Help! I need somebody. A Mapping Study about Expert Identification in Software Development

Carlos Eduardo Correa Braga
carlos.braga@aluno.ufes.br
Ontology & Conceptual Modeling Research Group (NEMO), Computer Science Department
Federal University of Espírito Santo
Vitoria, ES, Brazil

Paulo Sérgio dos Santos Júnior
paulo.junior@ifes.edu.br
LEDS Research, Department of Informatics
Federal Institute of Education, Science, and Technology of Espírito Santo
Serra, ES, Brazil

Monalessa P. Barcellos
monalessa@inf.ufes.br
Ontology & Conceptual Modeling Research Group (NEMO), Computer Science Department
Federal University of Espírito Santo
Vitoria, ES, Brazil

ABSTRACT

Context: Software development is a knowledge-intensive activity, and its success in an organization relies deeply on knowledge sharing. Knowledge management challenges are often increased in agile environments, which involve a lot of tacit knowledge, commonly acquired through experiences and hard to be made explicit. Therefore, knowledge sharing among practitioners is crucial. However, identifying suitable experts to share specific knowledge is not trivial. It involves not only discovering the individuals with the desired knowledge but also considering other factors that may improve the expert responsiveness, such as social connections and availability. *Objective*: Considering the important role experts play in knowledge sharing, we decided to investigate approaches that help identify experts that can share knowledge in software development. Our goal is to provide a panorama of the existing approaches and shine a light on research opportunities. *Method*: We carried out a systematic literature mapping and analyzed 17 publications. *Results*: The results show that most approaches have relied on code repositories as a source of evidence for identifying experts and, consequently, focus on supporting developers and aiding in the codification activity. Additionally, expert identification has been mostly automated, and factors beyond possessing the desired knowledge have often been disregarded. *Conclusion*: Although there are several expert identification approaches, there has been a lack of concern with factors that influence reaching the most suitable expert for a specific situation (e.g., considering the characteristics of the person seeking knowledge). Moreover, there is a need for deeper reflection on how to better explore different artifacts as sources of expert evidence and how to combine them to improve expert identification.

CCS CONCEPTS

• **Software and its engineering → Software development process management**; • **Information systems → Information retrieval**.

SBES 2023, September 25–29, 2023, Campo Grande, Brazil

ACM ISBN 979-8-4007-0787-2/23/09…$15.00
https://doi.org/10.1145/3613372.3613389

KEYWORDS

Expert Identification, Knowledge Sharing, Mapping Study

ACM Reference Format:
Carlos Eduardo Correa Braga, Paulo Sérgio dos Santos Júnior, and Monalessa P. Barcellos. 2023. Help! I need somebody. A Mapping Study about Expert Identification in Software Development. In *XXXVII Brazilian Symposium on Software Engineering (SBES 2023), September 25–29, 2023, Campo Grande, Brazil.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3613372.3613389

1 INTRODUCTION

Knowledge is a human specialty stored in people's minds, acquired through experience and interaction with their environment [42]. Historically, organizations' knowledge has been often undocumented, being represented through the skills, experience, and knowledge of its professionals, typically tacit knowledge [40], which makes its use and access limited and difficult [36].

Software development is a complex and knowledge-intensive activity. Agile software development methods, with their focus on tacit knowledge, have increased the importance of knowledge resources and knowledge sharing. Sharing experiences and knowledge can enhance teams' ability to handle uncertain and ambiguous situations, which is essential for the success of software projects [46]. Considering the importance of knowledge sharing and that there is a lot of knowledge available only in the minds of individuals, identifying who possesses the desired knowledge is crucial.

The individual who possesses knowledge and displays expertise in a specific domain is known as an *expert*. There are several definitions for expert in the literature, such as an individual who has skills in his domain of expertise [50]; an individual who has a great deal of knowledge and displays comprehensive skills in a specific area [23]; and a person who has the best knowledge and experience in particular topic [32]. In this work, we consider that experts are people who have knowledge of a subject and may have different levels of expertise. Moreover, along the text we use the term *knowledge* in a broad way, considering it a mix of theoretical knowledge, expertise, skills, and other assets that are often stored in people's mind and are therefore difficult to codify [10][17].

Identifying experts in software organizations is a challenging task, especially when it comes to the context of large-scale distributed software projects [25, 26, 43]. Individuals can seek out a particular knowledge by asking people and following a trail of referrals until they locate the appropriate expert [8]. This approach can

incur significant costs, such as effort repeated by different people looking for the same answers, and miscommunication that leads to taking the advice of not-so-expert experts who happen to be found quickly [8]. Moreover, knowledge possession is not the only factor that influences expert responsiveness. Other factors are also important, such as time availability [29], communication skills, and interest in building reputation [49].

Considering the important role that experts play in software development and also motivated by the difficulties faced by two of the authors when looking for experts that could share knowledge of some topics, we decided to investigate approaches that help identify experts to share knowledge in the software development context. Therefore, we performed the mapping study presented in this paper. A mapping study is a secondary study designed to give an overview of a research area through classification and counting contributions in relation to the categories of that classification. It makes a broad study on a topic of a specific theme and aims to identify available evidence about that topic [38]. The panorama provided by a mapping study allows identifying issues in the research topic that could be addressed in future research.

Before deciding to carry out our study, we searched the literature looking for secondary studies investigating expert identification. The one closest related to ours is the work by Husain et al. [17], which investigated expert finding systems in general (i.e., regardless the application domain). Other works, such as [23] and [6], studied some expert identification methods by analyzing their underlying algorithms and models (without adopting rigorous procedures of systematic reviews to conduct publications search, selection, and data extraction). Although the aforementioned studies address the expert identification subject, none of them regards the software development domain and, thus, do not address specific questions of this context, such as software development activities and roles supported by the approaches. Moreover, we are interested in works presenting expert identification approaches and that to some degree are concerned with knowledge sharing.

In our study, we analyzed 17 different expert identification approaches proposed to the software development domain. The results revealed a predominance of automated approaches that use code repositories as a source to identify experts aiming at helping developers with implementation issues. Although there have been several approaches, factors beyond the knowledge possession itself have not been a concern to reach the most suitable expert for a specific situation. For example, the characteristics or context of the person interested in the expert's knowledge (i.e., the seeker) have not been considered in most approaches. Moreover, there is a need for deeper reflection to make the most of different artifacts that can be sources of expert evidence.

This paper is organized as follows: Section 2 provides a brief background for the paper; Section 3 presents the research protocol used in our study; Section 4 summarizes the obtained results; Section 5 discusses the results; Section 6 regards the limitations of the study; and Section 7 presents our final considerations.

2 BACKGROUND

Knowledge is a mix of experience, values, contextual information, and expert insight that provides a framework for evaluating and incorporating new experiences and information [10]. It can be explicit or tacit. Explicit knowledge is structured and formal [44]. Consequently, it can be easily communicated and shared. On the other hand, tacit knowledge is hard to formalize and communicate to others since it is highly personal [34]. The interactions between individuals enable the creation of new knowledge through a continuous conversion between tacit and explicit knowledge [34].

In order to effectively handle knowledge in an organization, a knowledge management (KM) approach is necessary. KM is the process of capturing, distributing, and effectively using knowledge to achieve organizational goals. It involves creating an infrastructure and culture that supports the creation, sharing, and use of knowledge by individuals [2]. In particular, knowledge sharing is the process of transferring knowledge between individuals, groups, or organizations [2]. Codification and personalization approaches can support knowledge sharing. The former focuses on explicit knowledge, aiming to systematize and store knowledge. The latter, in contrast, focuses on tacit knowledge and involves creating a repository of information on knowledge sources (e.g., yellow pages) to facilitate knowledge sharing within the organization [14]. As the name suggests, personalization approaches rely on personal interaction. They require identifying who has specific knowledge and promoting knowledge sharing between the person who possesses the desired knowledge (the expert) and the person interested in that particular knowledge (the seeker).

Although nowadays there is a huge volume of data and information available, people often need to consult an expert to determine ways to solve their problems, i.e., people still seek the knowledge and guidance of an expert, and require comprehensive information regarding experts who can answer their questions [17]. This kind of knowledge sharing depends on finding a suitable expert, i.e., who possesses the desired knowledge and is willing to share it. Finding experts is an important task and has attracted much attention [23].

In software development, developers often rely on past experiences to recall similar tasks and identify the most knowledgeable individual for help. However, as the amount of tasks and artifacts increases, manually browsing this history becomes challenging. Therefore, there is a need for efficient methods to leverage knowledge and experience and support knowledge sharing in software development contexts [24].

The data overload problem and the specific nature of the experts' knowledge can hinder people from finding experts with the knowledge they require. Expert finding systems have been proposed to address this issue. They are information retrieval systems that identify candidate experts and rank them based on their expertise in a given subject [6, 17]. In these systems, expertise is extracted from evidence recorded in sources such as publications, reports, projects, homepages, social networks [5], and source code repositories[4, 37].

3 RESEARCH PROTOCOL

Given the challenges involved in knowledge sharing, particularly regarding the identification of suitable experts to share specific knowledge, we decided to carry out a mapping study aiming to investigate how experts have been identified in the software development context. We followed the process defined in [20], which

includes three phases: *planning*, when the research protocol is defined with the purpose of supporting study replicability as well as helping researchers to avoid bias when conducting the study; *conducting*, when the protocol is executed and data are extracted, analyzed and recorded; and *reporting*, when the results are recorded and made available to potentially interested parties.

The study **goal** was to investigate approaches that help identify suitable experts to share knowledge in the software development context. For achieving this goal, we defined the **research questions** presented in Table 1.

The **search string** adopted in the study contains three parts joined with the operator AND. The first part includes terms related to expert identification joined with the operator OR to allow synonyms. The second and third parts aim to restrict the scope to knowledge sharing in the software development context. For establishing the string, we performed tests using different terms, logical connectors, and combinations among them, and we selected the string that provided better results in terms of the number of publications and their relevance. More restrictive strings (for example, some including "knowledge sharing" or "knowledge transfer" in the second part or "software development" in the third part) excluded important publications identified during the informal literature review that preceded the study and that were used as control publications. More comprehensive strings (e.g., those separating terms put together in the first part of the string) returned too many publications out of the scope of interest. We used two papers ([35, 45]) identified during the informal literature review that preceded the systematic mapping as control publications to help us define the string. The search string used in the study is the following: *("expert* identification" OR "who knows what" OR "expert* retriev*" OR "expert* find*" OR "expert* locat*" OR "expert* recommend*" OR "expert* discover*" OR "finding expert*") AND "knowledge" AND "software"*.

Scopus and Engineering Village digital libraries were used as **sources** of publications. Scopus is a major database of peer-reviewed literature that indexes papers from other sources like IEEE, ACM, and Science Direct. Engineering Village is also a citation indexing platform, which indexes complementary sources.

Publication selection was performed in five steps. (S1) *Preliminary Selection and Cataloging*: the search string was applied in the search mechanism of each digital library considering the title, abstract, and keywords. (S2) *Duplications Removal*: publications indexed in both databases were identified and duplications were removed. (S3) *Selection of Relevant Publications – 1st filter*: the abstracts were analyzed considering the following inclusion (IC) and exclusion (EC) criteria: (IC1) the publication addresses approach that helps identify experts to share knowledge in the software development context; (EC1) the publication is a secondary study, a tertiary study, an editorial, a summary, proceedings or a short paper. (S4) *Selection of Relevant Publications – 2nd filter:* the full text of the papers was analyzed considering IC1, EC1, and the following additional criteria: (EC2) the publication is not written in English; (EC3) the publication is an older version of a publication already considered; (EC4) it was not possible to have access to the full text of the publication. (S5) *Snowballing*: as suggested in [20], the references of publications selected in S4 were analyzed by applying the first (S3) and second (S4) filters and, the publications presenting results related to the research topic were included in the study. This step was repeated until no new publication was found.

We used the StArt tool[1] to support publication selection. To consolidate data and support **data extraction**, publications returned in the publication selection steps were cataloged and stored in spreadsheets. We defined an id for each publication and recorded the publication title, authors, year, and vehicle of publication. Data from publications returned in S4 were extracted and organized into a data extraction table oriented to the research questions.

The first author performed publication selection and data extraction. The second author reviewed both. Once data has been validated, the first and the second authors carried out **data interpretation and analysis**. Quantitative data was tabulated and used in graphs and statistical analysis. The third author reviewed the results. Discordances were discussed and resolved. Finally, the three authors performed qualitative analysis considering the findings, their relation to the research questions, and the study purpose.

4 DATA EXTRACTION AND SYNTHESIS

Searches were conducted for the last time in March 2023 and the study considered papers published until 2022. The followed process and the number of publications selected in each step are presented in Figure 1.

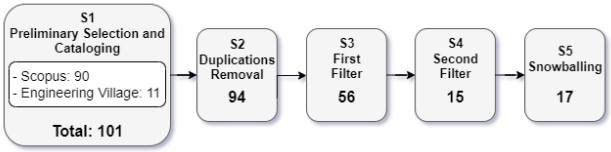

Figure 1: Publications selection

In the 1st step (S1), we performed a search in Scopus and Engineering Village using the defined search string. As a result, 101 papers were identified. In the 2nd step (S2), 7 duplicated papers were removed which represents a reduction of approximately 6.9% from the initial number of papers. In the 3rd step (S3), we applied criteria IC1 and EC1 considering the abstract. This reduced the number of papers to 56, representing a reduction of approximately 40.4% from the previous step. In the 4th step (S4), the selection criteria were applied considering the full text, resulting in the identification of 15 relevant papers – which represents a reduction of approximately 73.2% from the 3rd step, and a total reduction of approximately 85% from the initial quantity of papers. Finally, in the 5th step, we performed snowballing by checking the references of the 15 selected publications and identified two more publications, which in total added up to 17 publications. Detailed information about the selected publications, including a brief description and extracted data, can be found in our study package [7].

It is important to emphasize that this study focuses on publications presenting approaches that support expert identification to share knowledge in the software development context. Thus, publications presenting expert identification methods that not used on the software development or did not mention knowledge sharing

[1] https://bit.ly/3bW3Mo6

Table 1: Research questions and their rationale

ID	Research Questions	Rationale
RQ1	When and in which type of vehicle have the papers been published?	Provide an understanding of when and where (journal/conference/workshop) publications addressing expert identification in the software development context have been published, to analyze if the research topic has been addressed frequently and if it has been the target of mature or emergent research.
RQ2	What type of research has been done?	Investigate which type of research is reported in each selected paper, by considering the classification defined in [51]. This question, together with RQ1, helps evaluate the maturity of the research topic.
RQ3	Which artifacts have been used as sources for identifying experts?	Identify artifacts that have offered evidence that helps identify experts and investigate the comprehensiveness of the set of artifacts used with that purpose.
RQ4	Have experts been identified automatically?	Understand if there has been a concern with supporting expert identification automatically or if the approaches have relied on manual effort.
RQ5	Have the approaches been concerned with recommending the most suitable expert for a particular user/situation? If so, have they considered non-technical aspects?	Investigate if, besides identifying experts, the approaches have covered recommending the most suitable ones for a situation and verify if they have considered aspects such as proximity, seniority, availability, and willingness to answer, among others.
RQ6	Which roles (e.g., developer, project manager) have been supported by the approaches?	Identify which roles have been the seekers of experts in the proposed approaches (i.e., the roles that need knowledge shared by experts) and investigate if some roles have been predominant.
RQ7	Which processes/activities have been supported by the proposed approaches?	Verify if the approaches have been proposed to support the software life cycle as a whole or if they have focused on specific processes/activities (e.g., planning, implementation). In the last case, identify the processes/activities (taking [11] as a reference) that have been the focus of the approaches and verify if some of them have received more attention while others have not been considered.
RQ8	Have the approaches used any conceptual ground (e.g., knowledge theory, taxonomy, ontology)?	Investigate if the approaches have been grounded in any formal conceptualization and the purpose of using it.

were not captured, even if they could be applied in that context. For example, some works related to information retrieval address expertise identification from data extraction and present algorithms or models for this purpose. Works of this type were not the target of our study, because although they are related to expertise identification, they do not address expert identification for knowledge sharing in the software development context.

The selected publications are presented in Table 2. In the following, we present the data synthesis for each research question.

Publication year and type (RQ1): Figure 2 presents the number of publications over the years and their vehicle of publication. The main vehicle of publication has been conferences, with 10 publications (58.8%). Journals accounted for 4 publications (23.5%), and workshop accounted for 3 publications (17.7%).

Research type (RQ2): Table 3 shows the number of publications considering their research type according to the classification proposed in [51]. All the 17 selected publications propose a solution to a problem and argue its relevance, i.e., they are classified as *Proposal of Solution* research. 12 of them (70.6%) present some kind of evaluation. From these, six (35.3%) were evaluated in practice (i.e., also classified as *Evaluation Research*) and six (35.3%) evaluated characteristics of the solution not yet implemented in practical settings (i.e., *Validation Research*).

Sources for identifying experts (RQ3): We identified eight sources of evidence used to identify experts, as shown in Figure 3.

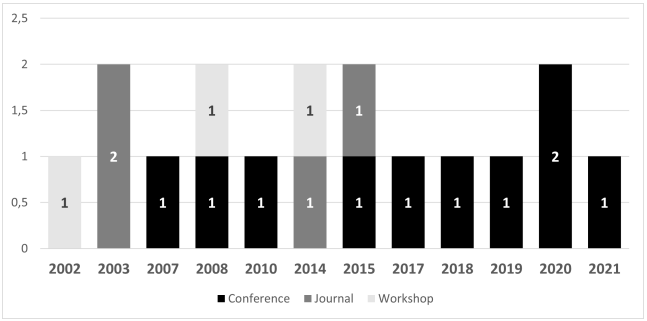

Figure 2: Publication year and vehicle

Code repository was used in 10 publications (58.8%), being the most used source. It was followed by electronic documents and mailing systems, used in four publications (23.5%) each. Ticket systems and chats were used in two publications (11.8%) each, and online forums, Question & Answering tools (e.g., Stack Overflow), and employee database were used in only one publication (5.9%) each. The sum of the values is greater than the number of investigated publications because some of them use more than one source for expert identification. Also, publication #5 does not inform the used source, and #13 uses a different approach (manual assessment).

Table 2: Selected publications

ID	Brief Description	Ref.
#1	Presents an approach based on a bot that identifies experts based on keywords (e.g., considering the number of times that a keyword is mentioned, the number of sent messages including the keywords).	[35]
#2	Presents an approach that uses data from Questions & Answers (Q&A) websites to profile users and recommends experts for answering a question.	[15]
#3	Uses knowledge models to help identity which developers (expert candidates) have more knowledge of specific elements of software projects.	[24]
#4	Proposes a collaborative network based on code review data retrieved from GitHub. To identify experts, it considers not only the specific knowledge but also expertise in collaboration on the required topics.	[41]
#5	Proposes a mobile application to support expert locations for global software development environments.	[9]
#6	Proposes an expert recommendation method based on knowledge embedding, which recommends appropriate experts for developers.	[13]
#7	Presents a framework for identifying experts by eliciting developers' expertise in software components using complexity analysis of source code.	[47]
#8	Presents an approach that combines natural language processing techniques, machine learning, statistical and search-based techniques to find experts in the open-source software context.	[31]
#9	Introduces the concepts of degree of knowledge (amount of expertise of candidate experts) and social relative importance (the social factor between the candidate experts and a query issuer) to recommend suitable experts.	[3]
#10	Introduces a degree-of-knowledge model that identifies experts by computing developer's knowledge based on developer's authorship and interaction data.	[12]
#11	Uses the developers' mailing list and source code history to support finding experts during coding.	[30]
#12	Presents an approach that identifies experts based on the interaction of individuals with software artifacts.	[19]
#13	Presents an approach that provides a list of experts to meet users requests. The list is based on manual assessment of friendship and skill information.	[45]
#14	Proposes an approach to build expert finder systems, supporting the adaptation of the search strategy according to the organization's needs.	[16]
#15	Presents an approach for identifying experts by using sensitive data and preserving privacy.	[1]
#16	Proposes a tool that allows identifying the most knowledgeable person related to a tool, language, and release version.	[28]
#17	Proposes an approach that uses social structures on GitHub (following-followed, watching, and collaboration networks) and programming behavior for identifying experts.	[27]

Table 3: Research Type

Research Type	Publications	Total	%
Proposal of Solution	#5, #7, #8, #14, #15	5	29.4%
Proposal of Solution & Evaluation Research	#1, #4, #9, #10, #12, #16	6	35.3%
Proposal of Solution & Validation Research	#2, #3, #6, #11, #13, #17	6	35.3%

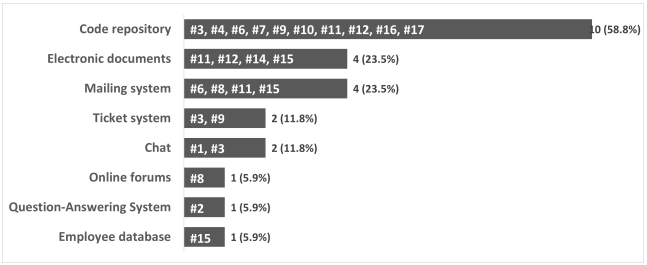

Figure 3: Sources for identifying experts

Automated Support (RQ4): There has been a predominance of publications that identify experts automatically. 15 approaches (88.2%) provide automated support. Only one approach (5.9%) requires that the individuals manually report their expertise and knowledge seekers input data evaluating expert candidates. One publication (5.9%) does not make clear whether there is automated support. Table 4 summarizes the publications considering the automated support aspect.

Use of non-technical aspects for recommending experts (RQ5): Most publications (12, i.e., 70.6%) are concerned with identifying experts and recommending the most suitable ones. However, only three of them (17.6%) consider non-technical aspects (e.g., previous collaboration and friendship relations) for recommending experts. Five publications (29.4%) identify experts but are not concerned with recommending the most suitable one for the seeker's needs. Table 5 summarizes these results.

Supported Roles (RQ6): Seven approaches (41.2%) aim at identifying experts to satisfy developers' needs (i.e., the developers are the expert seekers). Two approaches (11.8%) address managers'

Table 4: Automated support

Automated support	Publications	Total	%
Yes	#1, #2, #3, #4, #6, #7, #8, #9, #10, #11, #12, #14, #15, #16, #17	15	88.2%
No	#13	1	5.9%
Undefined	#5	1	5.9%

Table 5: Recommendation based on non-technical aspects

Use non-technical aspects	Publications	Total	%
Yes	#6, #9, #13	3	17.6%
No	#1, #2, #5, #7, #10, #11, #14, #15, #16	9	53%
Do not recommend	#3, #4, #8, #12, #17	5	29.4%

needs, while nine publications (52.9%) do not make explicit the supported roles. Some publications mention more than one supported role. Table 6 indicates the roles supported by each publication.

Table 6: Supported roles

Role	Publications	Total	%
Developer	#6, #8, #9, #10, #11, #14, #16	7	41.2%
Manager	#12, #16	2	11.8%
Undefined	#1, #2, #3, #4, #5, #7, #13, #15, #17	9	52.9%

Supported processes/activities (RQ7): Table 7 presents the processes/activities (based on [11]) supported by the approaches. Most publications (14, corresponding to 82.4% of the total number) are not focused on specific activities and can be used to support the software process according to the seeker's needs. The Implementation process is the target of two publications (11.8%). Human Resource Management is the focus of one (5.9%). One publication (5.9%) considers the use of experts to support Systems/Software Requirements Definition. The total value in Table 7 is higher than the number of investigated approaches because some of them are related to more than one activity.

Table 7: Supported processes/activities

Process/Activity	Publications	Total	%
Implementation	#10, #16	2	11.8%
Human Resource Management	#10	1	5.9%
Requirements Definition	#8	1	5.9%
General	#1, #2, #3, #4, #5, #6, #7, #9, #11, #12, #13, #14, #15, #17	14	82.4%

Use of conceptual ground (RQ8): Most approaches (14, corresponding to 82.4%) do not use any conceptual ground to help structure knowledge or help identify experts. Ontology is used in two approaches (11.8%) while taxonomy is considered in one approach (5.9%). Table 8 summarizes these results.

Table 8: Conceptual ground

Conceptual ground	Publications	Total	%
Ontology	#8, #14	2	11.8%
Taxonomy	#5	1	5.9%
None	#1, #2, #3, #4, #6, #7, #9, #10, #11, #12, #13, #15, #16, #17	14	82.4%

5 DATA ANALYSIS AND DISCUSSION

In this section, we provide additional information about the analyzed approaches and discuss the obtained results.

Looking at *when the studies were published* (RQ1), we can notice that the topic has been addressed for around 20 years but the number of publications per year is low (0.8 from 2002 to 2022) and there is no publication on the subject in some years (2004-2006, 2009, 2011-2013, 2016, 2022). Although research on this topic has been frequent in the last years, the big picture suggests that the topic can be further explored. Concerning the *research types* (RQ1), there is a low percentage of journal publications (23.5%), which generally require more mature works. Therefore, this fact can be seen as a reinforcement that research on this topic is not mature enough yet.

Results related to *research type* (RQ2) show that although around 70% of the works included some kind of evaluation, only half of these proposals were used in practical settings (around one third of all identified approaches). Several approaches were evaluated using actual data from repositories to identify experts, but their implications in real scenarios have not been explored yet. This can be a sign of difficulty in applying the proposed approaches in industry, which reinforces that research on this topic is not quite mature enough yet and there seems to be a gap between theory and practice that can be further explored.

Concerning the *sources used to provide expert evidence* (RQ3), code repositories have been the one most used. This is not a surprise, because every developer produces code, which is the main product in the software development context. Also, especially in the context of GIT repositories, additional information is provided through pull requests comments and commit messages, which contributes to obtaining information about the person who produced the code. However, using data from code repositories introduces a concern with the level of expertise. The existence of results produced by a developer (e.g., code related to a specific library) is not enough to state that that developer is an expert on the subject. The individual can indeed have knowledge of that activity or technology, but may not be an expert on that. Thus, using limited data to consider a developer an expert contributes to identifying not-so-expert experts and not meeting the seeker needs (or even propagating unsuitable knowledge) [8]. Therefore, additional data and rationales should be considered to reach an accurate expert identification.

Electronic documents and mailing systems have been the second most used sources of expert evidence. For example, in #14, a database of internal technical reports is used, and a component is responsible for establishing a link between documents and their authors, ranking them according to metrics such as document age. In #8, in turn, experts are identified by ranking the participants engaged in a set of discussions in mailing lists according to their intentions manifested by speech-acts. Some publications consider not only documents from office suite applications (e.g., Microsoft Word, LibreOffice Writer) as electronic documents, but also any kind of textual information in other formats (e.g., pages in wikis). Ticket systems have also been used. This kind of tool is commonly used for registering requirements (e.g., user stories) and bugs, and for tracking developers' activity towards these items – e.g., assignee developer, time spent, among other information. This information helps expert identification and is often deeply related to the coding activity (it is even possible to define explicit links between a user story and a commit, for example). Question-answering systems, chats, online forums and employee databases have also offered evidence for expert identification.

Although different types of artifacts have been explored, the predominance of code repositories suggests that the approaches have preferred the use of artifacts that stores data in a structured way and that contains a good amount of data. This certainly makes it easier to capture and analyze data providing evidence of people's expertise. However, other sources could be deeper explored. For example, due to the COVID-19 pandemic, organizations changed the way people work and communicate. There was an increasing use of tools such as mailing, chats (including message exchange applications), and shared digital boards, among others. Although the pandemic is over, some practices remain and will probably be kept in the organizations. Thus, there is an opportunity to leverage expert identification considering such artifacts. We also believe that other sources can be more explored, such as post-mortem documents, CI/CD logs, and incident management tools. For example, incident management tools and post-mortem documents gather valuable data about incidents: related infrastructure components and applications, applied workaround, root cause, definitive solution, and individuals who provided the workaround and the definitive solution, among other data. Then, it would be possible to relate/integrate these data and, when a critical incident occurs, recommend key individuals (experts) for sharing knowledge of how to mitigate its impact and provide a definitive solution. CI/CD logs, in turn, record data about builds and deploys (failed or successful), which could be used to identify individuals that usually fix breaking builds and recommend them as CI/CD experts (or even recommend an expert based on a specific error). Moreover, it is important to consider more than one source, so that they can complement each other and provide a more complete and accurate view of experts. This is the case of works such as #3 and #11. In this context, it is important to reflect on the weight of evidence provided by each source. Furthermore, there must be a concern with how to properly integrate evidence from different sources.

By analyzing *how the approaches have identified experts* (RQ4), we noticed that only one (#13) relies on knowledge manually reported by individuals and manual evaluation of experts. The approach authors argue that even though asking users to provide skill information is a sensitive issue, most people are willing to perform the extra work once they perceive they will benefit from expert identification. All the other works identify experts automatically. This result makes sense, considering that manual assessments are burdensome and knowledge is quite dynamic and may be impacted over time by many factors (e.g., forgetting and repetitions [21]). Even so, manual assessments can be useful to alleviate the cold start problem – e.g., when a recently hired individual has expertise in a particular subject due to previous experiences, but has not produced enough evidence in the organization context. External sources can help to address this issue as well.

Concerning *the use of non-technical aspects for recommending experts* (RQ5), first, we perceived that around one third of the approaches do not address expert recommendation. That means, although these approaches identify experts, they do not recommend, among the identified experts, which one would be more suitable for the seeker's needs. This is the case of approach #3, which uses the degree of interaction among developers, and between developers and artifacts, to measure developer knowledge and identify which ones have more knowledge of elements of software development projects. Only three out of the 12 approaches that recommend experts consider non-technical aspects. In #9, for example, a metric called Social Relative Importance is proposed to quantify the social distance between a knowledge seeker and an expert candidate aiming at identifying the most suitable expert based on that aspect. In #13, in turn, users rate each other using a scale between 0 (do not know this person) and 5 (especially close). Based on this aspect, it is possible that a close expert candidate with a lower expertise score is recommended over an unknown expert with a higher expertise score. The lack of approaches taking non-technical aspects into account suggests an important gap. Identifying or recommending experts considering only the knowledge and level of expertise they possess may not be enough to ensure effective knowledge sharing. Factors like lack of willingness to collaborate, limited time availability, lack of encouragement, and communication constraints can lead people to fail in knowledge-sharing [18] and, thus, should be considered when selecting experts to share knowledge.

As for the *roles supported by the approaches* (RQ6), there is a predominance of developers. Considering that code repositories have been the predominant source of expert evidence, this result was indeed expected. Developers can need other people's knowledge and guidance to complete tasks on a daily basis, especially when we consider the fast pace of technological evolution and the wide range of different languages, frameworks, and patterns used in software development. Experts can, thus, provide useful knowledge to help developers improve the quality of the developed results (i.e., improve software quality) as well as increase performance when producing them (i.e., process improvement). Managers were also mentioned. Expert identification can help them develop the team by boosting knowledge dissemination and identifying knowledge gaps. Expert identification can also help managers define teams and hire the most suitable candidate for a position. One could expect that publications using code repositories as a source of expert evidence would support developers. However, we noticed that this is not always the case. For example, approaches #12 and #16 use data from code repositories to support managers. Moreover, other sources

can also be used to identify experts and help developers (e.g., #8 uses online forums).

Many publications do not make explicit the roles supported for the proposed approach and, in these cases, we understand that they can be used to support several roles. We believe that this happens because such approaches do not refer to any specific knowledge or do not use sources of evidence addressing specific subjects. For example, if we consider electronic documents as the source of evidence, it is possible to extract knowledge related to different subjects, depending on the document content. For instance, if the document addresses requirements, it can provide information about experts in requirements and, thus, support requirement engineers. On the other hand, if the document concerns risk management, it can provide information about risk management experts and, therefore, help managers.

Regarding *processes/activities supported by approaches* (RQ7), most of the investigated approaches are not devoted to specific processes or activities. They do not limit the domain knowledge and, thus, aim to support the software process according to the seeker's needs. As a consequence, these approaches can be used regardless of the specific knowledge being sought. In this way, they can support a wide range of activities in software organizations. For example, approach #1 uses keywords to identify experts. Thus, it can be used to support different activities, depending on the knowledge the keywords refer to.

Some approaches focus on specific processes/activities. These approaches are important because they might better support processes/activities by considering their particularities, which can introduce the need for new requirements when identifying experts. For example, if one is interested in knowledge sharing to support requirements elicitation, the expert identification approach could consider roles related to the business domain (e.g., business stakeholders) in addition to the ones directly related to software development (e.g., requirements engineer, developer). Given that expert identification deals with satisfying knowledge needs and addresses discovering the link between desired knowledge and candidate expert, it was expected that some approaches would support Human Resource Management activities, such as identifying knowledge gaps, acquiring the desired knowledge, and developing the team. This is the case of the approach presented in #10, which consists of a case study involving a mentoring situation in which an experienced developer (expert) is identified to help a new team member get knowledge of and become familiar with the project source code.

One approach (#8) is devoted to Requirements Definition, which uses speech-acts combinations to discover knowledge related to requirements expressing feature requests and bugs. Also, two approaches focus on the Implementation process. When relating this result with the ones from RQ6, we noticed that *developers* appear more in RQ6 than *Implementation* in RQ7. This occurs because some publications do not limit the processes/activities supported and, as we explained before, they could be used to support several processes (including the implementation process).

Finally, concerning the *use of conceptual ground* (RQ8), more than 80%s of the approaches do not consider any conceptual foundation. Only a few are grounded in taxonomy or ontology. In #5, a taxonomy is used as a foundation to formalize software development and testing terms. In #8, in turn, a communication ontology is extended and used to support a more robust and faster design of expert finding systems. The small number of works committed with a formal conceptualization suggests a lack of concern with semantic issues, which can lead to conflicts whenever the same information item is given divergent interpretations, a situation that may not even be detected [48]. Neglecting these semantic conflicts can lead to solutions that fail in achieving their purposes [39]. Given that expert identification may involve different sources, artifacts, tools, and technologies, a shared conceptualization could support, for example, structuring knowledge, defining the solution conceptual architecture, and helping extract and integrate data from different sources [33]. In this way, expert identification approaches could benefit from the use of ontologies, taxonomies, and other conceptual grounds.

Based on the panorama provided by the study results, in summary, we can say that, in general, expert identification approaches that support knowledge sharing in the software development context have been automated and based mainly on data stored in source repositories (although other artifacts have also been used). Moreover, although the approaches have supported software development, they have not been devoted to specific roles or activities. There has been a lack of concern with semantics and non-technical aspects when identifying experts. The former may result in misunderstanding data and information considered to identify experts as well as in difficulties to integrate knowledge from different sources. The latter may result in the identification of experts that possess the desired knowledge but are not the most suitable ones to share it with a particular seeker.

6 LIMITATIONS OF THE STUDY

Even though the goal of a systematic mapping is to summarize all relevant research in an area, different sets of publications can be obtained considering the number of decisions and judgments taken [52]. As any study, our study has limitations that must be considered together with the results. In addition, some challenges can reach the researchers during a systematic mapping, such as how to select a comprehensive and relevant source of publications, how to consistently apply the inclusion/exclusion criteria, how to classify data, and how to interpret them. In this study, we experienced these challenges and carried out some actions aiming at minimizing their influence on the results. In this section, we discuss some of the study limitations and some challenges we faced.

One limitation refers to the subjectivity embedded in publication selection and data extraction.Publication selection and data extraction were performed by the first author. The second author performed the same steps and the results of each reviewer were then compared. We performed an analysis of the degree of concordance in publications selection to measure the level of agreement between the results obtained from the researchers in the selection process. For this, we calculated the kappa coefficient [22] and obtained the value 0.8, which, according to Landis and Koch [22], means substantial agreement. Discordance and possible biases were discussed until we reached a consensus.

Another limitation refers to the sources and adopted search string. Terminological problems in the search string may have led to missing publications. In order to minimize these problems, we

performed previous simulations in the selected databases. Even though we have used several terms, there are still synonyms that we did not use (e.g., specialist). Therefore, relevant publications may not have been captured in our study. Concerning the sources, we decided not to search any specific conference proceedings, journals, or grey literature. Thus, we have just worked with publications indexed by the selected electronic databases. The exclusion of these other sources makes the review more repeatable, but possibly some valuable studies may have been left out of our analysis. To minimize this limitation we performed backward snowballing. The fact that we did not carry out forward snowballing (i.e., look for relevant publications by analyzing the ones citing the publications selected in the study) has to be considered as a limitation that can cause relevant publications not to be captured. Moreover, we could not reach the full text of six publications that required payment to access the paper. We contacted the authors asking for the papers but we did not receive them.

The classification schemas for categorizing data in some research questions also have some limitations. Some of them were based on classifications previously proposed in the literature (e.g., type of research [51] and software process activities [11]). Others were established during data extraction (e.g., expert sources), based on data provided by the selected publications. Determining the categories and how publications fit them involves a lot of judgment. Therefore, other researchers could obtain different results.

Lastly, data interpretation also involves tacit knowledge and judgment, which may lead different people to get different conclusions. Aiming to minimize this threat, two of the authors represented the results using charts and tables, and interpretation was performed by the three authors iteratively, considering the research questions. In this way, complementary interpretations were combined and different interpretations were discussed.

7 FINAL CONSIDERATIONS

The increasing use of agile approaches and the changes in the way people work and communicate in software organizations involve a lot of tacit knowledge and have grown the need for knowledge sharing. Therefore, identifying people who have knowledge of specific subjects and can share it with others is of paramount importance.

In this paper, we presented a mapping study that investigated approaches for identifying experts to share knowledge in the software development context. A total of 101 publications were considered and 17 of them were selected. Eight research questions were defined to investigate the following facets: (i) distribution of the selected publications over the years and the type of vehicle; (ii) research type; (iii) artifacts used as sources for identifying experts; (iv) existence of automated support; (v) use of non-technical aspects; (vi) supported roles; (vii) supported processes and activities; and (viii) use of conceptual grounds. The study contributes by providing a panorama of research related to the topic. In summary, there is a predominance of automated solutions based mainly on data stored in source repositories. The approaches have supported software development as a whole (only a few are devoted to specific roles or activities) and there has been a lack of concern with semantics and non-technical aspects when identifying experts.

From the panorama revealed by the study, we can highlight some issues that can be further investigated in future research. First, although the subject is relevant, it seems to exist a gap between theory and practice. Thus, it is important to get closer to practical settings to better understand the practitioners' needs and the constraints that may impact the proposed solutions. Only by taking the proposals to the software industry it will be possible understand what really works and what needs to be improved.

Another point is related to the artifacts used as sources of expert evidence. Although several artifacts have been used, we believe that there are opportunities to better explore them by considering new artifacts (to leverage the full range of knowledge evidence) and combining them. In this context, data and information integration issues need to be taken into account. Moreover, it would be interesting to investigate which sources can be more suitable for different knowledge needs and how to make the most of them.

The limited set of specific roles and processes supported by the approaches indicates that the approaches have been general. On one hand, this is good because does not limit the solutions to a particular problem. On the other hand, specific needs and requirements related to certain roles and processes may have been neglected. This suggests the need for further investigation of the approaches' comprehensiveness and of their effectiveness to support different roles and processes. Also, knowledge needs of specific roles and processes could be identified in order to be properly met by expert identification approaches.

The lack of concern with non-technical aspects and conceptual ground indicates an important gap. Identifying experts based only on the possessed knowledge may work in some contexts, but when is the case of knowledge sharing, other important aspects should be taken into account because they will influence directly in knowledge sharing effectiveness. Therefore, further investigation on non-technical aspects would be welcome. As for conceptual grounds, as expert identification deals much with knowledge representation, the use of ontologies, taxonomies, thesaurus, and others could be helpful. For example, they could be used to provide a common conceptualization to support mapping knowledge evidence (particularly when using multiple sources) and achieve a finer-grained understanding of knowledge evidence.

In conclusion, expert identification has become more and more necessary to support people perform software-related activities. There are several proposals on this subject and advances have been achieved in the last years. However, there are still some issues to be addressed, such as the ones aforementioned. As future work, to complement the panorama provided by the mapping study, we intend to carry out a survey with software professionals aiming to investigate how they have shared knowledge and how experts have been identified. This will enable us to compare results from the literature and practice and get insights on how to contribute to both, researchers and practitioners.

ARTIFACT AVAILABILITY

The study package (study protocol, results from the publications selection and data extraction procedures) is available in [7].

ACKNOWLEDGMENTS

This research is funded in part by the Espírito Santo Research and Innovation Support Foundation (FAPES).

REFERENCES

[1] Eytan Adar, Rajan Lukose, Caesar Sengupta, Josh Tyler, and Nathaniel Good. 2003. Shock: Aggregating information while preserving privacy. 5 (2003).

[2] Maryam Alavi and Dorothy E. Leidner. 2001. Knowledge Management and Knowledge Management Systems: Conceptual Foundations and Research Issues. 25 (2001).

[3] Mohammad Y. Allaho and Wang-Chien Lee. 2014. Increasing the Responsiveness of Recommended Expert Collaborators for Online Open Projects.

[4] Daniel Atzberger, Nico Scordialo, Tim Cech, Willy Scheibel, Matthias Trapp, and Jürgen Döllner. 2022. CodeCV: Mining Expertise of GitHub Users from Coding Activities.

[5] Krisztian Balog, Leif Azzopardi, and Maarten de Rijke. 2009. A language modeling framework for expert finding. *Information Processing & Management* 45, 1 (2009).

[6] Krisztian Balog, Yi Fang, Maarten de Rijke, Pavel Serdyukov, and Luo Si. 2012. Expertise Retrieval. (2012).

[7] Carlos Eduardo Correa Braga, Paulo Sérgio dos Santos Júnior, and Monalessa P. Barcellos. 2023. *Supplementary material of the study "Help! I need somebody. A Mapping Study about Expert Identification in Software Development".* https://doi.org/10.5281/zenodo.8154833

[8] Christopher S. Campbell, Paul P. Maglio, Alex Cozzi, and Byron Dom. 2003. Expertise Identification Using Email Communications. Association for Computing Machinery, New York, NY, USA.

[9] Luis Guillermo Cordova-Moras, Oscar Mario Rodriguez-Elias, and Maria Trinidad Serna-Encinas. 2017. Expert Location Tool for Global Software Development Environments Based on Knowledge Profile Management: A Mobile Application Approach. Mérida.

[10] Thomas H. Davenport and Laurence Prusak. 1998. *Working knowledge: how organizations manage what they know.* Harvard Business School Press, Boston, Mass.

[11] Int. Organization for Standardization, Int. Electrotechnical Commission, Institute of Electrical, and Electronics Engineers. 2017. *ISO/IEC/IEEE 12207: 2017(E) First Edition 2017-11: ISO/IEC/IEEE Int. Standard - Systems and Software Engineering – Software Life Cycle Processes.*

[12] Thomas Fritz, Gail C. Murphy, Emerson Murphy-Hill, Jingwen Ou, and Emily Hill. 2014. Degree-of-Knowledge: Modeling a Developer's Knowledge of Code. 23, 2 (2014).

[13] Chenbo Fu, Mingming Zhou, Qi Xuan, and Hong-Xiang Hu. 2017. Expert recommendation in oss projects based on knowledge embedding. In *2017 Int. Workshop on Complex Systems and Networks (IWCSN).* IEEE, Doha, 149–155.

[14] Morten T. Hansen, N. Nohria, and Tom Tierney. 1999. What's your strategy for managing knowledge? *Harvard business review* 77 2 (1999), 106–16, 187.

[15] Chaoran Huang, Lina Yao, Xianzhi Wang, Boualem Benatallah, and Xiang Zhang. 2020. Software expert discovery via knowledge domain embeddings in a collaborative network. 130 (2020).

[16] Gareth Hughes and Richard Crowder. 2003. Experiences in Designing Highly Adaptable Expertise Finder Systems. In *Volume 1: 23rd Computers and Information in Engineering Conf., Parts A and B.* ASMEDC, Chicago, Illinois, USA, 451–460.

[17] Omayma Husain, Naomie Salim, Rose Alinda Alias, Samah Abdelsalam, and Alzubair Hassan. 2019. Expert Finding Systems: A Systematic Review. (2019).

[18] John Israilidis, Evangelia Siachou, and Stephen Kelly. 2020. Why organizations fail to share knowledge: an empirical investigation and opportunities for improvement. *Information Technology & People* ahead-of-print (09 2020).

[19] Andrea Janes, Alberto Sillitti, and Giancarlo Succi. 2008. Non-invasive Software Process Data Collection for Expert Identification. In *Int. Conf. on Software Engineering and Knowledge Engineering.*

[20] B Kitchenham and S Charters. 2007. *Guidelines for performing systematic literature reviews in software engineering.* Technical Report. Technical report, EBSE Technical Report EBSE-2007-01.

[21] Jacob Krüger, Jens Wiemann, Wolfram Fenske, Gunter Saake, and Thomas Leich. 2018. Do You Remember This Source Code?. In *Proc. of the 40th Int. Conf. on Software Engineering* (Gothenburg, Sweden) *(ICSE '18).* Association for Computing Machinery, New York, NY, USA, 764–775.

[22] J. Richard Landis and Gary G. Koch. 1977. The Measurement of Observer Agreement for Categorical Data. *Biometrics* 33, 1 (March 1977), 159.

[23] Shuyi Lin, Wenxing Hong, Dingding Wang, and Tao Li. 2017. A survey on expert finding techniques. *Journal of Intelligent Information Systems* 49 (10 2017).

[24] Edson M. Lucas, Toacy C. Oliveira, Daniel Schneider, and Paulo S. C. Alencar. 2020. Knowledge-Oriented Models Based on Developer-Artifact and Developer-Developer Interactions. 8 (2020).

[25] Vitor Mangaravite, Rodrygo L. T. Santos, Isac S. Ribeiro, Marcos Andre Gonçalves, and Alberto H. F. Laender. 2016. The LExR Collection for Expertise Retrieval in Academia. *Proc. of the 39th Int. ACM SIGIR Conf. on Research and Development in Information Retrieval* (2016).

[26] David W. McDonald and Mark S. Ackerman. 1998. Just Talk to Me: A Field Study of Expertise Location. Association for Computing Machinery.

[27] Wenkai Mo, Beijun Shen, Yuming He, and Hao Zhong. 2015. GEMiner: Mining Social and Programming Behaviors to Identify Experts in Github. *Proc. of the 7th Asia-Pacific Symposium on Internetware* (2015).

[28] A. Mockus and J.D. Herbsleb. 2002. Expertise Browser: a quantitative approach to identifying expertise. In *Proc. of the 24th Int. Conf. on Software Engineering. ICSE 2002.* 503–512.

[29] Roziah Mohd Rasdi and Gangeswari Tangaraja. 2022. Knowledge-sharing behaviour in public service organisations: determinants and the roles of affective commitment and normative commitment. 46 (2022).

[30] Alan Moraes, Eduardo Silva, Cleyton da Trindade, Yuri Barbosa, and Silvio Meira. 2010. Recommending experts using communication history.

[31] Itzel Morales-Ramirez, Matthieu Vergne, Mirko Morandini, Anna Perini, and Angelo Susi. 2015. Exploiting Online Discussions in Collaborative Distributed Requirements Engineering. In *Int. i* Workshop.*

[32] Muhammad Naeem, Muhammad Bilal Khan, and Muhammad Tanvir Afzal. 2013. Expert Discovery: A web mining approach. *Journal of AI and Data Mining* 1 (2013), 35–47.

[33] Julio Cesar Nardi, Ricardo de Almeida Falbo, and João Paulo A. Almeida. 2013. Foundational Ontologies for Semantic Integration in EAI: A Systematic Literature Review. In *Collaborative, Trusted and Privacy-Aware e/m-Services*, Christos Douligeris, Nineta Polemi, Athanasios Karantjias, and Winfried Lamersdorf (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 238–249.

[34] Ikujirō Nonaka and Hirotaka Takeuchi. 1995. *The knowledge-creating company: how Japanese companies create the dynamics of innovation.* Oxford University Press, New York.

[35] Ignacio Nuñez Norambuena and Alexandre Bergel. 2021. Building a bot for automatic expert retrieval on discord. In *Proc. of the 5th Int. Workshop on Machine Learning Techniques for Software Quality Evolution.* ACM, Athens Greece, 25–30.

[36] D.E. O'Leary. 1998. Enterprise knowledge management. 31 (1998).

[37] Johnatan Oliveira, Maurício Souza, Matheus Flauzino, Rafael Durelli, and Eduardo Figueiredo. 2022. Can Source Code Analysis Indicate Programming Skills? A Survey with Developers. Vol. 1621. Cham.

[38] Kai Petersen, Sairam Vakkalanka, and Ludwik Kuzniarz. 2015. Guidelines for conducting systematic mapping studies in software engineering: An update. (2015).

[39] Stanislav Vassilev Pokraev. 2009. *Model-driven semantic integration of service-oriented applications.* Ph. D. Dissertation. University of Twente.

[40] I. Rus and M. Lindvall. 2002. Knowledge management in software engineering. 19 (2002).

[41] Vinicius Schettino, Vitor Horta, Marco Antonio P. Araujo, and Victor Stroele. 2019. Towards Community and Expert Detection in Open Source Global Development. IEEE.

[42] Kurt Schneider. 2009. *Experience and Knowledge Management in Software Engineering.* Springer Berlin Heidelberg, Berlin, Heidelberg.

[43] Dawit Yimam Seid and Alfred Kobsa. 2003. Expert-Finding Systems for Organizations: Problem and Domain Analysis and the DEMOIR Approach. *Journal of Organizational Computing and Electronic Commerce* 13 (2003), 1 – 24.

[44] Eva. Semertzaki. 2011. *Special libraries as knowledge management centres / Eva Semertzaki.* Chandos Oxford. xxii, 314 p. ; pages.

[45] N. Sadat Shami, Y. Connie Yuan, Dan Cosley, Ling Xia, and Geri Gay. 2007. That's what friends are for: facilitating 'who knows what' across group boundaries. In *Proc. of the 2007 Int. ACM Conf. on Conf. on supporting group work - GROUP '07.* ACM Press, Sanibel Island, Florida, USA, 379.

[46] Amitoj Singh, Vinay Kukreja, and Munish Kumar. 2023. An empirical study to design an effective agile knowledge management framework. *Multimedia Tools and Applications* 82, 8 (March 2023), 12191–12209.

[47] Ralf Teusner, Christoph Matthies, and Philipp Giese. 2017. Should I Bug You? Identifying Domain Experts in Software Projects Using Code Complexity Metrics. In *2017 IEEE Int. Conf. on Software Quality, Reliability and Security (QRS).* IEEE, Prague, Czech Republic, 418–425.

[48] Holger Wache, Thomas Vögele, Ubbo Visser, Heiner Stuckenschmidt, Gerhard Schuster, H. Neumann, and S. Ubner. 2002. Ontology-Based Integration of Information - A Survey of Existing Approaches. *Proc. of the IJCAI'01 Workshop on Ontologies and Information Sharing, Seattle, Washington, USA, Aug 4-5* (08 2002).

[49] Wasko and Faraj. 2005. Why Should I Share? Examining Social Capital and Knowledge Contribution in Electronic Networks of Practice. 29 (2005).

[50] David Weiss and James Shanteau. 2003. Empirical Assessment of Expertise. *Human factors* 45 (02 2003), 104–16. https://doi.org/10.1518/hfes.45.1.104.27233

[51] Roel Wieringa, Neil Maiden, Nancy Mead, and Colette Rolland. 2006. Requirements engineering paper classification and evaluation criteria: A proposal and a discussion. *Requir. Eng.* 11 (March 2006), 102–107.

[52] Claes Wohlin, Per Runeson, Martin Hst, Magnus C. Ohlsson, Bjrn Regnell, and Anders Wessln. 2012. *Experimentation in Software Engineering.* Springer Publishing Company, Incorporated.