Experience Report on Developing an Ontology-based Approach for Knowledge Management in Software Testing

Érica Ferreira de Souza
Federal University of Technology – Paraná
Cornélio Procópio, Paraná, Brazil
ericasouza@utfpr.edu.br

Ricardo de Almeida Falbo
Federal University of Espírito Santo
Vitória, Espírito Santo, Brazil
falbo@inf.ufes.br

Marcos S. Specimille, Alexandre G. N. Coelho
Federal University of Espírito Santo
Vitória, Espírito Santo, Brazil
marcosspecimille@gmail.com
alexandregncoelho@gmail.com

Nandamudi L. Vijaykumar
National Institute for Space Research
Federal University of São Paulo
São José dos Campos, SP, Brazil
vijay.nl@inpe.br,vijaykumar@unifesp.br

Katia Romero Felizardo
Federal University of Technology – Paraná
Cornélio Procópio, Paraná, Brazil
katiascannavino@utfpr.edu.br

Giovani Volnei Meinerz
Federal University of Technology – Paraná
Cornélio Procópio, Paraná, Brazil
giovanimeinerz@utfpr.edu.br

ABSTRACT

Software testing is a knowledge intensive process. Thus, Knowledge Management (KM) emerges as a means to manage testing knowledge, and, consequently, to improve software quality. However, there are only a few KM solutions supporting software testing. This paper reports experiences from the development of an approach, Ontology-based Testing Knowledge Management (OntoT-KM), to assist in launching KM initiatives in the software testing domain with the support of Knowledge Management Systems (KMSs). OntoT-KM provides a process guiding how to start applying KM in software testing. OntoT-KM is based on the findings of a systematic mapping on KM in software testing and the results of a survey with testing practitioners. Moreover, OntoT-KM considers the conceptualization established by a Reference Ontology on Software Testing (ROoST). As a proof of concept, OntoT-KM was applied to develop a KMS called Testing KM Portal (TKMP), which was evaluated in terms of usefulness, usability and functional correctness. Results show that the developed KMS from OntoT-KM is a potential system for managing knowledge in software testing, so, the approach is able to guide KM initiatives in software testing.

CCS CONCEPTS

• **Software and its engineering → Software verification and validation**.

KEYWORDS

Knowledge Management, Knowledge Management System, Software Testing, Testing Ontology

SBQS'20, December 1–4, 2020, São Luís, Brazil
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-8923-5/20/12…$15.00
https://doi.org/10.1145/3439961.3439993

ACM Reference Format:
Érica Ferreira de Souza, Ricardo de Almeida Falbo, Marcos S. Specimille, Alexandre G. N. Coelho, Nandamudi L. Vijaykumar, Katia Romero Felizardo, and Giovani Volnei Meinerz. 2020. Experience Report on Developing an Ontology-based Approach for Knowledge Management in Software Testing. In *19th Brazilian Symposium on Software Quality (SBQS'20), December 1–4, 2020, São Luís, Brazil.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3439961.3439993

1 INTRODUCTION

Software testing is a knowledge intensive process. Knowledge Management (KM) emerges as a means to manage testing knowledge, and, consequently, to improve software quality. KM can be defined as a set of organizational activities that must be performed in a systematic way with the purpose of acquiring, organizing and sharing the different knowledge types in the organization [15]. The adoption of principles of KM in software testing can help testers to promote reuse of knowledge, to support testing processes and even to guide management decisions in organizations [25].

To enable testing knowledge reuse, software organizations should be able to capture this knowledge and make it available for their teams. However, there are only a few KM solutions in the context of software testing [25]. The major problem in organizations regarding software testing knowledge are low reuse rate of knowledge and barriers in knowledge transfer. This occurs because most of the testing knowledge in organizations is not explicit and it becomes difficult to articulate it [25]. On the other hand, implementing KM solutions in general is not an easy task. According to [28], a large number of organizations are taking great interest in the idea of KM, but, these organizations are not familiar with how and where to start since they lack the proper guidance to implement KM. So, with an orientation on how to implement new KM solutions in the organization, or even with a ready solution that can be customized, becomes interesting for organizations since it is an opportunity of continued cost reduction, quality improvement and reduction in software delivery [18].

With respect to technologies for KM, ontologies have been widely recognized as a key technology [8]. Ontologies can be used for establishing a common conceptualization to be used in the Knowledge

Management System (KMS) to facilitate communication, search, storage, and representation of knowledge [15]. However, only few initiatives have used an ontology-based approach for KM in software testing domain [25].

This paper reports our experiences on developing an approach to assist in launching KM initiatives in the software testing domain with the support of KMSs. In this paper, we present OntoT-KM, an Ontology-based Testing Knowledge Management approach. OntoT-KM provides a process to apply KM in software testing. OntoT-KM considers the conceptualization established by a software testing ontology. A striking feature of OntoT-KM is to describe how a testing ontology can be used for guiding KM initiatives in software testing. The software testing ontology used in OntoT-KM is a Reference Ontology on Software Testing, called ROoST [23].

Lessons learned and experiences acquired in conducting this study are presented on two main fronts. Firstly, OntoT-KM approach is presented with the objective of helping software organizations to implement an initial KM solution in software testing. Subsequently, a prototype of a KMS was developed, called Testing KM Portal (TKMP), both as a proof of concept from OntoT-KM approach, as well as understanding the needs of software development professionals in having a KMS in software testing ready and available for customization.

The main contributions of this research are the guidelines provided by OntoT-KM for guiding KM initiatives in software testing. These guidelines are supported not only by ROoST, but also from the findings of the mapping study [25] and the results of a survey with 86 testing practitioners. OntoT-KM was applied to develop TKMP, which was evaluated by test leaders of real projects in which it was applied. TKMP also was evaluated by 43 practitioners in terms of usefulness, usability and functional correctness. Such evaluation was designed applying the *Goal, Question, Metric* (GQM) paradigm [3] and *Technology Acceptance Model* (TAM) [4].

The remainder of this study is structured as follows. Section 2 presents the main research concepts. Section 3 presents OntoT-KM. Section 4 presents the application of OntoT-KM and the evaluation results. Section 5 discusses related works. Finally, in Section 6, we present our final considerations.

2 BACKGROUND

Software Testing consists of the dynamic Verification & Validation (V&V) activities of the behavior of a program on a finite set of test cases, against the expected behavior [1]. Testing activities are supported by a well defined and controlled testing process [1].

One of the main characteristics of software testing process is that it has a large intellectual capital component and can thus benefit from experiences gained from past projects [25]. During software testing, a large number of information is processed and generated. So, it can be considered a knowledge intensive process, making necessary for automated support acquiring, processing, analyzing and disseminating testing knowledge for reuse. In this context, Knowledge Management (KM) can be used [25].

KM can be viewed as the development and leveraging of organizational knowledge to increase organization's competitive advantage [29]. In general, KM formally manages the increase of knowledge in organizations in order to facilitate its access and reuse, typically by using Information Systems (ISs) and KMSs [8]. In particular, KMSs aim at supporting organizations in knowledge management, in an automated way.

One issue in KMSs is how to represent knowledge. One alternative is ontologies [15] as they are considered a key technology for KM [8], by defining the shared vocabulary to be used in the KMS facilitating knowledge communication, integration, search, storage and representation. In ontology-based KMSs, ontologies are typically used to structure the content of knowledge items, to support knowledge search, retrieval and personalization, serving as a basis for knowledge gathering, integration, and organization, and to support knowledge visualization, among others.

KM has shown important benefits for software organizations. In [25], we performed a Systematic Mapping (SM) looking for studies presenting KM initiatives in software testing. An SM is a secondary study for an overview of a research area through classification of the available evidence [10]. The main conclusions from this SM were: (i) There are few publications (only 15 studies were retrieved) addressing KM initiatives in software testing; (ii) The major problems that have motivated applying KM in software testing are low knowledge reuse rate and barriers in knowledge transfer; (iii) As a consequence, knowledge reuse and organizational learning are the main purposes for managing software testing knowledge; (iv) There is a great concern with both explicit and tacit knowledge; (v) Reuse of test cases is the perspective that has received more attention; (vi) KMSs are used in almost all initiatives (11 of the 15 studies); and (vii) Different technologies have been used to implement those KMSs, such as conventional technologies (databases, intranets, and Internet), Yellow Pages (or Knowledge Maps), Recommendation Systems, data warehouse and ontologies.

In particular, one finding drew our attention: only two studies actually used ontologies in a KM initiative applied to software testing ([11, 12]). This seems to be a contradiction, since, as pointed out by Staab et al. [27], ontologies are the glue that binds KM activities together, allowing a content-oriented view of KM. One possible explanation for this low number of studies is the fact that developing an ontology is a hard task, especially in complex domains, as is the case of software testing.

Based on the findings of the SM, we decided to perform a Systematic Literature Review (SLR) looking for ontologies on the software testing domain in the literature [24]. An SLR also is a secondary study that uses a well-defined process to identify available evidence [10]. From this SLR, 12 ontologies addressing this domain were identified. As the main findings, it is possible to highlight [24]: (i) Most ontologies have limited coverage; (ii) The studies do not discuss how the ontologies were evaluated; (iii) None of the analyzed testing ontologies is truly a reference ontology, i.e., a domain ontology that is constructed with the main goal of making the best possible description of the domain as realistic as possible; and, finally, (iv) 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, none of the analyzed ontologies is grounded in foundational ontologies. This motivated us to build ROoST, a Reference Ontology on Software Testing [23]. ROoST was developed for establishing a common conceptualization about the software testing domain.

Given the applicability of KM to improve software testing processes, we developed OntoT-KM for assisting companies that want to create their own solutions for KM initiatives in dynamic software testing, supported by a KMS. OntoT-KM is presented in the next section.

3 ONTOT-KM

OntoT-KM is supported by ROoST, in particular, to structure the KMS knowledge repository. OntoT-KM also are based on the findings of the SM presented in [25], and the results of a survey performed with testing practitioners presented in [26].

3.1 ROoST

ROoST is presented very briefly here, since it is not the scope of this paper to present the entire ontology. Details of the ontology can be found in [23].

Since the testing domain is complex, ROoST was developed in a modular way, comprising four modules (sub-ontologies): (i) **Software Testing Process and Activities** representing the testing process and the main activities that comprise it, namely Test Planning, Test Case Design, Test Coding, Test Execution, and Test Result Analysis; (ii) **Testing Artifacts** focusing on the artifacts used and produced by the testing activities; (iii) **Techniques for Test Case Design** looking at Testing Techniques, such as Black-box, White-box, Defect-based, and Model-based Testing Techniques; and (iv) **Software Testing Environment** addressing the main components of testing environment, including Test Hardware Resources, Test Software Resources and Human Resources.

In order to develop ROoST, the Systematic Approach for Building Ontologies (SABiO) [5] was adopted. SABiO method incorporates best practices commonly adopted in Software Engineering and Ontology Engineering, and addresses the design and coding of operational ontologies. Furthermore, ROoST has been developed by reusing and extending ontology patterns from the Software Process Ontology Pattern Language (SP-OPL) [6] and the Enterprise-Ontology Pattern Language (E-OPL) [7]. An Ontology Pattern Language (OPL) is a network of interconnected domain-related ontology patterns that provides holistic support for solving ontology development problems for a specific domain [6]. More recently, ROoST has been integrated into the Software Engineering Ontology Network (SEON) [19]. The full model of ROoST is available at *https://nemo.inf.ufes.br/projects/seon/*.

Given the size of ROoST, Figure 1 presents only its *Testing Process and Activities* sub-ontology. Concepts reused from Software Process Ontology are shown in gray. Specific Concepts are shown in white.

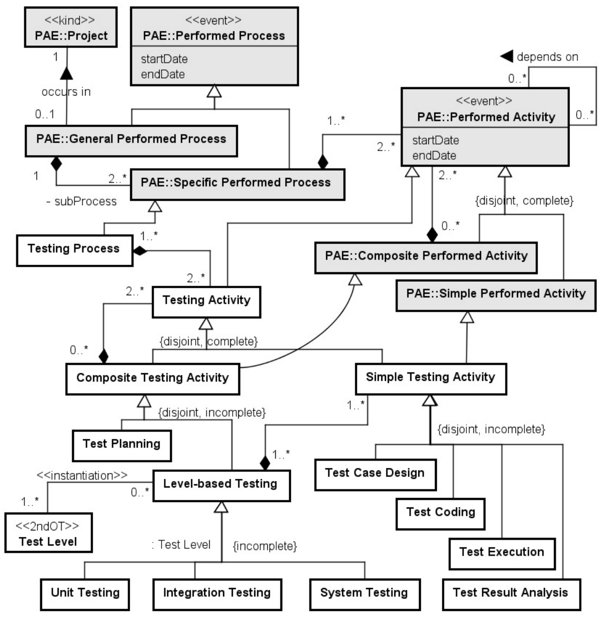
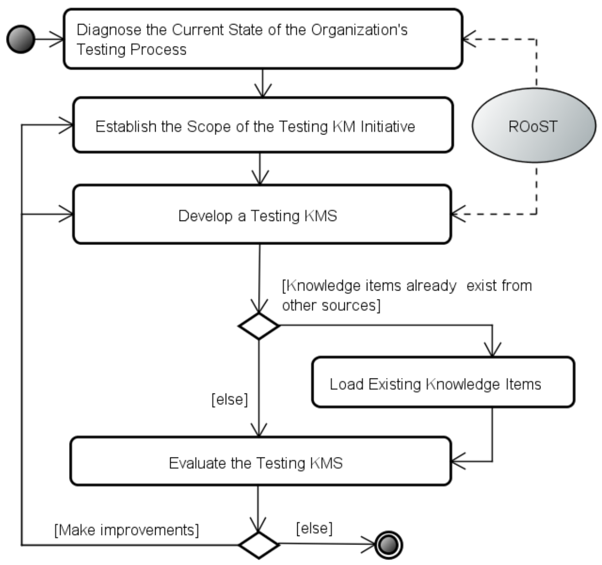


Figure 1: ROoST's Testing Process and Activities sub-ontology

3.2 OntoT-KM Process

OntoT-KM process comprises the following steps: (i) Diagnose the Organization's Testing Process; (ii) Establish the Testing KM Scope; (iii) Develop a Testing KMS; (iv) Load Existing Knowledge Items; and (v) Evaluate the Testing KMS. Figure 2 presents OntoT-KM process as a UML activity diagram. As this figure shows, ROoST is used to support steps (i) and (iii). Following, each process step is presented, describing the main guidelines that apply.



Diagnose the Current State of the Organization's Testing Process
Establish the Scope of the Testing KM Initiative
ROoST
Develop a Testing KMS
[Knowledge items already exist from other sources]
Load Existing Knowledge Items
[else]
Evaluate the Testing KMS
[Make improvements]
[else]

Figure 2: OntoT-KM Process

Step 1: Diagnose the Current State of the Organization's Testing Process

The very first step of the OntoT-KM process is to make a diagnosis of the current state of the organization's testing process. It refers to investigating the existing knowledge within the testing process, in order to identify knowledge assets and understand how and where testing knowledge is developed and used in the organization. This is an important step for organizations with low maturity. Once identifying the knowledge items, organizations can then proceed to manage them.

This step may be accomplished by surveys with questionnaires and/or interviews. This activity should consider the entire current state of software testing in the organization with respect to, at least, the following aspects: the adopted testing process, the activities of the candidate testing processes to be targeted by KM initiative, the artifacts produced during this process, the testing techniques applied, the test levels contemplated by the process, and the test environments adopted by the organization's software projects. Aspects related to KM should also be investigated, such as the current KM practices applied in the testing process, the organization's purpose of applying KM to software testing, problems related to testing knowledge in the organization, among others. ROoST can be used in this step as the common vocabulary for supporting the analysis of the current status, as well as to formulate the questions to be used in the questionnaires and/or interviews with the organization's testing practitioners. The results of these questionnaires/interviews should be used as guidelines for the next step (establish scope).

For this step, we suggest to ask, at least, the following questions to the testing practitioners participating in the diagnosis:

- *What are the testing activities that comprise the organization's testing process?* Evaluate the answers considering the consensual activities considered in ROoST (Test Planning, Test Case Design, Test Coding, Test Execution, and Test Result Analysis), and consider the possibility of improving the organization's testing process by aligning it to the testing process captured by ROoST.
- *In which activities of the Testing Process is KM more useful?* The activities pointed out in the previous step should be the ones considered here as possible answers.
- *What are the testing levels in which the organization performs tests?* Testing activities can be performed in different levels. Taking ROoST as basis, consider at least the following levels: Unit Testing, Integration Testing, and System Testing. However, if the organization tests software in other levels, these should be taken into account.
- *In which testing level is KM more useful?* The testing levels pointed out in the previous step should be the ones considered here as possible answers.
- *Which resources do you consider more important to have knowledge available about them when defining the testing environment?* According to ROoST, consider as possible answers for this question the following types of resources: Hardware, Software, and Human.
- *Concerning tacit and explicit knowledge, what are the types of knowledge you consider to be more important to manage during the software testing process?* Testing practitioners tend to consider both useful, but we need to evaluate which one is more important and also which is easier to implement. In general, for organizations starting a KM initiative in software testing, explicit knowledge is easier to handle. In particular, test cases highlight as the most important artifacts to be managed as a knowledge item, as pointed in SM [25] and the survey [26].
- *What is the purpose of applying KM in Software Testing? What benefits can KM bring to software testing?* This question captures the feeling of testing practitioners regarding why and how an organization can benefit by applying KM to software testing.

Step 2: Establish the Scope of the Testing KM Initiative

For the KM scope, it is necessary to familiarize with the organization needs. The organization must define the testing process activities that are to be supported, and knowledge types to be managed.

A major challenge for organizations is to know which knowledge is really useful, and thus identify potential knowledge items among the several knowledge assets generated in the testing process. Results from step 1 should be used here. In addition, it is suggested that organizations starts with small KM initiatives.

As a general guideline, we recommend to consider the survey results that we performed [26]. From this survey, Test Case Design and Test Planning were considered the most important testing activities to be supported by KM practices, and capturing as knowledge items their main outcomes, namely: Test Case and Test Plan.

Regarding Test Planning, the survey led to selection of testing techniques and definition of testing environment as the most important tasks to be supported by a KM initiative. Concerning knowledge about the testing environment, managing knowledge about human resources and software resource is pointed out as the most promising approach. Regarding knowledge about human resources, this impression is corroborated by the SM [25], where yellow pages and knowledge maps appear in various initiatives.

Step 3: Develop a Testing KMS

This phase concerns developing a KMS to support the KM initiative, and comprises the main activities for developing systems, in general: requirements specification, design, implementation and testing.

Requirements must be elicited and specified. Functional requirements may be created from use case models, class diagrams, and state diagrams to model the behavior of knowledge items throughout their existence in the KMS. Non-functional requirements should also be addressed, such as security, usability, accessibility, etc.

ROoST is very useful in this step. ROoST can serve as the initial conceptual model for the KMS, and thus as the basis for structuring the testing knowledge repository. Specific information (attributes of the classes in the conceptual model) should be identified, taking the characteristics of the organization's testing artifacts into account, and, in particular, information available in the tools used for supporting the testing process.

Furthermore, interoperability issues should also be analyzed. Ideally, software tools that are part of the test environment should be integrated with the Testing KMS to act together, interacting and exchanging data to obtain the expected results. In this context, possible knowledge items identified in these tools can be automatically converted/imported to the testing KMS.

Another key point is to define the KM process activities that are to be supported by the Testing KMS. We recommend providing support to the following typical activities of a KM process: creating knowledge items, evaluating knowledge items before making them available, searching knowledge items, assessing usefulness of available knowledge items, and maintaining the knowledge repository.

During the design of Testing KMS, developers should consider the platform in which the system is to be built, and non-functional requirements should be addressed. Several technologies can be used, including those that are commonly considered in KM solutions as Content Management Systems, Document Management Systems, and Wikis, as well as those considered intelligent KM solutions, such as knowledge-based and expert systems, reasoners, and semantic wikis. Once designed, the KMS should be coded and tested.

Step 4: Load Existing Knowledge Items.

For initially populating the knowledge repository of the Testing KMS, the organization should look for existing knowledge items. For instance, if the system must manage test cases, existing test cases can be imported to the Testing KMS. The existing knowledge items should be reengineered to ensure conformance with the knowledge repository structure. Mechanisms for loading and reengineering these knowledge items can be built to automate the loading process.

Step 5: Evaluate the Testing KMS.

Evaluation should be done to determine if the Testing KMS meets the expectations. Improvements can be carried out, implying a return to the previous steps. A suggestion to evaluate the testing KMS is to analyze some quality characteristics, such as usefulness, usability and functional correctness. To do that, two models can be considered: GQM [3] and TAM [4].

GQM is a measurement model, organized in three levels. In the first level (conceptual level), the study goals should be defined. The second level (operational level) refers to a set of questions that should be defined with the purpose of characterizing the evaluation or the accomplishment of a specific goal. Finally, in the last level (quantitative level), a set of metrics should be associated with questions, in order to answer them in a measurable way. The result of applying the GQM approach is the specification of a measurement system targeting a particular set of issues and a set of rules for interpreting measurement data [3].

TAM determines the acceptance of a given technology by users, considering two-factor analysis: usefulness and usability. Usefulness refers to how much the user realizes that a certain technology is useful to her in terms of productivity increase. The perception of usability refers to the effort reduction that the user achieves when using a given technology [4].

In this work, we decided to evaluate also another quality characteristic: functional correctness, a subcharacteristic of functional suitability in ISO/IEC 25010 product quality model. According to [9], functional correctness is the "degree to which a product or system provides the correct results with the needed degree of precision".

4 APPLYING ONTOT-KM

As mentioned earlier (Section 1), our experience in developing the OntoT-KM approach has two fronts. First we introduce the approach, and then we create a prototype of a KMS based on OntoT-KM that allows us to evaluate the approach, as well as obtain the opinion of software professionals in having a KMS in software testing ready and available for customization. Following, the KMS development based on OntoT-KM and the all process of the evaluations are presented.

To evaluate OntoT-KM approach, we applied it to build a prototype of a KMS for managing software testing knowledge, called Testing Knowledge Management Portal (TKMP). The resulting system was populated with data from two real projects and different evaluations were conducted. The projects were: (i) Amazon Integration and Cooperation for Modernization of Hydrological Monitoring (ICAMMH) Project [22]; and (ii) On-Board Data Handling (OBDH) software inside the Inertial Systems for Aerospace Application (SIA) Project [22].

ICAMMH Project was a collaboration involving the Brazilian Aeronautics Institute of Technology and the Brazilian National Water Agency, supported by the Brazilian Financial Foundation for Projects - FINEP. The project developed a pilot system for modernization and integration of telemetry points collected from hydrological data, as a basis for managing water resources in the Amazon region. The second project is devoted to develop a software for the onboard computer of SIA Project, which is a computational system for OBDH (On Board Data Handling) to Attitude and Orbit Control (AOC) of satellites that can be adapted for future space applications at National Institute for Space Research (INPE).

4.1 Diagnose the Current State of the Organization's Testing Process

As ICAMMH Project has already been finalized and the testing activities of the SIA project were only in the very initial phase, it was not possible to run the diagnostic step. This step was replaced by the findings from the survey with 86 testing practitioners we performed [26]. Out of these 86 participants, some are also team members and leaders of the TKMP and SIA project.

The survey's purpose was 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. The survey presents questions that addressed aspects considered both in the conceptualization of ROoST and by the SM presented in [25].

From the survey results, the following conclusions are considered: (i) the participants identified test case design and test planning as being the activities in which KM would be most useful. Therefore, test cases and test plans are considered the most useful artifacts to be reused; (ii) explicit knowledge was considered more important than tacit knowledge. Explicit knowledge represents the objective and rational knowledge that can be documented, and thus it can be accessed by many [14]. On the other hand, the tacit knowledge is the subjective and experience-based knowledge and typically remains only in people's minds [14]; (iii) among the most targeted artifacts to reuse, test cases stood out with 90.7%; and (iv) the purposes for which experts are more interested in applying KM in software testing are related to improving the quality of results in software testing, and reducing cost, time and effort spent in software project.

4.2 Establish the Scope of the Testing KM Initiative

Considering the main findings of the survey, **test case design** was considered the software testing activity to be supported, and **test**

case the main artifact to be managed. All relevant information for designing test cases had to be considered in the scope of the TKMP development. Thus, concepts related to Test Case in ROoST were also considered in the scope of initiative, namely: *Test Case Input, Expected Result, Test Result, Test Code, Test Case Designer* and *Testing Technique*.

Besides the test cases as the main artifacts to be managed, other knowledge items related to making tacit knowledge explicit were also considered in the scope of TKMP, namely: Lessons Learned (LL) and Knowledge regarding discussion.

These two types of knowledge items were considered in the scope of TKMP, since survey participants pointed out individual experiences and communication between test team members as the types of tacit knowledge with more significant importance to generate explicit knowledge items. In addition, meetings with the project leaders from ICAMMH and SIA projects also helped to reach this scope.

Still with respect to tacit knowledge, we decided that TKMP should also include a Yellow Page System, since survey participants pointed out human resources as the most useful resource to be managed and test case designers are in the scope of this KM initiative.

4.3 Develop a Testing KMS

Considering the scope defined in the previous activity, TKMP was developed. The main requirements specification were developed, such as use case diagram and class diagram. Figure 3 shows a partial use case diagram describing the main functionalities of TKMP and actors. The use cases in gray are general, in the sense that they apply to managing software engineering knowledge items of different nature. Use cases in white represent testing-specific features. *Developer* is the main actor, representing all types of professionals involved in the software development process. *Knowledge Manager* represents a user with specific permissions, guaranteeing access to features inherent only to a Knowledge Manager.

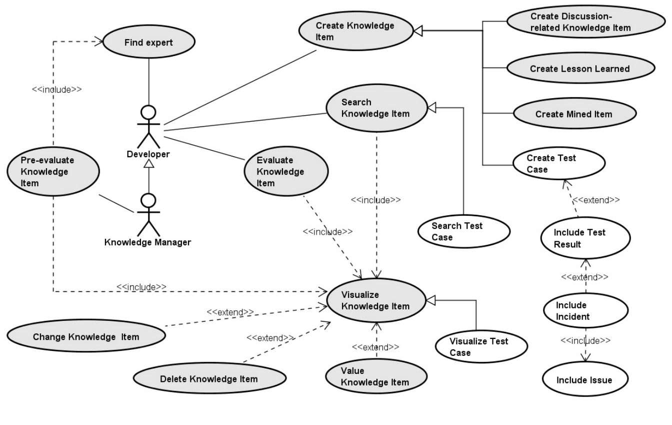


Figure 3: Functionalities of TKMP

ROoST conceptual model was used as the starting point for specifying TKMP, mainly to structure its knowledge repository regarding the software testing notions. Information from the software tools that compose the testing environment of ICAMMH Project was used as the basis for identifying attributes and enumerated types, in order to specify TKMP in details. These tools are TestLink[1], a web-based test management system, and MantisBT[2], a bug tracking system. TKMP project and others requirements specifications are currently available at *https://cutt.ly/KyBOLUn*.

4.4 Load Existing Knowledge Items

Once TKMP was developed, previous existing knowledge items in the two projects were loaded to the knowledge repository. Initially, TKMP's knowledge repository was populated with 1568 test cases extracted from ICAMMH Project. Next, other test cases from the SIA Project were also inserted in TKMP's knowledge repository using TKMP's functionalities.

In the context of the ICAMMH Project, test case related information was stored both in TestLink and MantisBT. Each one of these tools has its own data repository, implemented in different ways, demanding an analysis of the structure of each one to load the data. Moreover, each tool has its own terminology to represent the manipulated data, i.e., different terms are used to represent the same concept. Thus, to load existing test cases, a feature was developed to connect and get data from the repositories of MantisBT and TestLink, and then converts them into objects (instances) of the data schema of TKMP. ROoST was used for mapping concepts from the involved tools.

4.5 Evaluate the Testing KMS

Although TKMP is still considered a prototype, built as proof of concept for the OntoT-KM approach, we decided to conduct different evaluations for this KMS in order to get the feeling of software professionals in having a KMS available for customization.

TKMP went through a preliminary evaluation in two steps. Firstly, TKMP was evaluated by the leaders of the two aforementioned projects, ICAMMH and SIA. Secondly, TKMP was made available in the Web and software engineering practitioners were invited to use it and then to answer a questionnaire to give feedback in terms of usefulness, usability and functional correctness.

4.5.1 Evaluation with the project leaders. Once TKMP's knowledge repository was populated with data from the two real projects (ICAMMH and SIA), their leaders were requested to use and analyze the portal. Meetings with each of them included system demonstrations with data obtained from their projects. Both the leaders stressed the importance of such a system to better support the software testing processes.

With respect to ICAMMH Project, the leader observed that there was always a great loss of knowledge due to the turnover rate of the team members. In her words, "a KMS such as TKMP would be definitely beneficial for finding similar test cases to be reused in the design of new ones to other similar situations in different modules and also in future projects".

With respect to the SIA Project, the leader's evaluation was that TKMP would be very important for dealing with critical systems. However, he pointed out that a challenge would be to change team

[1]http://testlink.org/
[2]http://www.mantisbt.org/

members' culture, because many times the team is not ready or does not accept new concepts, tools and ideas.

4.5.2 Evaluation by software engineering practitioners. TKMP was also evaluated by 43 practitioners in Software Engineering and it was based in GQM, TAM and functional correctness. The evaluation based on the GQM paradigm involved four steps: Planning, Definition, Data Collection, and Interpretation.

(I) Planning and Definition. At GQM's conceptual level, measurement goals should be defined. We identified three goals for this evaluation, and from these goals, in the operational level, we defined seven questions, as Table 1 shows. Finally, in the quantitative level, we defined metrics associated to the questions, in order to answer them in a measurable way. For each question, as Table 2 shows, we defined five metrics, each one aiming at computing the number of participants that strongly disagree (MG.Q.1), disagree (MG.Q.2), neither agree nor disagree (MG.Q.3), agree (MG.Q.4) or strongly agree (MG.Q.5) with a statement corresponding to the question. Figure 4 summarizes the GQM approach we followed.

Table 1: Defined Goals and Questions

G1: Evaluate TKMP usefulness	**Q1.1.** What is the perceived usefulness of TKMP regarding creating software testing knowledge items? **Q1.2.** What is the perceived usefulness of TKMP regarding searching for software testing knowledge items? **Q1.3.** What is the perceived usefulness of TKMP regarding reusing software testing knowledge items? **Q1.4.** What is the perceived global usefulness of TKMP?
G2: Evaluate TKMP usability	**Q2.1.** To what extent do users recognize that it is easy to learn to use TKMP? (learnability) **Q2.2.** To what extent do users recognize that TKMP is appropriate for their needs? (appropriateness recognizability)
G3: Functional Correctness	**Q3.1** Do users notice inconsistencies when using TKMP?

Table 2: Metrics used in the GQM

MG.Q.1	Number of participants who strongly disagree
MG.Q.2	Number of participants who disagreed
MG.Q.3	Number of participants who neither agree nor disagree
MG.Q.4	Number of participants who agree
MG.Q.5	Number of participants who strongly agree

Table 3 presents the statements that we used to represent the questions in the questionnaire that participants answered.

Questions Q1.1–Q1.4 were used to characterize the portal usefulness, questions Q2.1–Q1.2 were used to collect data on the level of usability. Question Q3.1 was used to evaluate TKMP functional correctness. Table 4 shows how to interpret the results. The lines should be read as "IF <<expression>> THEN <<interpretation>>". For example, the interpretation of Question 3.1 (Q3.1) is "IF M1+M2 > M4+M5 THEN the users does not notice inconsistencies when using the TKMP", where M1, M2, M4, M5 are the responses given by the participants (metrics). It is important to notice that M1 and M2 (see Table 2) are answers that totally or partially disagree with the question. On the other hand, M4 and M5 are answers that totally or partially agree with the question.

Table 3: Statements used to refer to the questions

Q1.1	TKMP is useful to create software testing knowledge items.
Q1.2	TKMP is useful to search for software testing knowledge items.
Q1.3	TKMP is useful to reuse software testing knowledge items.
Q1.4	I would use or recommend the TKMP.
Q2.1	I learned to use the TKMP quickly.
Q2.2	I recognize TKMP as being suited to my tester needs.
Q3.1	I did not notice inconsistencies when using the TKMP.

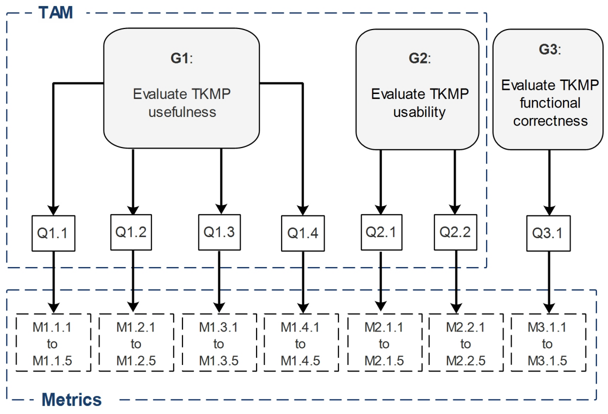


Figure 4: GQM approach to evaluate the TKMP

(II) Data Collection. The data used to evaluate the TKMP were based on the metrics presented above. To collect the data, we requested experts in software organizations to use TKMP to perform activities to create, validate and search for knowledge items. After using the tool, 43 participants answered a questionnaire containing the questions previously presented.

Considering the participants' profile, out of these 43, 8 hold Doctoral degrees, 13 hold Masters, 22 finished undergraduate programs. All of them are from the Software Engineering area and they have an average of six years of experience in the area. In relation to software testing knowledge, 42.9% of participants reported having basic knowledge, 37.2% reported having intermediate knowledge, and 23.3% considered to have advanced knowledge on software testing. A summary of the responses given by the participants is shown in Table 5. This table shows the number of the responses according to the goals, questions and metrics used.

(III) Interpretation. Figures 5, 6 and 7 present charts that show the answers per question used in our GQM model. These answers were interpreted according to Table 4:

Table 4: Results interpretation

01	For $Q1.i$, i=1 to 4: M1+M2 > M4+M5	TKMP is not useful for managing software testing knowledge items.
02	For $Q1.i$, i=1 to 4: M1+M2 < M4+M5	TKMP is useful for managing software testing knowledge items.
03	For $Q1.i$, i=1 to 4: M3 > M4+M5 or M1+M2 = M4+M5	We cannot say that TKMP is useful to manage software testing knowledge items
04	To $Q2.1$ and $Q2.2$: M1+M2 > M4+M5	TKMP cannot be easily used to manage software testing knowledge items.
05	To $Q2.1$ and $Q2.2$: M1+M2 < M4+M5	TKMP can be easily used to manage software testing knowledge items.
06	To $Q2.1$ and $Q2.2$: M3 > M4+M5 or M1+M2 = M4+M5	We cannot say that TKMP can be easily used to manage software testing knowledge items.
07	To $Q3.1$: M1+M2 > M4+M5	TKMP can be considered functionally correct.
08	To $Q3.1$: M1+M2 < M4+M5	TKMP cannot be considered functionally correct.
09	To $Q3.1$: M3 > M4+M5 or M1+M2 = M4+M5	We cannot say if TKMP is functionally correct or not.

Table 5: Results Summary

	Questions	Metrics					
Goal		M1	M2	M3	M4	M5	Total
G1	Q1.1	0	0	0	19	24	43
	Q1.2	2	2	8	17	14	43
	Q1.3	0	1	7	16	19	43
	Q1.4	0	4	9	8	22	43
G2	Q2.1	0	2	11	17	13	43
	Q2.2	1	0	11	17	14	43
G3	Q3.1	3	5	14	14	7	43

Goal 1: Evaluate TKMP usefulness - Figure 5 presents the chart generated from the answers related to TKMP usefulness. Applying the interpretation expressions shown in Table 4, in relation to this goal, the results show that the participants considered TKMP a useful tool for managing software test knowledge items.

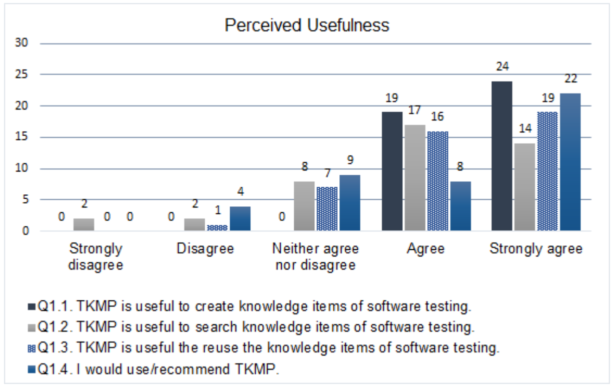


Figure 5: Questions and answers related to usefulness of TKMP

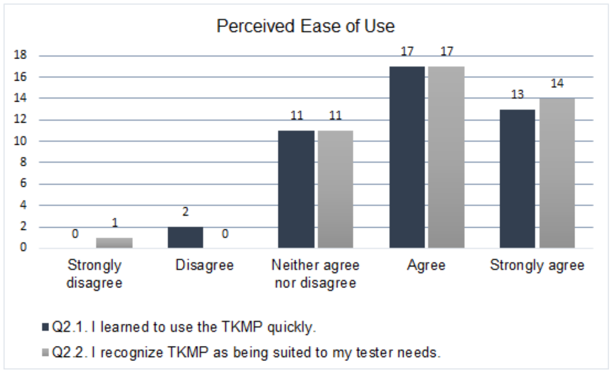


Figure 6: Questions and answers related to usability of TKMP

Goal 2: Evaluate the usability of TKMP - Figure 6 presents the chart generated from the answers related to usability. The results showed the participants considered that TKMP can be easily used to manage software testing knowledge items.

Goal 3: Evaluate the functional correctness of TKMP - Figure 7 presents the chart related to functional correctness. The results show that TKMP can be considered functionally correct. However, even the metrics pointing out that most of the participants did not find inconsistencies to the point that they were not able to use TKMP, by Figure 7, it is possible to notice that the participants found inconsistencies in TKMP. We consider this a normal result, since TKMP is still a prototype.

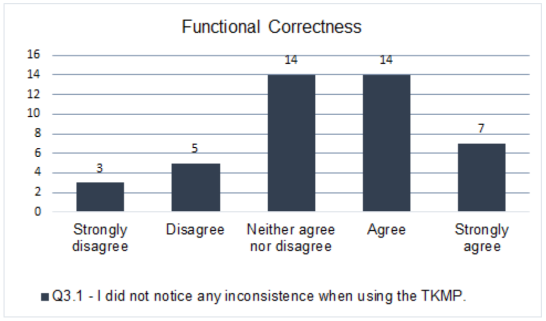


Figure 7: Question and answers related to functional correctness of TKMP.

4.6 Other Partial Applications of OntoT-KM

We also started to apply OntoT-KM in software organizations. Three companies evaluated OntoT-KM and TKMP. First, we conducted the diagnostic and scope definition activities in these three companies by applying a questionnaire based on the survey presented in [26].

Respondents to the questionnaire were software testers responsible for the software testing activities within the companies. For privacy reasons, we do not mention the company's names. However, some characteristics are: located in Brazil; medium sized software organizations; the main products they develop are systems for fiscal area, such as electronic fiscal receipt, metrology and also customized systems to meet the needs of customers from diverse segments. The main diagnosis results by the three companies are: (i) "Test Case Design" activity was the most useful; (ii) "Test Environment Structuring" was the testing planning activity in which KM is most useful; (iii) "Human resource" and "Software Resource" are considered the resources from which it is quite important to have knowledge available at the time of setting the test environment; (iv) the explicit knowledge was considered more important than tacit knowledge; (v) "Test Plan" and "Test Case" were considered the artifacts most reusable ones; (vi) There is no formal instrument for KM within the three companies; and (vii) "Increasing the testing process efficiency" and "Best test case selection" are the main expected benefits of applying KM in software testing.

The results were very close to the results obtained in the general survey applied to the 86 participants in [26]. From the diagnosis results in the companies, it was possible to establish the scope for software testing initiatives. Considering the main findings, *test plan* definition and *test case design* were considered the software testing activities to be first supported, and test cases the main knowledge item to be managed.

Until now, we have not conducted the remaining activities of OntoT-KM approach (develop a KMS Testing, Load Existing Knowledge Items and Evaluate the Testing KMS), although companies have shown interest in developing their own KMS solution. Our intention was that companies would also use TKMP. We proposed to the three companies the use of TKMP already developed from the research projects scope, since the diagnosis results were similar. Companies were at ease in uploading the organization's data in TKMP or to register new data, if they wish. The purpose was to analyze if an already existing KM tool, such as TKMP, could be customized by the organization to meet its current needs. Some suggested customizations were: (i) to implement a traceability matrix among test cases and lessons learned in order to assist the test coverage; (ii) to develop a repository of artifacts (historical basis); and (iii) turning TKMP into a plug-in to integrate with project management tools (e.g., JIRA, Redmine).

The two fronts analyzed in this study were well accepted by the three organizations that participated in the survey. The participants mentioned that it is interesting to have their own solution for a KMS using OntoT-KM. However, they mentioned this would be possible if the company had a team to develop the system. On the other hand, it is also attractive to have a more general and open source KMS available to be customized by the company. We believe that the experience we had from evaluations performed both OntoT-KM, as well as TKMP, give us motivations and directions for future works, for example, we intend to consider some of the customization suggestions and enhancements to be implemented in TKMP, since there is an intention to create a robust version of the portal to be made available to the Software Testing community.

4.7 Study Limitations

There were limitations in the study. The first limitation refers to a low representativeness of companies participating in the study (3 companies). Validating an approach, as OntoT-KM in a real environment, needs the authorization and trust of the organization to use its data and information. However, we noticed an enormous barrier to this. Several other companies were invited to apply OntoT-KM. While they recognize the benefits from KM in software testing, many refused to participate. The invited organizations mentioned that the idea of implementing KM in the organization, even with an existing tool, could generate an increased workload. This position is in line with the results detected through the SM conducted in [25]. Shortage of time is also a potential risk to incorporate KM principles in software testing, because knowledge sharing can imply in increasing the employee workload and costs. Our intention is to continue inviting software companies to participate in the research and look for strategies that allow the company to feel safe in relation to use or build a KMS, for example, allow the company to install the system on-premise and on their own database server.

A second limitation of this research concerns the sample size of the software engineering practitioners that answered the questionnaire. 43 practitioners answered the questionnaire. The results cannot be generalized. Therefore, we intend to replicate this survey as many software practitioners as possible in real projects in the industry.

Another limitation is related to step 1 of OntoT-KM. This step was not used on both projects (ICAMMH and SIA). To create a generic KMS, we intend to apply the diagnosis to as many software development companies as possible.

5 RELATED WORK

Dehghani and Ramsin [17] provide review of several methods for KMS development. In general, these methods provide activities, principles and techniques with the aim of applying KM in the organizations [2, 13, 16, 21]. Some of these KMS methodologies are briefly presented below.

In Rubenstein-Montano et al. [16], for example, a methodology in order to develop a KMS is presented. The methodology phases are: (i) A strategic planning; (ii) Models of logical and physical aspects; (iii) Development of the KMS prototype; (iv) V&V the KMS; and (v) Deploy and maintain the KMS. Similarly to the methodology of Rubenstein-Montano et al., OntoT-KM also proposes a planning stage, called diagnosis, as well as generation of models to support the construction of a KMS and its validation. However, the activities in OntoT-KM are supported by a software testing ontology.

In Sarnikar and Deokar [21], a methodology is presented to direct the development process based on the workflows within the organization. The methodology consists of 7 steps: (i) Business process model development; (ii) Knowledge intensity identification; (iii) Requirements identification; (iv) Knowledge sources identification; (v) Knowledge reuse assessment; (vi) Task-user knowledge profile development; and (vii) Designs the system components. Different from the OntoT-KM process, the Sarnikar and Deokar methodology presents the design and construction of KMS only in its last phase.

In Amine and Ahmed-Nacer [2], an ontology-based agile methodology is presented to develop a KMS. As with Amine and Ahmed-Nacer, we also use the resources of an ontology. However, the ontology that we use is not created based on the organization, but rather on an already validated domain ontology and that aims at establishing a common conceptualization about software testing.

Finally, the methodology presented by Moteleb et al. [13] aims at using practical experiences for developing KMSs in small organizations. OntoT-KM also analyzes the solution for the organization with a diagnostic. However, as mentioned, OntoT-KM is supported by software testing ontology.

6 CONCLUSIONS

This work presents our experiences on developing an approach to assist in launching KM initiatives in software testing. OntoT-KM provides guidelines to apply KM with the development of KMSs and based on a software testing ontology. Although there are approaches for developing KMSs [17], to the best of our knowledge, there is no approach devoted to developing a KMS for supporting KM in software testing. In this respect, OntoT-KM is an original contribution. Results show that the developed KMS from OntoT-KM is a potential system for managing knowledge in software testing, so, the approach, is able to guide KM initiatives in software testing.

It is important to note that this study is part of a research project that involves undergraduate, masters and doctoral students. Until now, our intention was to evaluate the OntoT-KM approach, as well as the KMS generated. Now we intend to apply the diagnosis to as many software development companies as possible to reach a common scope in order to be developed in a general KMS. This KMS will be part of an environment already maintained by this research project, called Software Engineering KNOWledge management diagnosis (SEKNOW) [20]. Currently, SEKNOW was developed only to analyze KM in software development organizations (diagnostics step), however given the evolution of research, SEKNOW has been undergoing adaptations to meet more activities related to KM and software organizations. We also intend as future work to extend TKMP considering other conceptualization established by ROoST. We also intend to conduct more experimental studies to confirm the evaluations results discussed in this paper. Finally, we have been investigating the synthesis on KM and Agile Software Development, and it will certainly be considered in the next stages of this project.

ACKNOWLEDGMENTS

The first author would like to thank professor Ricardo de Almeida Falbo (in memoriam) for successfully leading this work and sharing your valuable advice. Authors would like to thank: Brazilian Aeronautics Institute of Technology (ITA) and Brazilian Agency of Research and Projects Financing (FINEP) - Project 5206/06-ICAMMH; and SIA Project for providing the data. Brazilian funding agency CNPq Project 432247/2018-1. All participants who used the TKMP and answered the evaluation questionnaire.

REFERENCES

[1] A. Abran, P. Bourque, J.R. Dupuis, and W. Moore. 2004. *Guide to the software engineering body of knowledge - SWEBOK.* Technical Report. A project of the IEEE Computer Society Professional Practices Committee.
[2] M. Amine and M. Ahmed-Nacer. 2011. An agile methodology for implementing knowledge management systems: a case study in component-based software engineering. *Software Engineering Applications* 5 (2011), 159–170.
[3] V. R. Basili, C. Caldiera, and H.D. Rombach. 1994. *Guide to the software engineering body of knowledge - SWEBOK.* Technical Report. Goal Question Metric Paradigm, New York: John Wiley & Sons.
[4] F. D. Davis. 1993. User acceptance of information technology: system characteristics, user perceptions and behavioral impacts. *International Journal of Man-Machine Studies* 38 (1993), 475–487.
[5] R. A. Falbo. 2014. SABiO: Systematic Approach for Building Ontologies. In *8th Intern. Conference on Formal Ontology in Information Systems.*
[6] R. A. Falbo, M.P. Barcellos, J.C. Nardi, and G. Guizzardi. 2013. Organizing Ontology Design Patterns as Ontology Pattern Languages. In *Extended Semantic Web Conference.* Montpellier.
[7] R. A. Falbo, F. B. Ruy, G. Guizzardi, M. P. Barcellos, and J. P. A. Almeida. 2014. Towards an Enterprise Ontology Pattern Language. In *Symposium On Applied Computing*. Gyeongju.
[8] R. J. G. Herrera and M. J. Martin-B. 2015. A novel process-based KMS success framework empowered by ontology learning technology. *Engineering Applications of Artificial Intelligence* 45 (2015), 295–312.
[9] ISO/IEC JTC. 2011. ISO/IEC 25010 - Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models.
[10] B.A. Kitchenham and S. Charters. 2007. *Guidelines for performing Systematic Literature Reviews in Software Engineering.* Technical Report EBSE 2007-001. Keele University and Durham University, UK.
[11] X. Li and W. Zhang. 2012. Ontology-based Testing Platform for Reusing. In *Intern. Conference on Internet Platform for Reusing.* Henan, China, 86–89.
[12] Y. Liu, J. Wu, X. Liu, and G. Gu. 2009. Investigation of Knowledge Management Methods in Software Testing Process. In *Inter. Conference on Information Technology and Computer Science.* Kiev, 90–94.
[13] A.A. Moteleb, M. Woodman, and P. Critten. 2009. Towards a practical guide for developing knowledge management systems in small organizations. In *European Conference on Knowledge Management.* 559–570.
[14] I. Nonaka and H. Takeuchi. 1997. *The knowledge-creating company* (1 ed.). Oxford University Press, Oxford, USA.
[15] D.E. O'Leary and R. Studer. 2001. Knowledge Management: an Interdisciplinary Approach. *IEEE Intelligent Systems* 16, 1 (2001).
[16] B. R-Montano, J. Liebowitz, J. Buchwalter, D. McCaw, B. Newman, and K. Rebeck. 2001. A systems thinking framework for knowledge management. *Decision Support Systems* 31 (2001), 5–16.
[17] R. D. R. Ramsin. 2015. Methodologies for developing knowledge management systems: an evaluation framework. *Journal of Knowledge Management* 19 (2015), 682–710.
[18] M. Rokunuzzaman and K. P. Choudhury. 2011. Economics of Software Reuse and Market Positioning for Customized Software Solutions. *Journal of Software* 6 (2011), 31–1029.
[19] F. B. Ruy, R.A. Falbo, M.P. Barcellos, S. D. Costa, and G. Guizzardi. 2016. SEON: A Software Engineering Ontology Network.. In *20th Inter. Conference on Knowledge Engineering and Knowledge Management (EKAW).* 527–542.
[20] V. Santos, J. G. Salgado, E. F. Souza, K. R. Felizardp, and N. L. Vijaykumar. 2019. A Tool for Automation of Knowledge Management Diagnostics in Software Development Companies. In *Brazilian Conference on Software: Theory and Practice (CBSoft) - Tools Session.*
[21] S. Sarnikar and A. Deokar. 2010. Knowledge management systems for knowledge-intensive processes: design approach and an illustrative example. In *International Conference on System Sciences.* 1–10.
[22] E. F. Souza. 2014. *Knowledge management applied to software testing: an ontology based framework.* Ph.D. Dissertation. National Institute for Space Research (INPE), Brazil.
[23] E. F. Souza, R. A. Falbo, and N.L. Vijaykumar. 2017. ROoST:Reference Ontology on Software Testing. *Applied Ontology* 12 (2017), 59–90.
[24] E. F. Souza, R. A. Falbo, and N. L. Vijaykumar. 2013. Ontology in Software Testing: a Systematic Literature Review. In *Research Seminar Ontology of Brazil (ONTOBRAS).* Belo Horizonte, 71–82.
[25] E. F. Souza, R. A. Falbo, and N. L. Vijaykumar. 2015. Knowledge management initiatives in software testing: A mapping study. *Information and Software Technology* 57 (2015), 378–391.
[26] E. F. Souza, R. A. Falbo, and N. L. Vijaykumar. 2015. Using lessons learned from mapping study to conduct a research project on knowledge management in software testing. In *41st Euromicro Conference on Software Engineering and Advanced Applications (SEAA).* Madeira, Portugal, 208–215.
[27] S. Staab, R. Studer, H. P. Schurr, and Y. Sure. 2001. Knowledge processes and ontologies. *Intelligent Systems* 16 (2001), 26–34.
[28] J. Storey and E. Barnett. 2000. Knowledge management initiatives:learning from failure. *Journal of Knowledge Management* 4 (2000), 145–156.
[29] M.H. Zack and M. Serino. 2000. Knowledge Management and Collaboration Technologies. In *Knowledge, Groupware and the Internet.* Butterworth, 303–315.