e-Informatica Software Engineering Journal, Volume 12, Issue 1, 2018, pages: 51–78, DOI 10.5277/e-Inf180103

Knowledge Management in Software Testing: A Systematic Snowball Literature Review

Krzysztof Wnuk*, Thrinay Garrepalli*

*Software Engineering Research Group, Department of Software Engineering, Blekinge Institute of Technology, Karlskrona, Sweden

krw@bth.se, thga14@student.bth.se

Abstract

Background: Software testing benefits from the usage of Knowledge Management (KM) methods and principles. Thus, there is a need to adopt KM to the software testing core processes and attain the benefits that it provides in terms of cost, quality, etc. Aim: To investigate the usage and implementation of KM for software testing. The major objectives include 1. To identify various software testing aspects that receive more attention while applying KM. 2. To analyse multiple software testing techniques, i.e. test design, test execution and test result analysis and highlight KM involvement in these. 3. To gather challenges faced by industry due to the lack of KM initiatives in software testing.

Method: A Systematic Literature Review (SLR) was conducted utilizing the guidelines for snowballing reviews by Wohlin. The identified studies were analysed in relation to their rigor and relevance to assess the quality of the results.

Results: The initial resulting set provided 4832 studies. From these, 35 peer-reviewed papers were chosen among which 31 are primary, and 4 are secondary studies. The literature review results indicated nine testing aspects being in focus when applying KM within various adaptation contexts and some benefits from KM application. Several challenges were identified, e.g., improper selection and application of better-suited techniques, a low reuse rate of software testing knowledge, barriers in software testing knowledge transfer, no possibility to quickly achieve the most optimum distribution of human resources during testing, etc.

Conclusions: The study brings supporting evidence that the application of KM in software testing is necessary, e.g., to increase test effectiveness, select and apply testing techniques. The study outlines the testing aspects and testing techniques that benefit their users.

Keywords: KM, software testing, knowledge, systematic literature review

1. Introduction

Software testing is a complex task and requires various activities, techniques, tools, and resources [1]. Knowledge Management (KM) is extensively used in software testing and influences software testing processes, methods and models [1]. KM helps to capture, share, distribute, and understand knowledge that helps to create a competitive advantage for organizations [2], e.g., by utilizing previous project experience or sharing testing experience between team members [3–6].

The increasing complexity of software systems combined with the advent of distributed development models put more pressure on software organizations to manage organizational knowledge and intellectual capital. Also, there is a significant loss of intellectual capital due to staff turnover, restricted or limited knowledge [6–8]. The adoption of KM principles can help software testing experts to advance knowledge reuse and to encourage management discus-

sions across the organization. There are numerous benefits of applying KM in software testing such as [3, 4, 8–11]:

- increasing test effectiveness,
- decreasing costs, time and effort,
- determination and application of more suited testing techniques,
- determination and application of more suited testing techniques,
- enhancing the quality of results,
- supporting decision-making process.

Explicit knowledge testing can be documented and accessed by multiple individuals, e.g., in test manuals, procedures, test artifacts, test planning, test design specifications, testing logs [12, 13]. Tacit testing knowledge is subjective and hard to document [12] as it mainly forms test execution experiences and discussions with software testers etc. [14]. Insufficient KM during software testing leads to several negative consequences, e.g., low reuse of software testing knowledge, barriers in software testing knowledge adaption, a poor sharing environment of software testing knowledge, difficulties in optimal planning resources [1, 4].

This study focuses on testing aspects as activities during the testing process and the resulting artefacts, i.e. test planning, execution and test result analysis [5–7,15,16], test case design [9,17,18] and testing phases [14, 19]. It also focuses on testing techniques used to execute a software system and find errors [20]. The aim is to focus on the importance of KM in various software testing aspects as the literature lacks studies which focus on identifying the testing techniques that benefit from KM application. Therefore, this work concentrates on identifying the test design, execution, and analysis techniques that help from the KM application. It also explores the related challenges resulting from insufficient KM.

The paper is organized as follows: Section 2 focuses on giving the necessary theoretical background about KM and software testing and their corresponding practices along with its potential contribution to this study. Section 3 provides the research design details and objectives of this study and the addressed research questions. Section 4 contains the details of the research methodology, including considered methodologies and the conducted data analysis. Section 5 depicts the process of conducting the snowballing iterations while Section 6 analyses the results of the literature review. Section 7 lists the identified challenges and implications for research and practice, while Section 8 discusses the limitations of the study. The conclusions are formed in Section 9.

2. Background and related work

2.1. KM in software testing

Testing experience, as well as testing knowledge, are needed to gain a deeper understanding of the used testing techniques [21, 22]. However, testers do not tend to share the knowledge or information that they gain when using various testing techniques [7]. This implies that they miss an opportunity of sharing experiences and learning from each other, which limits their overall knowledge.

Many testers are self-educated and have limited education on the subject [23]. They require additional training [24]. This limited knowledge also results in a limited view about software testing techniques [25]. Technology transfer between research and industry is often limited, in consequence, not all new testing techniques are directly applied in industry [26].

Testers gain various types of knowledge and experiences from their work in software projects. Sharing this knowledge can help to avoid making similar mistakes and optimize testing activities. Efficient organizational knowledge sharing requires establishing efficient KM practices for knowledge creation, documentation, and management.

The primary objective of KM in software testing is to transfer testing knowledge and experience between individuals in the same way as testing documentation as well as utilizing tacit knowledge for supporting test design, execution, and interpretation. KM supports test planning, test result analysis and test outcomes [27]. The test design phase is also heavily dependent on

KM as it involves findings the test conditions and objectives and choosing the relevant information to implement planned test cases. Knowledge also helps to establish the satisfaction criteria against the testing outcomes.

KM supports testing techniques selection as it is often based on testers' experience and intuition, gained from various sources, such as testing the previous versions of the system, involvement in analysing and fixing the defects, working on development and maintenance as well as working with similar software systems [27]. Finally, KM strategies help to increase the effectiveness and efficiency of product testing [28]. Applying KM in software testing is essential to increase the testing level and enhance software quality [26].

2.2. Related work

Several studies looked into the state of the art solutions and practice of utilizing KM for software testing, e.g., [26]. Desai et al. [6] outlined the challenges faced due to the lack of KM, such as less re-use of software testing knowledge, barriers in the transfer of software testing knowledge, difficulties in achieving the most optimum distribution of human resources, etc. Taipale et al. [29] discussed KM practices in software testing and how to enhance the testing practices using KM strategies in organizational units. Wei et al. [14] discussed the implementation of the KM framework in mobile software systems testing and how it benefits the organization concerning decreased costs and increased productivity. Beer et al. [27] stressed that exploratory testing (described as simultaneous learning, test design, and test execution) requires substantial experience. De Souza et al. [1] discussed KM about software testing aspects, testing processes, test phases, test cases and testing techniques, etc. In a similar way, aspects that are related to KM practices are discusse, they encompass, e.g., KM model, knowledge capturing, knowledge elicitation, knowledge retrieval, knowledge dissemination. KM has been investigated for two decades and many tools and techniques were suggested, e.g., methods, tools, techniques, knowledge ontologies, knowledge maps, intranets, just to name a few. Most of the studies focus on storing explicit rather than tacit knowledge and only some studies provide empirical evidence [4, 6, 7, 29, 30], e.g., storage and re-use of test cases [1]. At the same time, many studies focus on implementing a KM framework to strengthen software testing process [5, 7]. From the surveyed papers, the following research gaps were identified:

– storing tacit knowledge and using appropriate testing aspects and techniques,
– focusing on the testing aspects and testing techniques and their importance in utilizing KM practices.

To summarize, so far no study has focused on identifying what type of knowledge is required to perform a particular kind of software testing techniques. This paper fills this research gap by explicitly focusing on finding out the testing techniques and the testing aspects that benefit from KM.

3. Research questions

This study has two goals: 1) to investigate which software testing aspects and techniques receive more attention when applying KM and 2) to identify the challenges faced due to the lack of KM practices.

These goals are detailed into the three research questions:

– RQ1: What are the KM and testing aspects that receive more attention while applying KM in software testing literature?
  Motivation: RQ1 is inspired by De Souza et al. [1] who conducted a systematic mapping to find out the studies related to KM in software testing. De Souza stated various testing aspects that get attention while applying KM in software testing literature but lacked the analysis of the importance of each testing aspect for KM. This paper focuses on identifying which testing aspects investigated in the literature in empirical studies.
– RQ2: What software testing techniques benefit most from the application of KM practices?

Motivation: RQ2 is partly based on the work of de Souza et al. [1] and Beer and Ramler [27], who claimed that exploratory and Ad-hoc testing techniques benefit from the application of KM. The paper further explores De Souza's findings as well explores more techniques which might be considered as important in the context of KM.

– RQ3: What are the challenges faced due to the lack of KM practices in software testing?
Motivation: RQ3 is inspired by Liu et al. [30] who identified the challenges that are faced due to the lack of KM. This article further explores their findings and identifies additional challenges that are faced due to the lack of KM.

4. Research design and methodology

Many authors stressed the importance of utilizing systematic approaches for building knowledge through literature, such as evidence-based software engineering [31], information systems research [32] and results from synthesis [33]. A systematic literature review study was performed for the needs of this article in which the snowballing literature review method suggested by Wohlin [34] was used, rather than a database search based review because 1) it was difficult to formulate a precise search creating the risk of receiving many irrelevant and superfluous papers [34–36], 2) the interdisciplinary nature of the studied area makes the database selection and the search string construction challenging [34,37], 3) snowballing is comparable to the multiple database searches and 4) it is suitable for expanding existing literature reviews with new aspects.

The principle benefits of utilizing snowballing are that it focuses on the cited or referenced papers, which in comparison with the database approach reduces the noise. Moreover, it is usually true that new studies cite one article among the previous pertinent studies or a systematic literature review study already done in a specific area [34].

Snowballing involves deriving the tentative start set of papers and conducting forward and backward snowballing in iterations. Wohlin proposed to use Google Scholar to discover the start set of papers and to evade the publisher bias [37]. However, in certain circumstances, Google Scholar provides significant noise and low certainty in terms of academic quality [38]. Thereby, the Engineering Village database was selected as the start set identification. Knisley recommended the Engineering Village as a prior database to search for papers in comparison with other databases [38]. Also, it was discovered that the Engineering Village offers auto stemming and related papers availability as additional features.

4.1. Data analysis

The qualitative data collected during the literature review were analysed using the narrative analysis technique that helps to create the narrative summary of the resulting studies for synthesis purposes [39]. The narrative analysis does not focus on one specific theme and therefore helps to discover recurring themes from the obtained data. The narrative analysis was used to develop the paper categorization presented in Section 6.1 and the testing aspect and techniques listed in Sections 6.4 and 6.5 The first and the second authors iteratively analysed the results and developed the themes.

The authors also applied grounded theory analysis [40, 41] mainly because they had pre-considered thought regarding the information they needed, contrary to what is recommended by Glaser and Strauss [42]. In the same vein, thematic analysis was excluded as an alternative analysis approach because it searches for the repetitions of themes within the accessible information [43].

4.2. Snowballing procedure

4.2.1. Deriving the tentative start set of paper

Step 1: Search string and database selection. Getting a representative and precise start set of papers is equally challenging for snowball as it is for the database searches [35]. A compre-

hensive search string was developed avoid the problem of inconsistent terminology.

The search string was formulated based on the research questions and the keywords derived from them, including the synonyms and alternatives. It was iteratively developed and it constantly enhanced available knowledge when relevant papers identified manually were read. When there was agreement and confidence that the search string covered the aspects that were the goal of the study, a pilot search was performed in which the Engineering Village database was queried and the first 500 results were analysed. Both authors screened these results independently and later compared and discussed relevance. The resulting search string terms are outlined in Table 1 and grouped into the two categories connected with the Boolean operators.

Table 1. The keywords used to query the Engineering Village database and identify the start set papers

Software testing keywords
Software testing – A1
Software test – A2
KM related keywords
KM – B1
Tacit knowledge – B2
Explicit knowledge – B3
Knowledge creation – B4
Knowledge acquisition – B5
Knowledge sharing – B6
Knowledge retention – B7
Knowledge valuation – B8
Knowledge use – B9
Knowledge discovery – B10
Knowledge Integration – B11
Knowledge theory – B12
Knowledge – B13
Knowledge engineering – B14
Experience transfer – B15
Technology transfer – B16

The search string run in the Engineering Village database was composed of the following Boolean formula: ("A1" OR "A2") AND ("B1" OR "B2" OR "B3" OR "B4" OR "B5" OR "B6" OR "B7" OR "B8" OR "B9" OR "B10" OR "B11" OR "B12" OR "B13" OR "B14" OR "B15" OR "B16").

Step 2: Tentative start set of papers. The search string was executed in the Engineering Village database and resulted in 4832 hits. Next, the inclusion criteria outlined below were applied, including only the papers written in English (IC1), which resulted in 2774 candidates and additional 85 were removed as they were not peer-reviewed (IC2). Next, the 2689 candidates were screened and 2404 were excluded based on title screening (IC4). The abstracts for the remaining 285 candidates were read and 63 papers were accepted. Later the introduction and conclusion sections of the 63 papers were read and as a result, 32 candidates were kept. Finally, the full papers were read and independent judgments regarding if they should be included or not were performed. The application of all inclusion criteria and the full read resulted in 16 candidate papers. These were analysed looking at their authors and publication venues. There were 3 papers which were excluded because they had a low number of references or citations and were less relevant for the scope of this study. As a result, the 13 papers that were left were heavily cited and had the most relevant references that increased the likelihood of better coverage of relevant studies [34]. The following inclusion criteria were used:

- IC1: Articles that are written in English and are published between 2003–2015. The primary reason behind choosing papers from 2003 or later is that KM initiatives in software testing were established around 2003 [1],
- IC2: Peer-reviewed articles published in relevant venues (conferences, workshops or journals in software engineering, software testing and knowledge management, computer science, information technology and science, computing and computer applications)
- IC3: Articles available in full text
- IC4: Articles that focus on KM practices used for supporting software testing (design, execution, and analysis) and/or deal with the industrial challenges due to the lack of KM under software testing.

4.2.2. Forward and backward snowballing in iterations

On the start set of 13 papers [1, 3–7, 9, 14, 27–30, 44], five iterations of backward and forward snowballing were performed, see Table 2 for details. Backward snowballing was conducted by looking at the references of each paper in parallel with forward snowballing by looking at citations. Google scholar was used to extract the citations for each of the papers. Both references and citations were inserted in an Excel file where both titles and abstracts were collected. The second author screened these citations and references in each of the iterations and categorized them into NO, MAYBE and YES categories. Next, the first author screened the MAYBE and YES papers and used his judgment whether they were relevant. After a discussion and reaching an agreement, the relevant candidates were included in the next iteration. The same inclusion criteria were used for all snowballing iterations.

4.3. Data extraction and synthesis

The data extraction properties outlined in Table 2 were derived during several discussions between the authors. The data were extracted into a spreadsheet where categories are mapped to the research questions. The data analysis checklist was also developed where the fulfillment of each of the aspects could be partial or full.

The second author performed the data extraction, supported by the discussion with the first author. The extracted data were synthesized by performing a narrative analysis as per the guidelines provided by Cruzes et al. [39] and Rodgers et al. [45]. Patterns in data were identified, and these patterns were grouped into various themes. To strengthen reliability, rigor and relevance criteria were applied for each paper, see Section 4.4.

4.4. Quality assessment based on rigor and relevance

The rigor and relevance assessment method was utilized according to the guidelines provided by Ivarsson and Gorschek [46]. Previous authors [47, 48] demonstrated that rubrics built the unwavering quality of the assessments as per the terms of inter-rater agreement among the researchers. The second author performed the data extraction supported by the first author who evaluated the results with objectivity in mind. Each paper was allotted with a score utilizing the objective criteria, customized for this study. No significant changes to the rigor and relevance scores suggested by Ivarsson and Gorschek were made, see Table A in Appendix A.

The secondary studies (literature reviews) were evaluated using different criteria. Firstly, it was evaluated if the motivation behind conducting the literature review was clearly stated. Secondly, the review process was examined, and a search for the precise descriptions of the search strategies and search strings, clear definition of acceptance criteria and unambiguous judgments of the validity of the identified studies was conducted. There was also a search for methodological flaws [49]. Finally, the empirical support for the claims provided by the secondary papers was sought and it was checked how well the empirical data were analysed. The fulfillment of each of the criteria was estimated as *Yes, No, Maybe.*

5. Results of the snowballing iterations

As a result of the above examination 13 papers (marked as P1 [5], P2 [3], P3 [4], P4 [6], P5 [14], P6 [1], P7 [29], P8 [9], P9 [30], P10 [27], P11 [28], P12 [7], P13 [44]) were chosen for the start-set from the 4832 candidate papers obtained from the Engineering Village database. Table 3 presents the summary of the snowballing iterations regarding the number of references and citations screened in each iteration.

Based on backward snowballing in five iterations, 843 references were thoroughly examined and evaluated among which 137 were removed based on the publication type, 7 did not match the Language criteria, 40 were duplicates, 323 were dismissed based on title screening, 84 were dismissed based on the year of publication, 202 were dismissed after reading the abstract, 11

Table 2. Data extraction strategy

Category	Data Properties	Mapping to Research Questions
General information	Author(s), Title, Publication Year, Abstract, Conclusions	RQ1, RQ2, RQ3
Type of Study	Evaluation study, Validation study, Proposing a solution, Opinion papers, Personal experience papers, Observational research	RQ1, RQ2, RQ3
Research methods	Case study, Survey, Mapping study, Experiment, Grounded theory, Action research, Unclear	RQ1, RQ2, RQ3
Study aims research outcomes	Does the study specify aspects of software testing that receive more attention while KM is applied?	RQ1
	Does the study specify any software testing techniques i.e., test design, test execution, test result analysis that benefit from the application of KM in Software testing?	RQ2
	Does the study provide any problems or challenges reported due to lack of KM practices in software testing?	RQ3
Data analysis	Aspects of software testing that receive more attention while KM is applied in software testing are properly specified (yes/no/partially)	RQ1
Data analysis	Software testing techniques that benefit from the application of KM are properly specified (yes/no/partially)	RQ2
Data analysis	Problems or challenges faced due to lack of KM practices in software testing explained (yes/no/partially)	RQ3

were excluded after reading the full text and 26 were dismissed as their full text was not available. Finally, 12 papers were obtained based on backward snowballing in five iterations.

During forward snowballing, 614 citations were analysed in five iterations among which 43 turned out to be duplicates, 89 citations were removed based on the publication type, 248 were excluded based on the title, 203 were removed after reading the abstract, 7 were omitted based on the language in they were published, i.e. other than English, 13 papers were removed after reading the full text and 2 were removed due to the unavailability of a full text. Finally, 10 papers were selected.

6. Literature review results analysis

35 papers were identified in five snowballing iterations among which 31 were primary and 4 were secondary studies. Figure 3 depicts the paper distribution over the years. Only five papers were written between 2003 and 2005 indicating that the research in KM in software testing became more common after 2003. Much of the work under KM in software testing was done during 2006–2009 meaning that the organizations started taking interest in utilizing KM in software testing to gain benefits and overcome the issues associated with software testing due to the lack of KM. Still, we see no clear increasing trend in Figure 1.

Out of 35 analysed papers 21 studies are conference articles indicating that conferences are the primary venue for communicating research in KM for software testing. Journals correspond to 34% of the studies (14 out of 35). Table A in Appendix A provides the list of publication venues. It appears to be clear that not only software engineering venues are utilized for communicating research about KM in software testing.

Next, it was analysed which of the three RQs each of the papers addressed. It turned out that 23 out of 35 studies reported various KM aspects (RQ1) during software testing, 12 papers discussed challenges (RQ3) faced due to the lack of KM practices in software testing, while ten stud-

Table 3. The summary of the number of citations and references screened in each snowballing iteration. I – Iteration, FS: Forward Snowballing, BS: Backward Snowballing, D – Duplicate, T – Based on Type, N – Based on Name, Y – Based on Year, L – Based on Language, EA – Excluded after reading the abstract, EF – Excluded after reading the full text, FN – Full text not available, IA – Included after reading Abstract, IF – Included after reading full text

Iteration	FS/BS	Papers rejected from FS and why	Papers rejected from BS and why	Papers Considered from FS and why	Papers Considered from BS and why
I1	140/346	D: 6, T: 19, N: 41, EA: 66, L: 3	T: 51, L: 5, D: 16, N: 105, Y: 44, EA: 87, EF: 11, FN: 20	IA: 4, IF: 2	IA: 4, IF: 2
I2	164/262	D: 13, T: 31, N: 55, L: 4, EA: 54, EF: 4	N: 135, D: 5, Y: 21, T: 37, L: 2, FN: 5, EA: 53	IA: 2, IF: 1	IF: 4
I3	294/178	D: 19, T: 39, N: 145, FN: 2, EA: 79, EF: 9	T: 44, D: 13, N: 61, Y: 14, EA: 44, FN: 1	IF: 1	IA: 1
I4	12/50	D: 4, N: 5, EA: 3	D: 4, N: 19, T: 5, Y: 5, EA: 16	–	IF: 1
I5	4/7	D: 1, N: 2, EA: 1	D: 2, N: 3, EA: 2	–	–

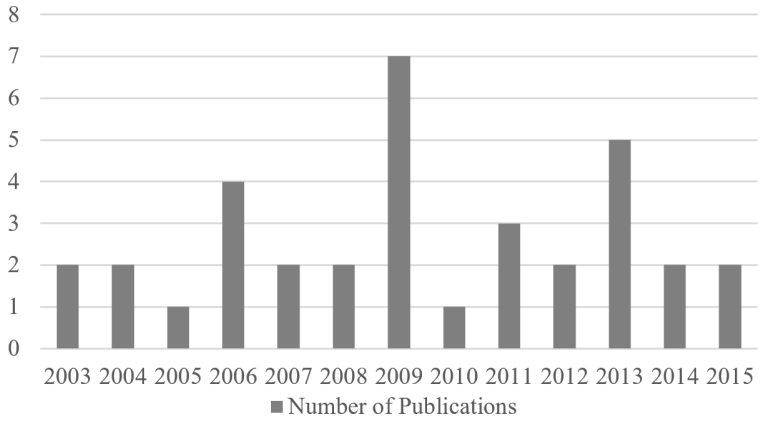

Figure 1. Publications over the years

ies focused on KM in testing techniques (RQ2) and helping testers to select better testing techniques.

6.1. Categorization of papers based on research methodology and studytype

In the analysed group 31 studies were primary studies and four were secondary studies (two systematic literature reviews and two systematic mapping studies). The 31 primary studies were categorized according to the research methodology (i.e. case study, survey, experiment, etc. as defined by Runeson et al. [50] and the type of study (i.e. evaluation, proposal, solution, opinion, experience based, etc. and constraints as defined by Wieringa et al. [51].

Evaluation research which utilized the case study research method dominated among the chosen papers – 16 articles [P2, P3, P4, P5, P8, P11, P12, P13, P18, P20, P25, P30, P31, P33, P34, P35] of which 3 were interview studies [P2, P11, P31], categorized as qualitative case studies. Evaluations using frameworks were found in 5 papers [P9, P14, P15, P27, P29]. The framework-proposal category encompassed 4 papers [P17, P22, P23, P26]. Two papers

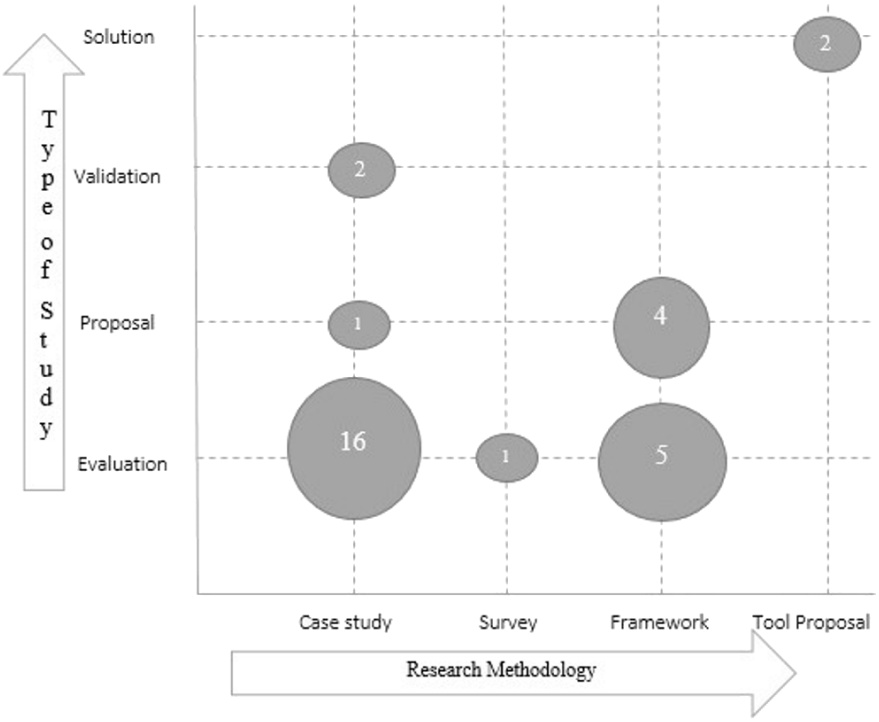

Figure 2. Categorization based on the type of the study and the research methodology aspects

were classified as case study-validation [P7, P10] and two papers as the tool a proposal-solution [P21, P32]. Finally, the categories such as case study-proposal [P19] and survey-evaluation [P1] received only one paper, see Figure 2 for details.

6.2. Quality assessment based on rigor and relevance

Figure 3 depicts the Rigor and Relevance analysis results where the primary studies are categorized into four quadrants (A, B, C and D) according to their rigor and relevance scores. The process of classification is detailed below.

- Papers which fall under the score from (0–1.5) are categorized as low rigor and those that fall in between the score of 2 as high rigor.
- Papers with the score from (0–2) are considered to have low relevance and the papers that fall score 2.5 or above are considered to have high relevance.

Altogether 13 studies were characterised as having high rigor and high relevance, quadrant A in Figure 3, and these outcomes are the most reliable. Also, 12 studies were classified under Quadrant C with high relevance and low rigor. Six papers fell under category D, which means they were characterised by low rigor and low relevance, where relevance scores prevail over rigor scores, see Table B in Appendix B for rigor and relevance scores.

6.2.1. Quality assessment of secondary studies

Table 4 shows the results of the quality assessment of secondary studies [P6, P16, P24, P28]. It was concluded that the four identified secondary studies present high quality and therefore trustable literature reviews.

6.3. KM aspects discussed in the selected studies (RQ1)

The subject of 23 studies were KM aspects which testers focus on during software testing, see Table 5. It occurred that 13 studies focused on knowledge representation while 12 studies focused on knowledge capturing. There were 8 papers which focused on knowledge management systems and 8 papers presented knowledge management models.

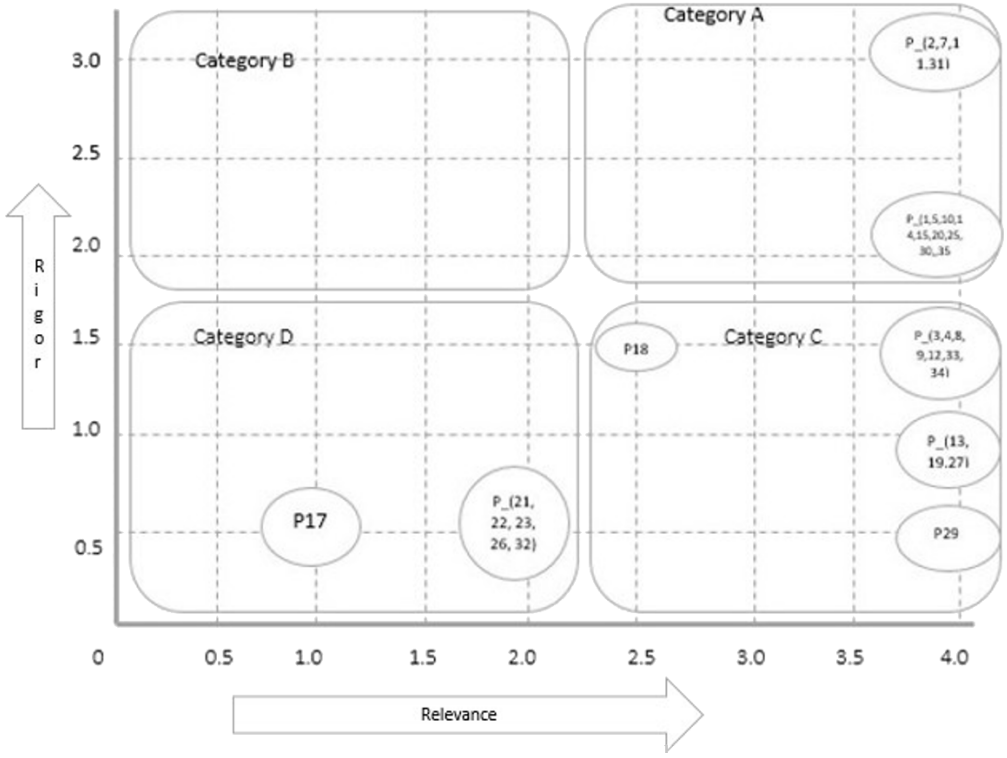

Figure 3. Rigor and relevance analysis results

Table 4. Quality assessment for secondary studies

Quality assessment question	P6	P16	P24	P28
Is the motivation behind conducting systematic literature review and mapping clearly expressed and defined?	Yes	Yes	Yes	Yes
Is the process of conducting systematic literature review or mapping clearly stated?	Yes	Yes	Yes	Partial
Is there any empirical evidence for the stated systematic literature review or mapping study?	Yes	Yes	Yes	Yes

KM Systems (KMS) are necessary to enable successful KM. Huseman and Goodman [52] consider KMS as an essential source for competitive advantage while Rajiv and Sarvary [53] claim that organizations without strong KM systems work inefficiently, which consequently influences their quality of work.

Eight studies [P1*, P3, P4, P5*, P8, P9, P12, P19] proposed various KMSs and discussed their importance for software testing. KMS were used to store, manage, search and share various kinds of knowledge with the help of knowledge documents [P3, P4, P9, P19], to store tacit knowledge to be reused by searching the relevant documents and resolve any raising issues [P12] or store and maintain daily and weekly tester discussions in a knowledge map [P8] or, also, store the experience gained in earlier testing cycles [P5*].

KMS provide several benefits, e.g., they help to reduce effort during testing, increase software quality [P1*, P5*], help the organizations adapt to turnover and faster respond to changes and downsize by making an experience of each individual widely accessible [P4]. It is interesting to note that all the eight papers focused on the importance of KMS and their benefits, rather than the details of how these systems are built and what strategies were used during their development. Thus, future research should focus on the strategies to be used to build an effective KMS and its usage in case study context.

Table 5. Research Focus on KM aspects over the years[a]

KM aspect	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015
KM System (KMS)	–	–	–	P12	P5*	P19	P3 P9	–	P1* P4	P8	–	–	–
KM Models	P27	–	–	–	–	P19	P3 P9	–	P1*	P8 P25*	P2*	–	–
Knowledge Representation	–	P15*	P14*	P18 P29	P5*	P19	P3 P9	–	P34	P8	P32 P33	P22	–
Knowledge capturing	–	P15*	P14*	P18	P5*	P19	P3 P9 P35*	P26 P35*	P1*	–	P2*	P21	–
Knowledge retrieval	–	P15*	P14*	P18	P5*	P19	P3 P9	–	–	–	P2*	–	–
Knowledge dissemination	–	–	–	P23	–	–	–	–	P1*	–	P2*	P21	–
Knowledge elicitation	P27	–	–	–	P5*	–	–	–	–	–	P2*	–	–
Knowledge packaging	–	P15*	P14*	P18	–	–	–	–	P1*	–	–	–	–
Knowledge evolution	–	P15*	P14*	P18	–	–	–	–	–	P8	–	–	–
Knowledge acquisition	–	–	–	P29	–	–	–	–	–	–	P32 P2*	P21	–
General[b]	P17	P28	–	–	P6 P7*	P10*	P13 P20* P30*	–	–	–	P16 P11* P24	P6	P31*

[a]An asterisk (*) indicates a paper with high rigor and relevance scores.

[b]Not focusing on any KM aspects but provides tools that support KM and knowledge or just defining KM aspects without implement them.

KM Models are models used for knowledge management and knowledge process aspects, such as knowledge carriers and knowledge technologies. Eight studies [P1*, P2*, P3, P8, P9, P19, P25*, P27] focused on KM models. Three studies [P3, P9, P19] used communication databases enriched by knowledge maps and testing knowledge databases. KM models can also be created based on reusable test case repositories extracted from similar projects or individuals' tacit knowledge and testing projects data by test specialists [P8].

KM models bring several benefits, e.g., increase test case reuse [P8], increase quality and decrease development time [P1*], develop testing lesson learned systems [P2*], or identify gaps in KM practices and fill in these gaps with potential solutions [P27].

Two models for building KM models were identified. The first model contains four phases: 1) absorption is related to acquiring new knowledge from the external environment of the organization, i.e., experts are brought into the organization, 2) diffusion concerns the dissemination of knowledge among individuals in the organization, i.e., these issues which are mostly resolved in email/discussion lists, search engines, best practices, 3) generation involves the improvement of new knowledge and the procedure of turning tacit knowledge into explicit information, i.e. through brainstorming sessions, joint design and source studies, 4) exploitation is referred to as the commercialization of knowledge [P27]. The second model contains five steps: 1) identify knowledge needs, 2) create knowledge, 3) store knowledge, 4) organize knowledge and 5) share knowledge [P25*].

The identified KM models focus on acquiring, improving, disseminating and storing testing knowledge. These findings may help to understand that KM models contributed to the increase in the reuse of testing knowledge in some papers [P25*, P27, P3, P9, P18].

Knowledge Representation focuses on representing test knowledge through various tools that support knowledge storage, e.g., ontologies, Software Requirement Specifications (SRSs), Test Procedure Specifications (TPSs), etc. Thirteen studies [P3, P5*, P8, P9, P14*, P15*, P18, P19, P22, P29, P32, P33, P34] focused on knowledge representations which were categorized into:

- Ontologies [P3, P8, P9, P19, P22, P29, P32], TPSs and SRSs as explicit knowledge representations. Ontologies serve as a medium in describing relative concepts, attributes and relations connected with knowledge [P3, P9, P19], they are also used to generate test cases for GUI testing [P34], or as the knowledge representation for performance testing [P22]. Ontologies were also used as knowledge representations for test case reuse [P8] and for supporting acquisition, organization, reuse and sharing testing knowledge [P29, P32]. Testing activities can be performed based on the ontologies associated with a software project [P29, P32]. Ontologies support test case generation from various artefacts in dissimilar domains [P33] or for organizational discussions [P2]. These results suggest that developing an ontology that possesses all of the above characteristics could result in generating productive testing outcomes. It is also worth exploring how to use these ontologies and strategies rather than how to develop them [P3, P9, P19, P22, P29, P32].
- Characterization schema [P14*] that contains test objectives, test scope, required testing technique, test case generations, and test tools is applied in post-project evaluations and summaries of experiences from testing activities. A characterization schema is a tool that supports knowledge representation. Vegas et al. developed and empirically evaluated the schema for assisting testing technique selection that generates a valid test case for a given project [P14*]. This study suggests effective schema generation for test design technique selection.

Knowledge Capturing includes codifying and documenting analytical testing knowledge in a manner that individuals can adapt and re-use for specific purposes. 13 studies [P1*, P2*, P3, P5*, P9, P14*, P15*, P18, P19, P21, P26, P33, P35*] focus on capturing testing knowledge in terms of using: 1) lessons learned, experiences, successes and failures [P2*], 2) knowledge of individuals from discussion forums and documents [P3, P9, P19], 3) external knowledge and its relation to internal knowledge [P1*], 4) feedback given by both producers and consumers using characterization schemata [P14*, P15*, P18], 5) experience and knowledge gained from applying various testing techniques [P26]. Three papers specified capturing general testing knowledge, e.g., knowledge and experience are recorded and represented to as a substantial quantity of component sequence in an XML file [P35*], recorded into a formal form (issue spreadsheet) [P5*] or in wikis [P21].

What is surprising is that the identified studies focus on Externalization (tacit to explicit), Internalization (explicit to tacit) aspects leaving aside Socialization (tacit to tacit) and combination (grouping all the explicit knowledge).

Knowledge Acquisition is the focus of four studies [P2, P21, P29, P32] with the help of wikis [P21], ontologies [P29, P32] or lessons learned [P2]. Surprisingly, the studies do not outline any process that needs to be executed while defining the rules unlike [P14*] which outlined such a 10-step process for knowledge capture. It can thus be concluded that researchers should focus on knowledge acquisition processes and techniques.

Knowledge Elicitation is the focus of three papers [P2*, P5*, P27]. They utilized: 1) an architectural model for knowledge elicitation based on the lesson learned systems (a KM manager as well as expert testers verify the elicited knowledge) [P2*], 2) eliciting expert knowledge whenever it is required and capturing it in spreadsheets [P5*] or 3) acquiring knowledge from the external environment during the absorption

phase [P27]. The architectural model presented by Andrade et al. [P2*] focuses on 1) defining the structure of software testing lessons learned, 2) setting up the procedures for the management of lessons learned and 3) supporting the design of tools that manage lessons learned. Despite promising results, papers [P5*, P27] proposed only the knowledge elicitation tools and failed to provide the processes for knowledge elicitation.

Knowledge Dissemination covers disseminating testing knowledge through various KM practices, such as internalization, externalization, combination, and socialization. Only four studies [P1*, P2*, P21, P23] focused on the ways to disseminate testing knowledge. In two studies [P1*, P21], knowledge is available in a useful, readable format to the individuals who need it. Andrade et al. used active knowledge dissemination, where the software testing lesson learned systems disseminate the lessons learned as per various parameters (e.g., scattering of conceivably helpful lessons learned towards the beginning of every testing activity using a testing activity descriptor). The second way is passive knowledge dissemination where the user is responsible for communicating the software testing lessons learned system and asking for the conveyance of lessons learned [P2*]. Lee developed a KM framework with seven cyclic steps for disseminating testing knowledge: identify relevant knowledge, collect the knowledge that is needed, adapt knowledge, organize knowledge in a readable format and apply the knowledge assets to situations where there is a need for it [P23].

Knowledge Retrieval covers returning testing information in a structured format contrary to just capturing the knowledge. Eight studies [P2*, P3, P5*, P9, P14*, P15*, P18, P19] focused on knowledge retrieval mechanisms and tools or artefacts that support them. Three studies [P14*, P15*, P18] provided a systematic structured format of storing the knowledge regardless of the testing technique.

Knowledge Packing covers strategies or methods used in packing captured knowledge, e.g., knowledge databases. In [P14*, P15*, P18], a characterization schema encompassing various attribute levels, such as tactical, operational and historical, was developed for packaging the experience of individuals for various testing activities. In [P1], knowledge packing is done with the aid of a KM System by following the knowledge lifecycle from acquisition to an application.

Knowledge Evolution covers evolution aspects, such as the evaluation and maintenance of testing knowledge. There were four studies [P8, P14, P15, P18] covering this aspect. Three studies propose a characterization schema [P14*, P15*, P18] where a librarian maintains the repository by taking care of the coherence of the information it contains and updates the repository based on the feedback provided by consumers and producers. In one study, a knowledge analyst is assigned to analyse conducted discussions and update the knowledge repository [P8]. Three studies consider knowledge evaluation as the most important element [P14*, P15*, P18] but fail to provide methods, steps and strategies for supporting knowledge evolution and thereby recommendations for software organizations.

6.4. Software testing aspects that benefit from the application of KM practices (RQ1)

In the studied group 9 studies [P2, P3, P6, P9, P16, P19, P23, P24, P27] provide only a general discussion about KM and how to apply knowledge in software testing, however, they lack discussions on specific testing aspects. Two studies [P5, P7] focused on dealing with KM applied in a project where testing is outsourced to a third party (this is not considered a testing aspect). The only difference is that the process is carried out elsewhere but all the activities of this process are similar. The remaining papers are analysed according to the following testing aspects that are summarized in Table 6 and the research focus on the testing aspects over the years is summarized in Table 7.

Testing process. Seven studies [P1*, P4, P12, P21, P25*, P29, P32] focus on KM in the context of a testing process (test planning, test case design, test execution and test result analysis), see Table 7.

Table 6. Description of the testing aspects analysed in the study

Testing Aspect	KM Utilization
Testing process (7 studies)	**Test planning:** The main aim is to manage knowledge in the context of test scenario creation, test cases design, data preparation as well as a test environment. **Test execution:** the goal is to manage knowledge during test execution, in a number of test cycles on the basis of the project. For example, most of the projects run two test cycles by adhering to time and cost conditions. **Test result analysis:** The aim is to manage knowledge during test result analysis.
Test cases and code (3 studies)	Manage knowledge about the test cases. For example: reusing the test cases Manage knowledge about the test code (which takes into account test scripts and drivers)
Testing phases (2 studies)	Apply KM strategies to phases in software testing such as unit testing, component testing, integration testing, system testing, acceptance testing, alpha testing, beta testing.
Testing techniques (10 studies)	Manage knowledge on testing techniques, to help testers to choose a better suited testing technique for designing test cases, executing and analysing the tests.
Testing types (6 studies)	Manage knowledge in a specific software testing type such as GUI, load testing etc.
Testing resources and tools (2 studies)	Manage knowledge about usage of testing tools or resources.

Table 7. Research Focus on testing aspects over the years[a]

KM aspect	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015
Testing process	–	–	–	P12 P29	–	–	–	–	P1* P4	P25*	P32	P21	–
Test case and code	–	–	–	–	–	–	P35*	P26	–	P8	–	–	–
Test levels and testing phases	–	–	–	P29	P5*	–	–	–	–	–	–	–	–
Testing technique	P17	P15* P28	P14*	P18	–	P10*	P13 P20* P30*	–	–	–	P11*	–	–
Testing type	–	–	–	P12	–	–	P33 P35*	–	P34	–	–	P22	P31*
Testing resource	–	–	–	P29	–	P10*	–	–	–	–	–	–	–
General	P27	–	–	P23	P5* P7*	P19	P3 P9	–	–	–	P2* P16 P24	P6	–

[a]An asterisk (*) indicates a paper with high rigor and relevance scores.

Abdullah et al. [P1*] focus on utilizing a community of practice for managing testing knowledge. Their model involves community of practice, KMS functionality in software testing and KMS architecture. The software testing process structures the testing knowledge, i.e. it begins with the system requirements for product specification and development, which comprises system design and coding, and proceeds to the verification and validation of product.

Desai et al. [P4] reported that KMS integrated with software testing, data warehousing, and mining helps to store and retrieve relevant knowledge and to discover different modules which are scattered along memory locations. This improves the software testing process, including test planning, test case development, test execution, test result analysis and reporting.

Nogeste et al. [P12] concluded that applying KM improves test planning and increases tacit knowledge capturing for subjects not experiences in KM. Abdou et al. [P21] advocate that a software testing process should be enriched with solutions used by Open Source Software communities regarding test planning, forming test design, considering test specifications, test implementation, deriving test cases or test suites, test execution, accepting test results.

Sirathienchai et al. [P25*] proposed three models for test planning, test preparation and test reporting which leverage on KM. Firstly, cost assessment is performed, followed by the performance evaluation of the software testing process performed by different experienced personnel utilizing the project duration, cost, and quality. Finally, a comparative financial analysis is done to find the best solution by return on investment, payback, and benefit cost ratio. The findings from the case study revealed that the long-term continuous investment on KM can improve the testing process performance more efficiently than the short-term counterpart.

Barbosa [P29] defined a testing process based on an ontology that combines the development paradigm and the testing strategy. This ontology (Reference Ontology on Software Testing) captures the relevant testing knowledge and stores it in a repository. Individuals can use the stored knowledge during testing. The main difference between ROost and other ontologies is that ROost was developed mainly following a well-established method named SABiO, which was used in several ontology development efforts [54]. ROost covers aspects related to the software testing process and its activities, artefacts that are utilized and produced by the activities, testing techniques for test case design and test environment including human, software and hardware resources similar to OntoTest [P29] with a prime motive to manage testing knowledge.

Test case and test code. Three studies [P8, P26, P35*] discussed the use of KM in test cases and test code reuse. Li et al. used an ontology representation and a knowledge model [P8] for test case reuse. Upheld by the management level, a testing center built a reusable test case repository with more than 12,000 cases, complemented with an organizational exchange library. The case study results demonstrate that the effectiveness and efficiency of the test case design and the work circumstances of test engineers and managers were improved.

Nasser et.al [P26] suggested a knowledge-based software test generation framework that permits to characterize the domain and system specific coverage criteria for different software artefacts and domains, specifically concentrating on test cases. By utilizing the custom coverage criteria, test specialists can control what tests are to be incorporated in generated test suites. For this reason, the framework used reasoning with ontologies to address the test case selection issue for re-use. Based on the ontology, test individuals can choose and select relevant previous test cases for a given project.

Li [P35*] presented a test case generation model based on test code reuse for GUI testing. The testers experience is recorded and represented as extensive segments in an XML document, where segments are the instances defined in a GUI ontology. Next, all components that are related to data elements are distinguished and marked in a sequence which is connected to data elements. In step 4, for each sequence set, data dependent elements among user related

components are recorded. In step 5, for every sort of knowledge components, the outcomes with comparative sensible relations are figured out, and if the recurrence of a legitimate connection surpasses the normal level, they are concluded as a rule for test case generation. This approach was evaluated in a case study which indicated that test case generation for GUI testing was found more efficient.

Test levels or testing phase. Two studies [P5*, P29] focused on the application of KM at a specific testing level or phases such as unit, integration or system testing. Wei and Ying [P5*] emphasizes that to deliver high quality and high productivity of testing during system testing, a KM framework should be integrated into the organization in such a way that test knowledge can be shared between individuals in the organization to sustain the system test and maintain the quality of testing. Barbosa et al. [P29] suggested an ontology which captures all the relevant knowledge that takes place during the testing phases and stores it in the repository.

Testing techniques. Ten studies [P10*, P11*, P13, P14*, P15*, P17, P18, P20*, P28, P30*] discussed KM and software testing techniques. Test case generation is one of the leading aspects of software testing and is closely linked to the selection of testing techniques [55]. Vegas and Basili [P14*, P15*] proposed a characterization schema that includes comprehensibility, the maturity level of the individuals performing testing, cost of application, inputs, dependencies, repeatability, software type, experience required to use a given technique and knowledge required to apply this technique. Beer and Ramler [P10*] claimed that ad hoc testing, subsuming casual testing methodologies and exploratory testing, is benefited through the application of KM practices, where test case design and execution are interwoven to design new test using the information and experience gained constantly.

Itkonen et al. [P11*] suggested that exploratory testing techniques benefit from adapting KM practices. Knowledge in exploratory testing can be utilized as data to guide exploratory test design and to perceive failures, e.g., s a test oracle to differentiate between an expected correct outcome and an incorrect defective outcome [56]. Moreover, knowledge together with the observed actual behavior of the tested system can be utilized to make new better tests during exploratory testing. The authors also found that as the domain knowledge, the system knowledge and generic knowledge are required to recognize failures.

Koznov et al. [P13] claimed that one of the main obstacles in transferring formal methods to industry is a lack of KM methods in this area and focusing on explicit rather than tacit knowledge, e.g., model based testing needs well defined and documented requirements which are not set in industrial projects. Tinkham and Kaner [P17] listed the factors which contribute to a tester's choice of exploration style, such as tester's skills, experience, detailed knowledge on the usage of a technique and personality (including learning style). All these factors are essentially for a perfect utilization of exploratory testing which happens through capturing, storing testing knowledge which, in turn, can be done through KM practices. Itkonen et al. [P20*] indicated that knowledge engineering techniques play a crucial role for more effective use of testing techniques.

Testing type. It deals with selecting software aspects to be tested, while the testing techniques deal with how a specific part of the software will be tested. Six studies [P12, P22, P31*, P33, P34, P35*] focused on managing knowledge in a specific software testing type, such as performance, GUI, endurance testing.

Nogeste and Walker [P12] conducted a case study which proved that a KM based regression process is necessary since regression testing is heavily dependent on tacit and explicit knowledge identification, collection, sharing and documentation.

Frietas and Vieria [P22] developed an ontology for the core knowledge used for performance testing. Since ontologies serve as the representation of domain knowledge that empowers knowledge sharing among different applications, the paper investigated the impact of ontologies on performance testing. The results indicate that this ontology can also be extended

to endurance and stress testing both of which are subclasses of performance testing for better results.

Valeh et al. [P33] applied knowledge management techniques in automated software testing to enhance the control over test generation. The results indicate that the use of ontology brings benefits for the automated testing specification of extensible test oracles which can model test specialists' mental model and lend themselves to define custom coverage criteria. The system grants control to a test specialist to determine or indicate which test cases ought to be produced and generated to increase the quality of test suites. Moreover, the produced test suite ontology is programming language independent and can be deciphered into various languages and reused.

Gentry and Shirazi [P31*] discovered that Canadian software development organizations utilize in-house manual software testers when tacit knowledge is obliged to successfully test a software application. Software development companies will probably keep manual testing in-house, since the relationships between testers and other internal employees may build the viability of testing. Software development organizations are more averse to outsource manual testing when domain specific knowledge is essential to test the product.

Nasser et al. [P34] proposed ontology-based test case generation to facilitate GUI testing and produce test cases from the users' viewpoint. GUI testing is knowledge-intensive and requires both the knowledge of GUI systems and extensive experience, hence a knowledge-based technique was suggested.

Li et al. [P35*] proposed an ontology based semi-automatic approach to generate test cases using testers' experience. The approach is based on a GUI testing ontology and examines the source code with reverse engineering techniques. Secondly, the test case generation rules are extracted from the testers' experience. The evaluation results indicate that the usage of knowledge representations and management provides support in test case generation for GUI testing in terms of greater efficiency.

Testing resources or tools. They represent resources that can be humans (testers, test managers or test analysts) or hardware (equipment, software, testing tools or supporting systems). Hardware and software resources are characterized as the testing environment which can be utilized to automate the testing methods. Two studies [P10*, P29] discuss KM concerning testing tools and resources. Beer and Ramler [P10*] focus on experience with tools when planning test case automation. Extensive experience with the setup and the utilization of tools was required and indicated as a critical issue for producing reliable test results. Barbosa et al. [P29] classified the software resources needed to perform testing (including testing tools) into primary, organizational and supporting tools.

6.5. Software testing techniques that benefit from the application of KM practices (RQ2)

Model-based testing benefits from the application of KM practices [P13]. Exploratory (ad hoc) testing is mentioned as a testing type in a few papers such as [P17], but it is also called as a testing technique in a few papers such as [P6, P10*, P16] and a testing approach also in a few papers [P11*]. In this study exploratory testing is considered as a testing technique because it is recognized as test design by [57, 58].

7. Challenges due to lack of KM practices

Twelve papers [P1*, P3, P4, P6, P7*, P9, P16, P19, P20*, P22, P25*, P32] discussed challenges faced due to the lack of KM practices, they are outlined in Table 8.

CH1: Low software testing knowledge reuse rate [P1*, P3, P4, P6, P9, P19] due to the lack of KM practices, learning and knowledge reuse are limited. Failure to capture individual knowledge and experience leads to repeating the same mistakes even though there are individuals in the organization rectify mistakes or prevent them from reoccurring. Low testing knowledge reuse

Table 8. Challenges faced due to lack of KM in software testing

Challenges faced	P1*	P3	P4	P6	P7*	P9	P16	P19	P20*	P22	P25*	P32
CH1: Low reuse rate of software testing knowledge	X	X	X	X		X		X				
CH2: Barriers in Software testing knowledge transfer	X	X	X	X		X	X	X				
CH3: Poor knowledge sharing environment	X	X	X	X		X		X				
CH4: A serious loss of software testing knowledge		X	X	X		X		X				
CH5: Optimal distribution of human resources quickly		X	X			X		X				
CH6: Determining if adequate testing has been performed			X									
CH7: Difficulties in achieving test coverage			X									
CH8: Determining if the outputs are correct			X									
CH9: Documentation is not being updated					X							
CH10: Troubleshooting documentation was inaccurate					X							
CH11: Insufficient schedule and release information					X							
CH12: Define satisfaction criteria.			X									
CH13: Increase in cost and time							X					X
CH14: Decreasing test effectiveness												X
CH15: Lacked practices for logging and tracking in testing									X			
CH16: Knowledge exchange							X		X			
CH17: Identify whether the most critical aspects of test components are tested									X			
CH18: Less support for decision making										X		
CH19: Testing knowledge not considered in planing				X								
CH20: Lacking skills											X	
CH21: Missing high severity defects											X	

also increases the effort to accomplish a task in software testing. Even if an organization has a few testing knowledge databases, most of the staff neglect to use them without the aid of KM practices, which contributes to low test knowledge reuse.

CH2: Barriers in software testing knowledge transfer [P1*, P3, P4, P6, P9, P19] and knowledge transfer without the proper application of KM practices are challenging. Also, individuals always search for the knowledge that they require and do not search the entire repository. Yellow pages can serve as a medium for rectifying this problem. Moreover, IT staff is not able to understand new testing knowledge without the aid of KM practices. The reason for this is that most of the knowledge in organizations is tacit, obtained through experience and difficult to articulate. KM representation technologies help to overcome this challenge.

CH3: Poor sharing environment for software testing knowledge [P1*, P3, P4, P6, P9, P19], the lack of a formally established, unique and sorted knowledge sharing environment negatively impacts communication. A knowledge sharing model as indicated by Sirathienchei [16] has to be accumulated within an organization to overcome this issue.

CH4: Serious loss of software testing knowledge [P3, P4, P6, P9, P19], the insufficient application of KM practices leads to knowledge and experience accumulation around only a few members of staff. Therefore, maintaining knowledge repositories and databases that store individual knowledge and make use of it is required. Also, a sudden staff turnover leads to the loss of testing knowledge.

CH5: No possibility to quickly achieve the most optimum distribution of human resources [P3, P4, P6, P9, P19], KM helps to integrate humans, processes, and technology. In a situation when management does not have any idea about the staff's knowledge level, even an ideal team will not be optimally formed in testing projects which have negative impact on achieving the optimum distribution of human resources [4].

CH6: Determination whether adequate testing is done [P4], the application of knowledge as a test oracle gives answers to the question when testing should be stopped and points out whether adequate testing is done or not. Therefore, with the help of KM practices, this issue can be resolved [6].

CH7: Difficulties in achieving test coverage [P4], the lack of KM practices hinders the identification of the untested parts of the code base. Moreover, another challenge is the fact that reusable test cases may be neglected and not stored in the repository, which increases the testing effort.

CH8: Determination whether the outputs are correct or not [P4], knowledge can be used as a test oracle to identify whether the obtained code execution results acomply with the expected outcomes [17]. Thereby, the lack of KM practices may have a negative impact on determining whether the outputs are correct because relevant knowledge is neglected.

CH9: Documentation is not updated [P7*], updating knowledge repositories is rarely done, which results in outdated repositories and relying on them when a problem occurs provides inaccurate results [29]. In such a case, knowledge evolution and maintenance methods help to allocate knowledge analysts or a specially selected person, e.g., a librarian who maintains the repository by taking care of the coherence of the information it contains and updates the repository regularly as indicated by Vegas et al. [59].

CH10: Troubleshooting documentation is inaccurate [P7*], knowledge documents that retrieve human knowledge, such as expert knowledge, are not efficiently maintained [29]. Knowledge managers and experts are to be allocated to check knowledge databases as well as to verify the knowledge that is accumulated and stored in the repository and rectify occurring problems as indicated by Andrade et al. [3].

CH11: Schedule and release of information from the testing organization to development are found to be insufficient [P7*], the documentation was not up-to-date and insufficient for planning.

CH12: Determination what decision should be made about the software when testing is completed, whether to proceed further and develop satisfaction criteria [P4].

CH13: Increase in cost and time [P32] due to the lack of relevant knowledge.
CH14: Decreasing test effectiveness [P32] because essential testing knowledge is not available.
CH15: Less support for decision making [P22] as critical knowledge is not available when needed.
CH16: Testing knowledge not adequately considered for test planning [P6] and test execution.
CH17: Insufficient test technique skills [P25*] since the test team consists of several roles which encompass different responsibilities and knowledge that needs to be communicated and shared.
CH18: No high severity defect detection is another challenge faced due to the lack of KM practices in software testing [P25*].
CH19: No methods for logging in and tracking testing activities based on experience [P20*].
CH20: Transfer of the required knowledge to testers and utilizing it [P20*].
CH21: Focusing testers' attention to ensure that the most important aspects of the tested features are tested [P20*].

7.1. Implications for research and practice

The analysis of the **KM aspects discussed in the selected studies (RQ1)** brings several implications for research and practice. Firstly, there appears to be a lot of focus on knowledge representation and knowledge capturing. This focus is unsurprising as it results in a rather technical focus on KM application, creating or managing knowledge databases or repositories or building additional tools into the testing environment, which allows for the development of enhanced knowledge documentation. Secondly, knowledge acquisition or elicitation received little attention in the surveyed papers. This has implications for software testing, especially for software companies that base their products on the OSS code or other external sources. These companies need to be more active in knowledge acquisition or elicitation since extensive knowledge is available in OSS communities (also testing experience or competence). Thirdly, knowledge dissemination (especially outside the testing teams) received little attention. However, the authors believe this aspect will be dominant in the successful testing of software products that are greatly based on open source software or external sources. For example, efficient testing knowledge dissemination with other companies involved in OSS communities can help to reduce testing costs and efforts as the communities can take over large parts of testing responsibilities. Fourthly, not much e attention was devoted to understanding how testing knowledge was created, especially tacit knowledge. Since many software companies work in Agile-inspired environments, it is believed that focusing on tacit knowledge management remains critical here. Fifthly, most of the papers [P1, P2, P3, P6, P9 and P19] identified or discussed some testing aspects but failed to discuss their importance, or connected these aspects (e.g., test case and testing phase) to testing processes (e.g., test planning, test case design, test execution and test result analysis) [1]. It is postulated that researchers should adjust the focus of research endeavors and introduce some of these aspects into exploring KM for software testing.

Looking at the **importance of additional testing techniques and types (RQ1 and RQ2)**, a possible implication from these results is that the suggested techniques and types are seldom validated. Moreover, it remains greatly unclear which testing methods to use in each of the software testing activities. Therefore, researchers should focus more on creating operational guidelines regarding which testing methods to use for which activities. Next, regression testing and GUI testing are considered to gain strong benefits from using the ontologies or KM models. More research needs to be conducted to provide similar analysis and clearly identify what testing techniques require what type of knowledge and how much these testing techniques are sensitive to, e.g., eliciting or creating tacit knowledge. Most of the studies have not specified and have not focused on the knowledge relevant for a specific testing technique. The taxonomy that summarizes the types of knowledge that support various testing techniques and their types is what is clearly missing in the current literature.
Focusing on tacit knowledge remains important since no study has focused on identifying the

importance of tacit knowledge management for software testing. It is also important to explore the importance of tacit knowledge and how to identify it, capture it and store it. In addition, there is a need to explore testing aspects as well as determine what testing types are dependent on efficient tacit knowledge management.

Identify mitigation strategies for the identified challenges (RQ3), there is a need to identify the mitigation strategies concerning each of the identified challenges and provide tools, recommendations, and techniques to overcome those challenges. The most distinct challenges are associated with knowledge reuse, knowledge transfer or knowledge sharing (CH1, CH2, CH3), and they clearly show that more research focus should be given to these areas. From the point of view of software companies, these areas will become dominant in the next years as more software is co-created in open source software communities or externally acquired from external software organizations. Moreover, insufficient knowledge sharing or transfer often results in losing the knowledge that is critical and therefore substantial additional costs are borne when restoring this knowledge. Thus, it is postulated that researchers in KM and testing should broaden their focus areas and expand the technical aspects by adding human aspects, knowledge reuse topics as well as organizational aspects that lead to increased knowledge sharing.

8. Validity threats

Validity threats under the snowballing phase of the thesis are discussed according to the four validity categories suggested by Wohlin et al. [60].

Internal validity threats are minimized by creating and maintaining a review protocol which encompassed the details of the search string formulation and start set identification, inclusion and exclusion criteria used, the quality assessment being carried out, etc. The risk for judgment error was minimized by performing the independent evaluation of the two authors who later compared and discussed the results. Both authors worked closely together and discussed any questionable cases. Moreover, internal validity threats are mitigated by following the mapping guidelines provided by Petersen et al. [61] and quality assessment criteria as per the guidelines provided by Ivarsson and Gorschek [46]. Finally, there is still some risk that the studied positive testing outcomes are the result of other aspects than applying KM techniques. It is planned to explore this aspect in future work when these relationships are explored in detail.

Construct validity focuses on various potential confounding factors regardless of whether a study could capture the intended knowledge, i.e. to achieve the aims and objectives. One of the main concerns for this research is multiple definitions of KM. This threat was mitigated by adopting the well cited definition by Davenport [2]. As indicated by Kaner [62], construct validity depends on the question of "How does one recognize that they are measuring what they usually think they are measuring against?". The search string structure could be one of the construct validity threats in this study. Therefore, the search string was iteratively formulated with extensive discussions between the authors. Next, data extraction could also be the source of validity threats. To avoid these threats, supervisor's assistance was accepted and all updates at each step were sent for approval.

External validity considers the capability to generalize results outside the studied context. Most of the studies fall under the case study research category with high rigor and relevance scores as most of them were conducted in industrial contexts. Thus, the outcomes can be considered industry pertinent and are more generalized. For the studies that received low rigor and relevance scores, it remains to be determined if the ideas suggested in these studies have high generalizability.

Reliability considers the degree of repeatability and whether the data and analysis depend on a specific researcher. To strengthen reliability, each step of the snowballing process was documented, including the database search. The same applies to each step of data collection and analysis and they can be backtracked, if needed. The quality assessment of the chosen papers was

ensured by using rigor and relevance criteria according to objective assessment criteria. The properties and aspects identified from the papers were mapped with the research questions to achieve the objectives of the study.

9. Conclusions

Software testing is knowledge-intensive and the use of KM practices and tools provides a wide range of benefits regarding the increase in capital and quality [1]. This paper focuses on the implementation of KM in software testing and on exploring the importance of KM in each of the software testing aspects and testing techniques. Also, the paper presents the challenges faced due to the lack of KM in software testing. The topic is explored in a systematic literature review.

Looking at the testing aspects identified in the study (RQ1), the results indicated that KM is mainly used to support the selection or execution of testing techniques (10 studies) or optimization of the testing processes (7 studies). At the same time, managing testing resources or knowledge about test cases or the test code has been greatly underrepresented. Knowledge elicitation, dissemination, acquisition, evolution and packaging receive little attention in the surveyed literature indicating that knowledge is mainly managed during software testing within a project or an organization and less attention is devoted to further knowledge sharing. Knowledge management system, models, representation, capturing and retrieval are the main KM areas that the surveyed literature focuses on.

Looking at the testing techniques that benefit from the application of KM practices (RQ2) the results indicate that ad hoc and exploratory testing gain more benefits from utilizing KM techniques than model-based testing techniques. This appears to be logical since model-based testing operates on highly formalized knowledge (models) where extensive reasoning can frequently be applied. Ad hoc or exploratory techniques rely more heavily on tacit knowledge and therefore demand more KM techniques.

This study identifies 21 challenges faced due to the lack of KM practices in software engineering (RQ3) and the most frequently mentioned challenges are associated with testing knowledge reuse, transfer, and sharing. Moreover, the risk of losing testing knowledge appears to be one of the prominent challenges. To summarize, this paper has made the following contributions:

- Exploring various testing aspects that are focused on while KM is applied in software testing literature. Moreover, the importance of each of the software testing aspect concerning KM was explored.
- Discovering that each of the testing aspects is focused on while KM is applied, albeit few of them are very important in the KM context.
- Determining the importance of each of the software testing techniques (i.e. design, execution and result analysis techniques) in the KM context along with obtaining the knowledge which is required for each technique so as to provide recommendations to store the tacit knowledge just in case any technique turns out to be important in the context of KM and utilize tacit knowledge.
- Uncovering various challenges that are faced due to the lack of KM in software testing literature

In future work, the authors plan to conduct case studies and investigate how KM is utilized during software testing by software-intensive organizations. There are also plans to explore the enabling factors that allow achieving good testing coverage without KM techniques. It is planned to study what modeling framework and models can support software testing tacit knowledge capture, analysis, storing and reuse. Finally, tacit knowledge management in software testing will also become the focus of further studies.

Acknowledgments

This work is supported by the IKNOWDM project from the Knowledge Foundation in Sweden (20150033).

References

[1] E.F. de Souza, R. de Almeida Falbo, and N.L. Vijaykumar, "Knowledge management initiatives in software testing: A mapping study," *Information and Software Technology*, Vol. 57, 2015, pp. 378 – 391. [Online]. http://www.sciencedirect.com/science/article/pii/S0950584914001335

[2] T.H. Davenport and L. Prusak, *Working knowledge: How organizations manage what they know.* Harvard Business Press, 1998.

[3] J. Andrade, J. Ares, M.A. Martínez, J. Pazos, S. Rodríguez, J. Romera, and S. Suárez, "An architectural model for software testing lesson learned systems," *Information and Software Technology*, Vol. 55, No. 1, 2013, pp. 18–34.

[4] 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*, Vol. 2. IEEE, 2009, pp. 90–94.

[5] R. Abdullah, Z.D. Eri, and A.M. Talib, "A model of knowledge management system in managing knowledge of software testing environment," in *5th Malaysian Conference in Software Engineering (MySEC)*. IEEE, 2011, pp. 229–233.

[6] A. Desai and S. Shah, "Knowledge management and software testing," in *Proceedings of the International Conference & Workshop on Emerging Trends in Technology*. ACM, 2011, pp. 767–770.

[7] K. Nogeste and D.H. Walker, "Using knowledge management to revise software-testing processes," *Journal of Workplace Learning*, Vol. 18, No. 1, 2006, pp. 6–27.

[8] L. Xu-Xiang and W.N. Zhang, "The PDCA-based software testing improvement framework," in *International Conference on Apperceiving Computing and Intelligence Analysis (ICACIA*. IEEE, 2010, pp. 490–494.

[9] X. Li and W. Zhang, "Ontology-based testing platform for reusing," in *Sixth International Conference on Internet Computing for Science and Engineering (ICICSE)*. IEEE, 2012, pp. 86–89.

[10] J. Kajihara, G. Amamiya, and T. Saya, "Learning from bugs (software quality control)," *IEEE Software*, Vol. 10, No. 5, 1993, pp. 46–54.

[11] C. O'Dell and C. Jackson Grayson Jr, "Knowledge transfer: discover your value proposition," *Strategy & Leadership*, Vol. 27, No. 2, 1999, pp. 10–15.

[12] I. Nonaka and H. Takeuchi, *The knowledge-creating company: How Japanese companies create the dynamics of innovation*. Oxford University Press, 1995.

[13] A.D. Marwick, "Knowledge management technology," *IBM Systems Journal*, Vol. 40, No. 4, 2001, pp. 814–830.

[14] O.K. Wei and T.M. Ying, "Knowledge management approach in mobile software system testing," in *IEEE International Conference on Industrial Engineering and Engineering Management*. IEEE, 2007, pp. 2120–2123.

[15] T. Abdou and P. Kamthan, "A knowledge management approach for testing open source software systems," in *International Performance Computing and Communications Conference (IPCCC)*. IEEE, 2014, pp. 1–2.

[16] J. Sirathienchai, P. Sophatsathit, and D. Dechawatanapaisal, "Simulation-based evaluation for the impact of personnel capability on software testing performance," *Journal of Software Engineering and Applications*, Vol. 5, No. 08, 2012, p. 545.

[17] V.H. Nasser, W. Du, and D. MacIsaac, "An ontology-based software test generation framework." in *The 22nd International Conference on Software Engineering and Knowledge Engineering, SEKE*, 2010, pp. 192–197.

[18] H. Li, F. Chen, H. Yang, H. Guo, W.C.C. Chu, and Y. Yang, "An ontology-based approach for gui testing," in *33rd Annual IEEE International Computer Software and Applications Conference*, Vol. 1. IEEE, 2009, pp. 632–633.

[19] E.F. Barbosa, E.Y. Nakagawa, and J.C. Maldonado, "Towards the establishment of an ontology of software testing." in *International Conference on Software Engineering & Knowledge Engineering*, 2006, pp. 522–525.

[20] N. Juristo, A.M. Moreno, and S. Vegas, "Reviewing 25 years of testing technique experiments," *Empirical Software Engineering*, Vol. 9, No. 1, 2004, pp. 7–44.

[21] A.C.C. Natali, A.R.C. da Rocha, G.H. Travassos, and P.G. Mian, "Integrating verification and validation techniques knowledge into software engineering environments," *Proceedings of 4as Jornadas Ibeoamericanas de Ingeniería del Software e Ingeniería del Conocimiento, JIISIC*, Vol. 4, 2004, pp. 419–430.

[22] N. Juristo, A.M. Moreno, and S. Vegas, "Towards building a solid empirical body of knowledge in testing techniques," *ACM SIGSOFT Software Engineering Notes*, Vol. 29, No. 5, 2004, pp. 1–4.

[23] R.L. Glass, R. Collard, A. Bertolino, J. Bach, and C. Kaner, "Software testing and industry needs," *IEEE Software*, Vol. 23, No. 4, 2006, pp. 55–57.

[24] R. Jain and S. Richardson, "Knowledge partitioning and knowledge transfer mechanisms in software testing: An empirical investigation," in *Proceedings of the 1st Workshop on Advances and Innovations in Systems Testing*, 2007.

[25] S. Vegas, "Identifying the relevant information for software testing technique selection," in *International Symposium on Empirical Software Engineering, ISESE*. IEEE, 2004, pp. 39–48.

[26] X. Liu, G. Gu, L. Yongpu, and W. Ji, "Research and application of knowledge management model oriented software testing process," in *11th Joint International Conference on Information Sciences*. Atlantis Press, 2008.

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

[28] J. Itkonen, M.V. Mäntylä, and C. Lassenius, "The role of the tester's knowledge in exploratory software testing," *IEEE Transactions on Software Engineering*, Vol. 39, No. 5, 2013, pp. 707–724.

[29] O. Taipale, K. Karhu, and K. Smolander, "Observing software testing practice from the viewpoint of organizations and knowledge management," in *First International Symposium on Empirical Software Engineering and Measurement*. IEEE, 2007, pp. 21–30.

[30] L. Xue-Mei, G. Guochang, L. Yong-Po, and W. Ji, "Research and implementation of knowledge management methods in software testing process," in *WRI World Congress on Computer Science and Information Engineering*, Vol. 7. IEEE, 2009, pp. 739–743.

[31] B.A. Kitchenham, T. Dyba, and M. Jorgensen, "Evidence-based software engineering," in *Proceedings of the 26th International Conference on Software Engineering*. IEEE Computer Society, 2004, pp. 273–281.

[32] J. Webster and R.T. Watson, "Analyzing the past to prepare for the future: Writing a literature review," *MIS quarterly*, 2002, pp. xiii–xxiii.

[33] W. Hayes, "Research synthesis in software engineering: a case for meta-analysis," in *Sixth International Software Metrics Symposium*. IEEE, 1999, pp. 143–151.

[34] C. Wohlin, "Guidelines for snowballing in systematic literature studies and a replication in software engineering," in *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*, ser. EASE '14. New York, NY, USA: ACM, 2014, pp. 38:1–38:10. [Online]. http://doi.acm.org/10.1145/2601248.2601268

[35] B. Kitchenham, O.P. Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, "Systematic literature reviews in software engineering – a systematic literature review," *Information and software technology*, Vol. 51, No. 1, 2009, pp. 7–15.

[36] B. Kitchenham, R. Pretorius, D. Budgen, O.P. Brereton, M. Turner, M. Niazi, and S. Linkman, "Systematic literature reviews in software engineering – a tertiary study," *Information and Software Technology*, Vol. 52, No. 8, 2010, pp. 792–805.

[37] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," Keele University & University of Durham, EBSE Technical Report EBSE 2007-01, 2007.

[38] C.W. Knisely and K.I. Knisely, *Engineering communication*. Cengage Learning, 2014.

[39] D.S. Cruzes and T. Dybå, "Research synthesis in software engineering: A tertiary study," *Information and Software Technology*, Vol. 53, No. 5, 2011, pp. 440–455.

[40] C. Goulding, *Grounded theory: A practical guide for management, business and market researchers*. Sage, 2002.

[41] R. Hoda, J. Noble, and S. Marshall, "Using grounded theory to study the human aspects of software engineering," in *Human Aspects of Software Engineering*. ACM, 2010, p. 5.

[42] B.G. Glaser, A.L. Strauss, and E. Strutzel, "The discovery of grounded theory; strategies for qualitative research." *Nursing research*, Vol. 17, No. 4, 1968, p. 364.

[43] M. Dixon-Woods, S. Agarwal, D. Jones, B. Young, and A. Sutton, "Synthesising qualitative and quantitative evidence: a review of possible methods," *Journal of Health Services Research & Policy*, Vol. 10, No. 1, 2005, pp. 45–53.

[44] D. Koznov, V. Malinov, E. Sokhransky, and M. Novikova, "A knowledge management approach for industrial model-based testing," in *Proceedings of the International Conference on Knowledge Management and Information Sharing*, 2009, pp. 200–205.

[45] M. Rodgers, A. Sowden, M. Petticrew, L. Arai, H. Roberts, N. Britten, and J. Popay, "Testing methodological guidance on the conduct of narrative synthesis in systematic reviews: effectiveness of interventions to promote smoke alarm ownership and function," *Evaluation*, Vol. 15, No. 1, 2009, pp. 49–73.

[46] M. Ivarsson and T. Gorschek, "A method for evaluating rigor and industrial relevance of tech-

nology evaluations," *Empirical Software Engineering*, Vol. 16, No. 3, 2011, pp. 365–395.

[47] A. Jonsson and G. Svingby, "The use of scoring rubrics: Reliability, validity and educational consequences," *Educational Research Review*, Vol. 2, No. 2, 2007, pp. 130–144.

[48] B. Moskal, K. Miller, and L. King, "Grading essays in computer ethics: rubrics considered helpful," *ACM SIGCSE Bulletin*, Vol. 34, No. 1, 2002, pp. 101–105.

[49] A. Vickers, "Ensuring scientific rigour in literature review," *Acupuncture in Medicine*, Vol. 13, No. 2, 1995, pp. 93–96.

[50] P. Runeson, M. Host, A. Rainer, and B. Regnell, *Case study research in software engineering: Guidelines and examples*. John Wiley & Sons, 2012.

[51] R. Wieringa, N. Maiden, N. Mead, and C. Rolland, "Requirements engineering paper classification and evaluation criteria: A proposal and a discussion," *Requirements Engineering*, Vol. 11, No. 1, 2006, pp. 102–107.

[52] P. Goodman Jon and C. Huseman Richard, *Leading with Knowledge: The Nature of Competition in the 21st Century*. Sage, London, 1999.

[53] L. Rajiv and M. Sarvary, "KM and competition in the consulting industry," 1999, p. 485.

[54] R. de Almeida Falbo, "Experiences in using a method for building domain ontologies," in *The 16th International Conference on Software Engineering and Knowledge Engineering, SEKE*, 2004, pp. 474–477.

[55] S. Vegas and V. Basili, "A characterisation schema for software testing techniques," *Empirical Software Engineering*, Vol. 10, No. 4, 2005, pp. 437–466.

[56] A. Abran, P. Bourque, R. Dupuis, and J.W. Moore, *Guide to the software engineering body of knowledge – SWEBOK*. IEEE Press, 2001.

[57] M. Cataldo, P.A. Wagstrom, J.D. Herbsleb, and K.M. Carley, "Identification of coordination requirements: implications for the design of collaboration and awareness tools," in *Proceedings of the 2006 20th Anniversary Conference on Computer Supported Cooperative Work*. ACM, 2006, pp. 353–362.

[58] D. Graham, E. Van Veenendaal, and I. Evans, *Foundations of software testing: ISTQB certification*. Cengage Learning EMEA, 2008.

[59] S. Vegas, N. Juristo, and V.R. Basili, "A process for identifying relevant information for a repository: A case study for testing techniques," in *Managing Software Engineering Knowledge*. Springer, 2003, pp. 199–230.

[60] C. Wohlin, P. Runeson, M. Höst, M.C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in software engineering*. Springer Science & Business Media, 2012.

[61] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, "Systematic mapping studies in software engineering." in *EASE*, Vol. 8, 2008, pp. 68–77.

[62] C. Kaner and W.P. Bond, "Software engineering metrics: What do they measure and how do we know?" in *In METRICS 2004. IEEE CS*. Citeseer, 2004.

[63] E.F. de Souza, R. de Almeida Falbo, and N.L. 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.

[64] A. Tinkham and C. Kaner, "Learning styles and exploratory testing," in *Proceedings of the Pacific Northwest Software Quality Conference*, 2003.

[65] S. Vegas, N. Juristo, and V. Basili, "Packaging experiences for improving testing technique selection," *Journal of Systems and Software*, Vol. 79, No. 11, 2006, pp. 1606–1618.

[66] J. Itkonen, M.V. Mantyla, and C. Lassenius, "How do testers do it? An exploratory study on manual testing practices," in *Proceedings of the 3rd International Symposium on Empirical Software Engineering and Measurement*. IEEE Computer Society, 2009, pp. 494–497.

[67] A. Freitas and R. Vieira, "An ontology for guiding performance testing," in *Proceedings of the 2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, Vol. 1. IEEE Computer Society, 2014, pp. 400–407.

[68] T.E. Lee, "Applying knowledge management approach for software testing," in *Advances and Innovations in Systems Testing*, 2007.

[69] E.F. de Souza, R. de Almeida Falbo, and N.L. Vijaykumar, "Ontologies in software testing: a systematic literature review," in *VI Seminar on Ontology Research in Brazil*, 2013, p. 71.

[70] C. Kerkhof, J. van den Ende, and I. Bogenrieder, "Knowledge management in the professional organization: a model with application to CMG software testing," *Knowledge and Process Management*, Vol. 10, No. 2, 2003, pp. 77–84.

[71] S. Vegas, N. Juristo, and V.R. Basili, "Maturing software engineering knowledge through classifications: A case study on unit testing techniques," *IEEE Transactions on Software Engineering*, Vol. 35, No. 4, 2009, pp. 551–565.

[72] R. Gentry and F. Shirazi, "A knowledge management analysis of an in-house manual software testing," *International Journal of Computer Application*, Vol. 1, No. 5, 2015, pp. 13–37.

[73] E.F. de Souza, R. de Almeida Falbo, and N.L. Vijaykumar, "Using ontology patterns for building a reference software testing ontology," in *17th IEEE International Enterprise Distributed Object Computing Conference Workshops (EDOCW)*. IEEE, 2013, pp. 21–30.

[74] V.H. Nasser, W. Du, and D. MacIsaac, "Knowledge-based software test generation." in *The 21st International Conference on Software Engineering and Knowledge Engineering, SEKE*, 2009, pp. 312–317.

[75] H. Li, H. Guo, F. Chen, H. Yang, and Y. Yang, "Using ontology to generate test cases for GUI testing," *International Journal of Computer Applications in Technology*, Vol. 42, No. 2-3, 2011, pp. 213–224.

Appendix A. Publication venue for the selected papers

Table A. Publication venue for the selected papers

Publication	Type	ID
Malaysian Conference in Software Engineering (MySEC)	Conference	P1 [5]
Information and Software Technology	Journal	P2 [3]
International Conference on Information Technology and Computer Science	Conference	P3 [4]
International Conference and Workshop on Emerging Trends in Technology (ICWET)	Conference	P4 [6]
International Conference on Industrial Engineering and Engineering Management	Conference	P5 [14]
Information and Software Technology	Journal	P6 [1]
International Symposium on Empirical Software Engineering and Measurement	Conference	P7 [29]
International Conference on Internet Computing for Science and Engineering (ICICSE)	Conference	P8 [9]
WRI World Congress on Computer Science and Information Engineering	Conference	P9 [30]
Euromicro Conference on Software Engineering and Advanced Applications	Conference	P10 [27]
IEEE Transactions on Software Engineering	Journal	P11 [28]
Journal of Workplace Learning	Journal	P12 [7]
International Conference on Knowledge Management and Information Sharing	Conference	P13 [44]
Empirical Software Engineering	Journal	P14 [55]
International Symposium on Empirical Software Engineering, ISESE	Conference	P15 [25]
International Conference on Software Engineering and Knowledge Engineering	Conference	P16 [63]
Pacific Northwest Software Quality Conference	Conference	P17 [64]
Journal of Systems and Software	Journal	P18 [65]
Joint International Conference on Information Sciences	Conference	P19 [26]
International Symposium on Empirical Software Engineering and Measurement	Conference	P20 [66]
International Performance Computing and Communications Conference (IPCCC)	Conference	P21 [15]
International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)	Conference	P22 [67]
Information Systems Research	Conference	P23 [68]
Ontology Research Journal	Journal	P24 [69]
Journal of Software Engineering and Applications	Journal	P25 [16]
International Conference on Software Engineering and Knowledge Engineering	Conference	P26 [17]
Knowledge and Process Management	Journal	P27 [70]
Empirical Software Engineering	Journal	P28 [20]
International Conference on Software Engineering and Knowledge Engineering	Conference	P29 [19]
Software Engineering Journal	Journal	P30 [71]
International Journal of Computer Application	Journal	P31 [72]
International Enterprise Distributed Object Computing Conference Workshops (EDOCW)	Conference	P32 [73]
International Conference on Software Engineering and Knowledge Engineering	Conference	P33 [74]
International Journal of Computer Applications in Technology	Journal	P34 [75]
Computer Software and Applications Conference (COMPSAC)	Conference	P35 [18]

Appendix B. Quality assessment based on rigor and relevance

Table B. Quality assessment based on rigor and relevance

Paper	Context	Study design	Validity	Rigor sum	Subjects	Scale	Research method-ology	Context	Relevance sum
P1	1	1	0	2	1	1	1	1	4
P2	1	1	1	3	1	1	1	1	4
P3	1	0.5	0	1.5	1	1	1	1	4
P4	1	0.5	0	1.5	1	1	1	1	4
P5	1	1	0	2	1	1	1	1	4
P7	1	1	1	3	1	1	1	1	4
P8	1	0.5	0	1.5	1	1	1	1	4
P9	1	0.5	0	1.5	1	1	1	1	4
P10	1	1	0	2	1	1	1	1	4
P11	1	1	1	3	1	1	1	1	4
P12	1	0.5	0	1.5	1	1	1	1	4
P13	0.5	0.5	0	1	1	1	1	1	4
P14	1	1	0	2	1	1	1	1	4
P15	1	1	0	2	1	1	1	1	4
P17	0	0.5	0	0.5	0	1	0	0	1
P18	1	0.5	0	1.5	0.5	1	0	1	2.5
P19	0.5	0.5	0	1	1	1	1	1	4
P20	1	1	0	2	1	1	1	1	4
P21	0	0.5	0	0.5	0	1	1	0	2
P22	0	0.5	0	0.5	0	1	1	0	2
P23	0.5	0	0	0.5	0	1	1	0	2
P25	1	1	0	2	1	1	1	1	4
P26	0	0.5	0	0.5	0	1	1	0	2
P27	0.5	0.5	0	1	1	1	1	1	4
P28	0.5	0.5	0	1	1	0.5	0.5	1	3
P29	0	0.5	0	0.5	1	1	1	1	4
P30	1	1	0	2	1	1	1	1	4
P31	1	1	1	3	1	1	1	1	4
P32	0	0.5	0	0.5	0	1	1	0	2
P33	1	0.5	0	1.5	1	1	1	1	4
P34	1	0.5	0	1.5	1	1	1	1	4
P35	1	1	0	2	1	1	1	1	4