight that the mapping study established a solid baseline that served as an important guide for our research efforts. In particular, it showed us a gap in the research topic concerning ontology-based approaches for KM in software testing, and that more primary studies in this topic were needed. The identification of this gap was essential for us to define the research questions to be investigated, and to define the research strategies to follow. Moreover, the mapping gave us several studies as inspiration, and also as a baseline for comparison. Finally, we could confirm some of the mapping findings with the opinion of testing stakeholders, by performing a survey. In the case of the survey, no published before, the mapping findings again helped us, in this case, for defining the survey questions.

Concerning the survey, it is important to highlight that the survey is also a contribution of this work, since it provides important insights and evidences for further research in KM applied to software testing.

ACKNOWLEDGMENT

Authors of this paper would like to thank: the Brazilian Aeronautics Institute of Technology (ITA); the Brazilian Agency of Research and Projects Financing (FINEP) - Project FINEP 5206/06 - ICA-MMH; and On-Board Data Handling (OBDH) Project, inside Inertial Systems for Aerospace Application (SIA) Project for providing the data for this project research.

REFERENCES

[1] R. Glass, V. Ramesh, and I. Vessey, "An analysis of research in computing disciplines," *Commun. ACM*, vol. 47, no. 6, pp. 89–94, 2004.

[2] P. O. Brereton, B. A. Kitchenham, D. Budgen, M. Turner, and M. Khalil, "Lessons from applying the systematic literature review process within the software engineering domain," *Journal of Systems and Software*, vol. 80, no. 4, pp. 571–583, 2007.

[3] B. A. Kitchenham, D. Budgen, and O. P. Brereton, "Using mapping studies as the basis for further research: a participant-observer case study," *Journal of Information and Software Technology*, vol. 53, pp. 638–651, 2011.

[4] J. Andrade, J. Ares, M. Martínez, J. Pazos, S. Rodríguez, J. Romera, and S. Suárez, "An architectural model for software testing lesson learned systems," *Journal of Information and Software Technology*, vol. 55, pp. 18–34, 2013.

[5] E. F. Souza, R. A. Falbo, and N. L. Vijaykumar, "Knowledge management initiatives in software testing: A mapping study," *Journal of Information and Software Technology*, vol. 57, pp. 378–391, 2014.

[6] V. R. Benjamins, D. Fensel, and A. G. Pérez, "Knowledge management through ontologies," in *2nd International Conference on Practical Aspects of Knowledge Management (PAKM)*, Basel, Switzerland, 1998.

[7] E. F. Souza, R. A. Falbo, and N. Vijaykumar, "Using ontology patterns for building a reference sofware testing ontology," in *The 8th International Workshop on Vocabularies, Ontologies and Rules for the Enterprise and Beyond (VORTE)*, Vancouver, BC, 2013, pp. 21 – 30.

[8] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, "Systematic mapping studies in software engineering," in *Proceedings of the 12th international conference on Evaluation and Assessment in Software Engineering*, 2008, pp. 68–77.

[9] E. F. Souza, R. A. Falbo, and N. Vijaykumar, "Knowledge management applied to software testing: A systematic mapping," in *The 25th International Conference on Software Engineering and Knowledge Engineering (SEKE)*, Boston, USA, 2013, pp. 562–567.

[10] ——, "Ontologies in software testing: A systematic literature review," in *In VI Workshop on Ontology Brazil(ONTOBRAS)*, Belo Horizonte/MG, 2013, pp. 71–82.

[11] I. Burnstein, *Practical Software Testing: A Process-oriented Approach*. New York: Springer, 2003.

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

[13] A. Abran, P. Bourque, R. Dupuis, and W. James, *Guide to the Software Engineering Body of Knowledge (SWEBOK)*. USA: IEEE Press, 2004.

[14] R. Black and M. J. L., *Advanced Software Testing: guide to the ISTQB advanced certification as an advanced technical test analyst*. USA: Oreilly and Assoc, 2008.

[15] D. E. O'Leary and R. Studer, "Knowledge management: An interdisciplinary approach," *Intelligent Systems*, vol. 16, pp. 24–25, 2001.

[16] H. M. Kim, "Developing ontologies to enable knowledge management: Integrating business process and data driven approaches," in *Workshop on Bringing Knowledge to Business Processes*, 2000, pp. 20–22.

[17] S. Staab, R. Studer, H. P. Schurr, and Y. Sure, "Knowledge processes and ontologies," *IEEE Intelligent Systems*, vol. 16, pp. 26–34, 2001.

[18] F. Bjørnson and T. Dingsøyr, "Knowledge management in software engineering: A systematic review of studied concepts, findings and research methods used," *Information and Software Technology*, vol. 50, pp. 1055–1068, 2008.

[19] I. Rus and M. Lindvall, "Knowledge management in software engineering," *IEEE Trans. on Software Engineering*, vol. 19, pp. 26–38, 2002.

[20] J. Itkonen, M. V. Mantyla, and C. Lassenius., "The role of the tester's knowledge in exploratory software testing," *IEEE Trans. on Software Engineering*, vol. 39, pp. 707–724, 2013.

[21] S. Jalali and C. Wohlin, "Systematic literature studies: database searches vs. backward snowballing," in *International symposium on Empirical software engineering and measurement*, 2012, pp. 29–38.

[22] Y. Liu, J. Wu, X. Liu, and G. Gu, "Investigation of knowledge management methods in software testing process," in *International Conference on Information Technology and Computer Science*, Kiev, Ukraine, 2009, pp. 90–94.

[23] X. Li and W. Zhang, "Ontology-based testing platform for reusing," in *In Sixth International Conference on Internet Computing for Science and Engineering*, Henan, China, 2012, pp. 86–89.

[24] L. v. E. Abecker, "Ontologies for knowledge management," *International Handbooks on Information Systems*, pp. 435–454, 2004.

[25] D'Aquin and A. M., Gangemi, "Is there beauty in ontologies?" *Applied Ontology*, vol. 6, no. 3, pp. 165–175, 2011.

[26] R. Pretorius and D. Budgen, "A mapping study on empirical evidence related to the models and forms used in the uml," in *Proceedings of 2nd ACM-IEEE International Symposium on Empirical Software Engineering and Measuremen (ESEM)*, 2008, pp. 342–344.

[27] D. Budgen, A. J. Burn, O. P. Brereton, B. A. Kitchenham, and R. Pretorius, "Empirical evidence about the uml: a systematic literature review," *Software - Practice and Experience*, vol. 41, no. 4, pp. 363–392, 2011.

[28] C. Zhang and D. Budgen, "What do we know about the effectiveness of software design patterns?" *IEEE Transactions on Software Engineering*, vol. 38, no. 5, pp. 1213–1231, 2012.

[29] W. Janjic and C. Atkinson, "Utilizing software reuse experience for automated test recommendation," in *Proceedings of the 8th International Workshop on Automation of Software Test (AST)*, San Francisco, USA, 2013, pp. 100–106.