



**UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO**  
**CENTRO TECNOLÓGICO**  
**PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA**

Carlos Eduardo Correa Braga

# **Using Skill Manifestations to identify Who Knows What in Software Organizations**

Vitória, ES

2024

Carlos Eduardo Correa Braga

# **Using Skill Manifestations to identify Who Knows What in Software Organizations**

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Informática da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Mestre em Informática.

Universidade Federal do Espírito Santo – UFES

Centro Tecnológico

Programa de Pós-Graduação em Informática

Supervisor: Prof. Monalessa Perini Barcellos, D. Sc

Vitória, ES

2024

# Acknowledgements

I believe that every experience in life presents an opportunity to evolve as a human being. In the journey to complete this work, I had to evolve in so many aspects. I had to learn that, as Steve Jobs once said, focus is more about saying 'no' than saying 'yes'. I also needed to improve my ability to manage both my time and my emotions. However, I was not alone on this path, and achieving this accomplishment was only possible with the support of many people.

I am deeply grateful to my parents, Maria and Claudio, for their effort to provide me with quality education throughout my life. I am also grateful to my wife Raysa for supporting me during this journey. It is truly a privilege to have her by my side. I am grateful to my dear friends - França, Murillo, Paulo, Abraão, and Camillo - whose encouragement and support have contributed to my pursuit of this achievement.

I am immensely grateful to Prof. Monalessa for providing me with the opportunity to accomplish this and for her guidance, sharing so many invaluable lessons not only in academic research but also in life. I also would like to thank Prof. Falbo (in memoriam) for introducing me to the world of academic research: I have finally made it. I am grateful to Prof. Vítor and Prof. Sheila for dedicating their time to review this work and for sharing your invaluable perceptions that have helped improve it.

Finally, I would like to express my gratitude to PPGI-UFES, and the Federative Republic of Brazil, for offering this postgraduate program that contributed to my academic, professional, and personal growth.

# Resumo

O desenvolvimento de software é uma atividade fortemente baseada em conhecimento e seu sucesso em uma organização depende do compartilhamento de conhecimento existente. Os desafios de gestão do conhecimento são frequentemente potencializados em ambientes ágeis, que envolvem muito conhecimento tácito, comumente adquirido por meio de experiências e difícil de explicitar. Portanto, o compartilhamento de conhecimento entre os profissionais é crucial. No entanto, identificar especialistas adequados para compartilhar conhecimento não é uma tarefa simples. Isso envolve não apenas descobrir os indivíduos com o conhecimento desejado, mas também considerar outros fatores que podem impactar na efetividade da troca de conhecimento, como proximidade social e disponibilidade. Além disso, a grande quantidade de dados e a natureza específica do conhecimento dos especialistas podem dificultar a identificação de especialistas com o conhecimento necessário. Sistemas de identificação de especialistas têm sido propostos para resolver esse problema. Eles são sistemas de recuperação de informação que identificam candidatos a especialistas e os classificam com base em sua expertise em um determinado assunto. No entanto, desenvolver sistemas de identificação de especialistas é desafiador devido ao envolvimento de conceitos complexos como expertise, habilidade, conhecimento e outros conceitos relacionados. Além disso, integrar diversas fontes heterogêneas que oferecem evidências de expertise apresenta uma dificuldade. As ontologias podem ser úteis nesse aspecto ao fornecer semântica aos dados e resolver conflitos semânticos que podem levar ao mau uso ou à interpretação incorreta dos dados. Com o objetivo de utilizar ontologias para apoiar a identificação de especialistas, este trabalho propõe *iKnow*, uma abordagem baseada em uma conceituação bem fundamentada que auxilia no desenvolvimento de sistemas para descobrir quem sabe o quê em uma organização de software, considerando dados armazenados nos repositórios da organização e fatores não técnicos que podem afetar o compartilhamento de conhecimento. Um sistema foi desenvolvido utilizando a abordagem proposta para evidenciar como ela pode ser usada para apoiar o desenvolvimento de sistemas de identificação de especialistas, demonstrando a viabilidade da proposta. A ferramenta foi usada para apoiar a identificação de especialistas para fins de compartilhamento de conhecimento dentro de uma empresa. Os resultados preliminares sugerem que a ferramenta desenvolvida seguindo *iKnow* é útil.

**Palavras-chaves:** sistemas de busca de especialistas, gerência de conhecimento, engenharia de software, ontologias.

# Abstract

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 make 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. Moreover, 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. Developing expert-finding systems is challenging due to the involvement of complex concepts such as expertise, skill, knowledge, and other related concepts. Additionally, integrating diverse heterogeneous sources that offer evidence of expertise poses a difficulty. Ontologies can be helpful in this matter by providing semantics to data and addressing semantic conflicts that can lead to data misuse or misinterpretation. Aiming to benefit from the use of ontologies to identify experts, this work proposes *iKnow*, an approach based on a well-founded conceptualization that helps in the development of systems to find who knows what in a software organization considering data stored in the organization repositories and non-technical factors that may affect knowledge sharing. A computational tool was developed using the proposed approach to illustrate how it can be used to aid the development of expert-finding systems, demonstrating the proposal's feasibility. The tool was used to support expert identification for knowledge-sharing purposes within a company. The preliminary results suggest that the tool developed by following *iKnow* is useful.

**Keywords:** expert finding systems, knowledge management, software engineering, ontology.

# List of Figures

Figure 1 – Overview of Design Science Research cycles applied in this work . . . .	16
Figure 2 – Modes of knowledge creation (NONAKA, 1994) . . . . .	19
Figure 3 – Factors that influence knowledge sharing between individuals in organizations (IPE, 2003) . . . . .	20
Figure 4 – Ontology generality levels as a continuum (adapted from (FALBO et al., 2013; RUY, 2017)) . . . . .	24
Figure 5 – Fragment of Core-O Personal Competence Ontology (adapted from (CALHAU et al., 2024)) . . . . .	27
Figure 6 – Fragment of the Core-O Competence Type Ontology (adapted from (CALHAU et al., 2024)) . . . . .	28
Figure 7 – Publications selection . . . . .	33
Figure 8 – Publication year and vehicle . . . . .	36
Figure 9 – Sources for identifying experts . . . . .	37
Figure 10 – Core-O extract used in <i>iKnow</i> . . . . .	48
Figure 11 – iKnow steps . . . . .	50
Figure 12 – O*Net skill types summary for Software Developer . . . . .	52
Figure 13 – <i>Org</i> spreadsheet . . . . .	56
Figure 14 – <i>Org</i> hypothetical information model . . . . .	57
Figure 15 – An architecture for developing applications to support the identification of who knows what . . . . .	58
Figure 16 – Skill types identified in BMC . . . . .	64
Figure 17 – An extended fragment of Core-O . . . . .	66
Figure 18 – Azure Boards conceptual model . . . . .	67
Figure 19 – Datadog conceptual model . . . . .	68
Figure 20 – Git conceptual model . . . . .	69
Figure 21 – ExpertFY conceptual model . . . . .	70
Figure 22 – ExpertFY architecture . . . . .	70
Figure 23 – Search panel with filters in ExpertFY . . . . .	71
Figure 24 – Search results in ExpertFY . . . . .	71
Figure 25 – Profile detail section in ExpertFY . . . . .	72
Figure 26 – Profile skills and recommendations section in ExpertFY . . . . .	72
Figure 27 – Profile tasks in ExpertFY . . . . .	73
Figure 28 – Fragment of the feedback questionnaire about the use of ExpertFY . . . . .	75
Figure 29 – Participants Profile Form . . . . .	98
Figure 30 – Assumptions Evaluation Form (part 1) . . . . .	99
Figure 31 – Assumptions Evaluation Form (part 2) . . . . .	100

Figure 32 – Non-Technical Factor Selection Form . . . . .	101
Figure 33 – Summarized results of assumptions evaluation . . . . .	102

# List of Tables

Table 1 – OntoUML stereotypes relevant to this work . . . . .	26
Table 2 – Research questions and their rationale . . . . .	30
Table 3 – Selected publications . . . . .	34
Table 4 – Research Type . . . . .	36
Table 5 – Automated support . . . . .	37
Table 6 – Recommendation based on non-technical aspects . . . . .	37
Table 7 – Supported roles . . . . .	38
Table 8 – Supported processes/activities . . . . .	38
Table 9 – Conceptual ground . . . . .	38
Table 10 – Naming convention . . . . .	49
Table 11 – Correlation between <i>iKnow</i> steps and expert-finding topics defined in (HUSAIN et al., 2019) . . . . .	51
Table 12 – Aspects addressed by expert-finding system approaches . . . . .	60
Table 13 – Skill type manifestation assumptions . . . . .	65
Table 14 – Score level of agreement for assumption . . . . .	65
Table 15 – Azure Boards concepts mapping . . . . .	68
Table 16 – Datadog concepts mapping . . . . .	68
Table 17 – Git concepts mapping . . . . .	69
Table 18 – Participants characteristics . . . . .	76
Table 19 – Selection data by participant . . . . .	76
Table 20 – Specific objectives of this work . . . . .	83
Table 21 – Skill type manifestation assumptions . . . . .	96
Table 22 – Weighted Options . . . . .	101
Table 23 – Assumptions score sorted descending by score . . . . .	102

# Contents

<b>1</b>	<b>INTRODUCTION</b>	<b>10</b>
1.1	Context and Motivation	10
1.2	Objectives	12
1.3	Methodological Approach	13
1.4	Structure of this Document	15
<b>2</b>	<b>BACKGROUND</b>	<b>18</b>
2.1	Knowledge Management	18
2.2	Expert Finding Systems	21
2.3	Ontologies	22
2.3.1	Unified Foundational Ontology (UFO) and OntoUML	24
2.3.2	Core-O: A Competence Reference Ontology	25
2.4	Concluding Remarks	27
<b>3</b>	<b>A MAPPING STUDY ABOUT EXPERT IDENTIFICATION IN SOFTWARE DEVELOPMENT</b>	<b>29</b>
3.1	Introduction	29
3.2	Research Protocol	30
3.3	Data Extraction and Synthesis	33
3.4	Data Analysis and Discussion	39
3.5	Threats to Validity	43
3.6	Concluding Remarks	45
<b>4</b>	<b>AN ONTOLOGY-BASED APPROACH TO SUPPORT THE DEVELOPMENT OF SYSTEMS TO IDENTIFY WHO KNOWS WHAT IN SOFTWARE ORGANIZATIONS</b>	<b>47</b>
4.1	Introduction	47
4.2	iKnow: An Ontology-based Approach to Support the Development of Systems to Identify Who Knows What in Software Organizations	49
4.3	Related Work	59
4.4	Concluding Remarks	61
<b>5</b>	<b>EXPERTFY: A SYSTEM TO IDENTIFY WHO KNOWS WHAT IN A SOFTWARE ORGANIZATION</b>	<b>62</b>
5.1	Introduction	62
5.2	ExpertFY: <i>An Expert for You</i>	62

5.2.1	Study Design . . . . .	62
5.2.2	Study Execution and Results . . . . .	63
<b>5.3</b>	<b>A Survey with ExpertFY Users . . . . .</b>	<b>72</b>
5.3.1	Survey Planning . . . . .	73
5.3.2	Survey Execution . . . . .	74
5.3.3	Survey Results . . . . .	76
5.3.4	Survey Discussion . . . . .	78
<b>5.4</b>	<b>Threats to Validity . . . . .</b>	<b>80</b>
<b>5.5</b>	<b>Concluding Remarks . . . . .</b>	<b>81</b>
<b>6</b>	<b>CONCLUSION . . . . .</b>	<b>82</b>
<b>6.1</b>	<b>Final Considerations . . . . .</b>	<b>82</b>
<b>6.2</b>	<b>Contributions . . . . .</b>	<b>84</b>
<b>6.3</b>	<b>Future Work . . . . .</b>	<b>85</b>
	<b>BIBLIOGRAPHY . . . . .</b>	<b>87</b>
	<b>APPENDIX . . . . .</b>	<b>95</b>
<b>A</b>	<b>Artifacts used in Application of iKnow . . . . .</b>	<b>96</b>
A.1	Introduction . . . . .	96
A.2	Assumptions and Non-Technical Factors . . . . .	96
A.3	Results . . . . .	97
<b>B</b>	<b>Artifacts used in ExpertFY Evaluation Study . . . . .</b>	<b>103</b>
B.1	Instructions Document . . . . .	103
B.2	Consent Form . . . . .	106
B.3	Participants Profile Form . . . . .	107
B.4	Feedback Questionnaire . . . . .	108

# 1 Introduction

*This chapter presents the context, motivation and objectives of this work, as well as the adopted research approach and the structure of this dissertation.*

## 1.1 Context and Motivation

Knowledge is a human specialty stored in people's minds, acquired through experience and interaction with their environment (SCHNEIDER, 2009). Historically, organizations' knowledge has often been undocumented, being represented through the skills, experience, and knowledge of their professionals, typically tacit knowledge (RUS; LINDVALL, 2002), which makes its use and access limited and difficult (O'LEARY, 1998).

Knowledge is a mix of experience, values, contextual information, and expert insight that provides a framework for evaluating and incorporating new experiences and information (DAVENPORT; PRUSAK, 1998). It can be explicit or tacit. Explicit knowledge is structured and formal (SEMERTZAKI, 2011). 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 (NONAKA; TAKEUCHI, 1995). The interactions between individuals enable the creation of new knowledge through a continuous conversion between tacit and explicit knowledge (NONAKA; TAKEUCHI, 1995).

In order to effectively handle knowledge in an organization, a knowledge management (KM) approach is necessary. KM comprises 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 (ALAVI; LEIDNER, 2001). In particular, knowledge sharing (or knowledge transfer) is the process of transferring knowledge between individuals, groups, or organizations (ALAVI; LEIDNER, 2001).

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 (SINGH; KUKREJA; KUMAR, 2023). 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 of expert in the literature, such as an

individual who has skills in their domain of expertise (WEISS; SHANTEAU, 2003); an individual who has a great deal of knowledge and displays comprehensive skills in a specific area (LIN et al., 2017); and a person who has the best knowledge and experience in a particular topic (NAEEM; KHAN; AFZAL, 2013). 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 minds and are therefore difficult to codify (DAVENPORT; PRUSAK, 1998; HUSAIN et al., 2019).

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 (HUSAIN et al., 2019). 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 (LIN et al., 2017).

Identifying experts in software organizations is a challenging task, especially when it comes to the context of large-scale distributed software projects (MCDONALD; ACKERMAN, 1998; SEID; KOBSA, 2003; MANGARAVITE et al., 2016). Individuals can seek out a particular knowledge by asking people and following a trail of referrals until they locate the appropriate expert (CAMPBELL et al., 2003). 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 (CAMPBELL et al., 2003). Moreover, knowledge possession is not the only factor that influences expert responsiveness. Other factors are also important, such as time availability (RASDI; TANGARAJA, 2022), communication skills, and interest in building reputation (Wasko; Faraj, 2005).

In software development, developers often rely on past experiences to recall similar tasks and identify the most knowledgeable individual for help. However, as the number 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 (LUCAS et al., 2020).

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 (BALOG et al., 2012; HUSAIN et al., 2019). In these systems, expertise is extracted from evidence recorded in sources such as publications, reports, projects, homepages, social

networks (BALOG; AZZOPARDI; RIJKE, 2009), and source code repositories (OLIVEIRA et al., 2022; ATZBERGER et al., 2022).

Developing expert-finding systems is challenging due to the involvement of complex concepts such as expertise, skill, knowledge, and other related concepts. Additionally, integrating diverse sources of expertise poses a difficulty (HUSAIN et al., 2019). For instance, software organizations often use various applications to support different aspects of software development (FITZGERALD; STOL, 2017), and they may contain valuable data that can be leveraged as evidence of expertise. However, the applications are often heterogeneous, i.e., each application implements its own data and behavioral models and focuses on specific aspects of the software process, with little concern for sharing and integration aspects, leading thus to several semantic conflicts (CALHAU; FALBO, 2010).

Ontologies can be helpful in this matter. An ontology is a formal and explicit specification of a shared conceptualization (STUDER; BENJAMINS; FENSEL, 1998). They can be used to capture and organize knowledge based on a common vocabulary to deal with interoperability and knowledge-related problems. Thus, in the expert finding context, they can help by providing a common understanding of concepts related to knowledge and experts. Moreover, they can be used as interlingua to map the concepts used by different applications, enabling data understanding and integration (CALHAU; FALBO, 2010).

The literature has proposed some works to support finding experts, such as (LUCAS et al., 2020; MORAES et al., 2010; MOCKUS; HERBSLEB, 2002). 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 code repositories. Additionally, 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 data from different sources. The latter may result in the identification of experts who possess the desired knowledge but are not the most suitable ones to share it with a particular seeker (BRAGA; JR; BARCELLOS, 2023).

To bridge these gaps, in this work, we propose to use ontologies and consider non-technical factors to aid in identifying *who knows what* in software organizations. Moreover, we consider different artifacts and tasks as skill manifestations (i.e., evidence of that skill) that help identify experts.

## 1.2 Objectives

This work has the **main objective** of *exploring ontologies, software development tasks and artifacts, and non-technical factors to help identify who knows what in software*

organizations. This main objective can be detailed in the following *specific objectives*:

- *SO1*. Obtain a panorama of the state of the art about expert identification in the software development context;
- *SO2*. Propose an ontology-based approach to support identifying who knows what in software organization considering software development tasks, produced artifacts, and non-technical aspects;
- *SO3*. Develop a computational solution to support identifying who knows what based on the proposed approach;
- *SO4*. Apply the proposed solution to solve expert identification problems in practical settings.

### 1.3 Methodological Approach

The methodological approach adopted in this work followed the *Design Science Research* (DSR) paradigm, which concerns extending human and organizational capabilities by creating new and innovative artifacts (HEVNER, 2007). DSR was selected as the research approach because the object of study is an artifact — specifically, an ontology-based approach to support the identification of who knows what in software organizations — and its evaluation is possible in a real organizational environment.

DSR comprises the following steps (PEFFERS et al., 2007): (i) *Problem identification and motivation*; (ii) *Definition of the objectives for a solution*; (iii) *Design and development*; (iv) *Demonstration*; (v) *Evaluation*; (vi) *Communication*. These six steps are associated with three cycles that characterize DSR as an iterative process, as defined by Hevner (2007): *Relevance Cycle*, *Design Cycle*, and *Rigor Cycle*.

A DSR project begins with the *Relevance Cycle*, which involves defining the problem to be addressed, requirements, and criteria for evaluating the results (HEVNER, 2007), including steps (i) and (ii). The *Design Cycle* involves developing and evaluating artifacts or theories to solve the identified problem (HEVNER, 2007), comprising steps (iii), (iv), and (v). The *Rigor Cycle* refers to using and generating knowledge (HEVNER, 2007), comprising step (vi), and the use of knowledge and foundations along with the work.

In the *Problem identification and motivation* step, the problem was identified at firsthand in practice by the author of this dissertation, when working on software development as a tech manager. This author noticed problems in identifying individuals in the organization with the needed knowledge to help others execute their tasks. It was often necessary to contact multiple individuals before finding the one with the appropriate expertise to support others to complete a task. Another common pattern involved contacting

long-standing members of the organization who, despite lacking the specific expertise needed, could typically guide the seeker to the right individual for assistance. These situations revealed the existence of tacit knowledge about who knows what within the organization. This tacit knowledge is difficult to communicate and, therefore, is not easily accessible, especially for new members of the organization.

The problem was also identified in the literature. Initially, an informal literature review was performed to learn about the research topic. As a result, the problem to be focused on by this work was established as the need to address difficulties involved in identifying who knows what in software organizations for knowledge-sharing purposes. To further explore the subject and identify its main gaps, we performed a systematic mapping of the literature to investigate the state of the art about expert identification. The mapping results indicated, among others, that (i) there is an opportunity for exploring a wider range of artifacts and sources of expert evidence; (ii) the existing approaches have been generic and lack considering specific needs and requirements; (iii) non-technical aspects important for knowledge sharing have often been disregarded; (iv) there has not been a concern with semantic aspects, and the rationale used for identifying who knows what has not been explicitly stated.

Considering the identified problem, the gaps pointed out by the mapping study, and the successful use of ontologies to address semantic and knowledge-related problems, in the step *Definition of the objectives for a solution*, we decided to develop an ontology-based approach to develop systems to support identifying who knows what in software organizations. As requirements to the approach, we defined that it must: (R1) use ontologies to assign semantics and identify who knows what; (R2) guide the steps to be followed to create the who knows what identification solution; (R3) consider different artifacts and tasks as source of knowledge evidence; and (R4) consider non-technical aspects. In addition to the requirements to be met by the proposed approach, we defined the following criteria to evaluate it: (C1) the approach must be useful, and (C2) its use must be feasible.

During the *Design and development* step, we developed *iKnow*, an ontology-based approach to support the development of systems to identify who knows what in software organizations. To address R1, *iKnow* uses Core-O (CALHAU et al., 2024), an ontology about competencies, to support the identification of who knows what and assign semantics to data about skill manifestations. As defined in (CALHAU et al., 2024), human capabilities (skills, in particular) are manifested by individuals through the execution of human tasks, where each human task creates an artifact as a task output - e.g., the implementation of a web application creates code files as a task output. Furthermore, skills relate to the ability to apply knowledge to perform a task. In this sense, the execution of tasks serves as evidence of an individual's skill and indicates their knowledge in the domain. Therefore, adding the proper semantics, we can use data on human task executions and their outputs

(i.e., skill manifestations) to identify who knows what in software organizations. To meet R2, *iKnow* defines an eight-step process. To satisfy R3, *iKnow* considers several artifacts and tasks as skill manifestations. As for R4, *iKnow* allows considering different non-technical aspects.

In the *Demonstration* step, we used *iKnow* to develop ExpertFY, a system to support the identification and recommendation of experts, which demonstrated that using the proposed approach is feasible (C2). Then, in the *Evaluation* step, ExpertFY was used in an organization and evaluated by five software developers. The results show that the computational solution developed by using *iKnow* is useful, which also suggests that *iKnow* is useful (C1).

Finally, in the *Communication* step, we produced artifacts to communicate the research results to the interested parties. This involved elaborating this dissertation and one paper published in the 37th Brazilian Symposium on Software Engineering (SBES 2023) (BRAGA; JR; BARCELLOS, 2023), and a study package (BRAGA; JÚNIOR; BARCELLOS, 2023).

All the aforementioned steps were based on the relevant literature, which includes papers, books, and other materials about Knowledge Management, Expert Finding, and Ontology. The main contribution of this work is *iKnow*, an approach for systematically developing systems that support the identification of experts for knowledge-sharing purposes. There are also other contributions: (i) the mapping study investigating expert identification to share knowledge in software development; and (ii) ExpertFY, a system to support the identification and recommendation of experts. Figure 1 summarizes the Design Science cycles performed in this work.

## 1.4 Structure of this Document

This opening chapter presented the core ideas of this dissertation, depicting the context, motivation, objectives, and the followed research approach. The following chapters are also part of this dissertation:

- Chapter 2 (Background): provides the background for this work, addressing aspects related to knowledge management, expert finding systems, and ontologies.
- Chapter 3 (A Mapping Study about Expert Identification in Software Development): presents a systematic mapping of the literature that investigated experts identification to share knowledge in software development.
- Chapter 4 (iKnow: An Ontology-based Approach to Support the Development of Systems to Identify Who Knows What in Software Organizations): presents the

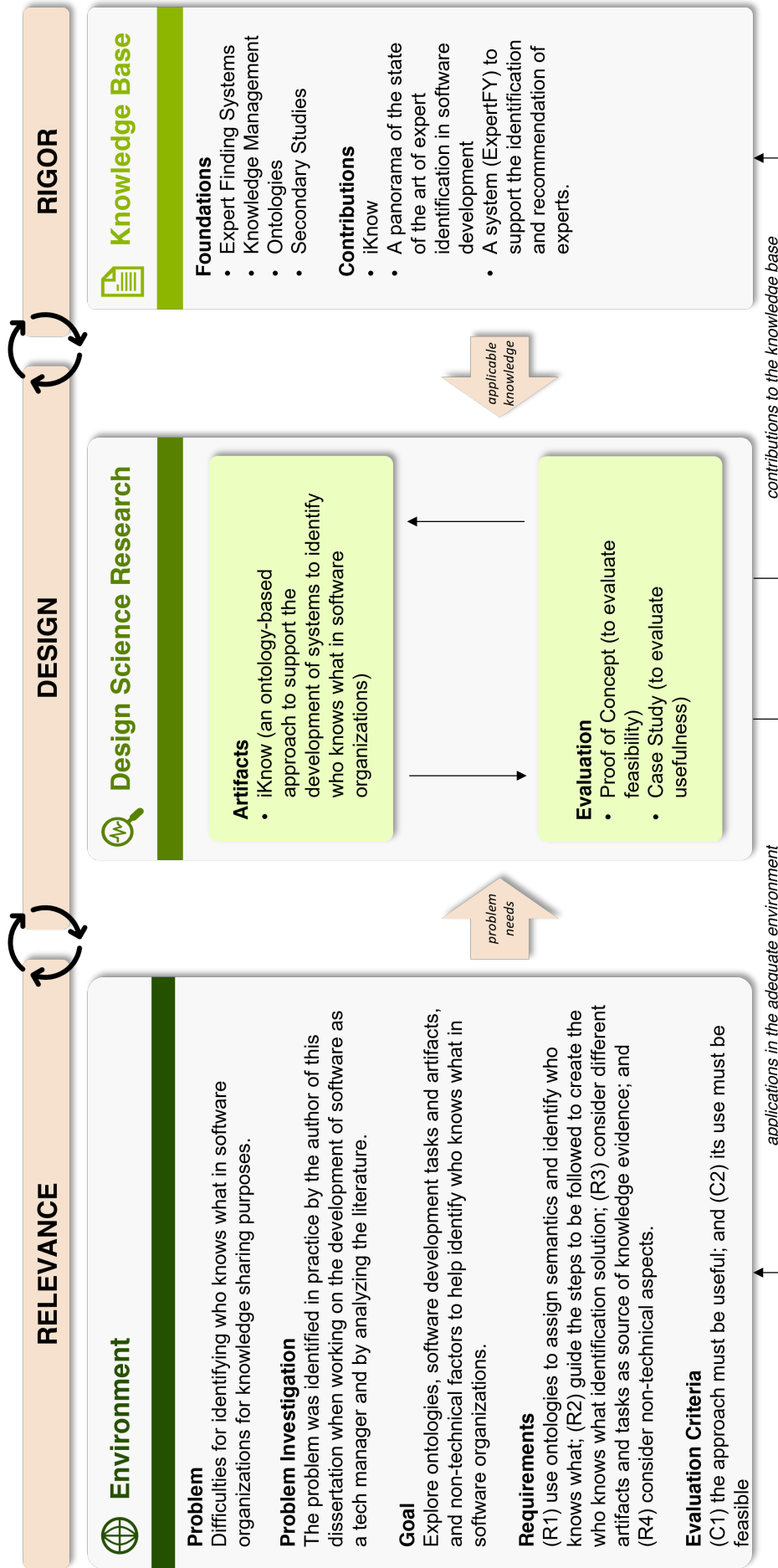


Figure 1 – Overview of Design Science Research cycles applied in this work

approach proposed in this work. It guides the development of systems to support the identification of who knows what in an organization for knowledge-sharing purposes.

- Chapter 5 (ExpertFY: A System to Identify Who Knows What in a Software Organization): reports the application of the proposed approach to developing a system (ExpertFY) used in an organization and the evaluation of it. The use of *iKnow* to develop ExpertFY served as a proof of concept and demonstrated the feasibility of using the proposal. The evaluation of ExpertFY helped evaluate *iKnow*'s usefulness.
- Chapter 6 (Conclusion): presents the final considerations, contributions, and proposals for future work to continue and improve the proposed work.
- Appendix A (Artifacts used in Application of *iKnow*): presents the forms used in the application of *iKnow*.
- Appendix B (Artifacts used in ExpertFY Evaluation Study): presents the forms and the instructions document used in ExpertFY evaluation study.

## 2 Background

*This chapter presents the background for this work. Section 2.1 addresses Knowledge Management. Section 2.2 concerns Expert Finding Systems. Section 2.3 regards Ontologies, presenting the theoretical foundations and Core-O, the ontology used in this work. Finally, Section 2.4 presents the concluding remarks of the chapter. This chapter concerns knowledge used to ground our work, thus, in the DSR context, it is related to the Rigor cycle (Figure 1).*

### 2.1 Knowledge Management

As presented in Section 1.1, explicit knowledge is structured and formal (SE-MERTZAKI, 2011), and 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 (NONAKA; TAKEUCHI, 1995). Knowledge is created through a continuous conversion between tacit and explicit knowledge (NONAKA, 1994). This conversion can occur in four different modes according to the SECI model (NONAKA, 1994) presented in Figure 2, as follows:

- **Socialization:** is the conversion of tacit knowledge into tacit knowledge through social interactions and shared experience among organizational members.
- **Externalization:** is the conversion of tacit knowledge into explicit knowledge through symbolic representation of tacit knowledge, usually in the form of models, textual documents, and diagrams, among others.
- **Combination:** is the conversion of explicit knowledge into explicit knowledge by merging, categorizing, reclassifying, and synthesizing existing explicit knowledge.
- **Internalization:** is the conversion of explicit knowledge into tacit knowledge, usually by reading documents, for example.

Externalization and socialization 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 (HANSEN; NOHRIA; TIERNEY, 1999). As the name suggests, socialization approaches rely on personal interaction. They require identifying who has specific knowledge and promoting knowledge sharing between the person who possesses

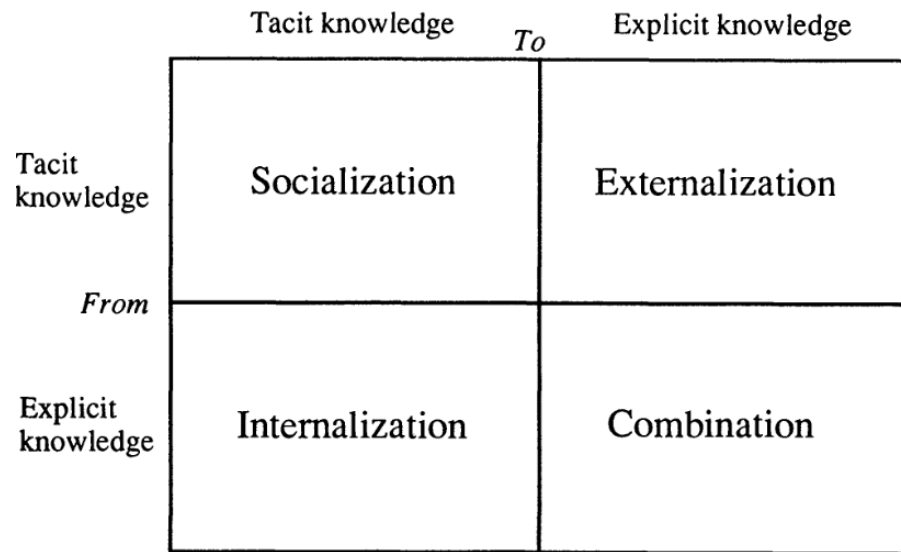


Figure 2 – Modes of knowledge creation (NONAKA, 1994)

the desired knowledge (the expert) and the person interested in that particular knowledge (the seeker).

Knowledge Management (KM) is the deliberate and systematic coordination of an organization's people, technology, processes, and organizational structure to add value through reuse and innovation (DALKIR, 2013). This coordination is achieved through creating, sharing, and applying knowledge as well as through feeding the valuable lessons learned and best practices into corporate memory to foster continuous organizational learning (DALKIR, 2013). KM supports not only the *know-how* of a company, but also the *know-where*, *know-who*, *know-what*, *know-when*, and *know-why* (RUS; LINDVALL, 2002).

Alavi e Leidner (2001) proposed a framework describing organizations as knowledge systems consisting of four knowledge processes: knowledge creation; knowledge storage and retrieval; knowledge sharing; and knowledge application.

*Knowledge creation* involves developing new content or replacing content within the organization's tacit and explicit knowledge. As previously presented, the SECI model (NONAKA, 1994) identifies four modes of knowledge creation. Information systems designed to support collaboration, coordination, and communication processes can facilitate teamwork and increase an individual's contact with other individuals. Therefore, this may accelerate the growth of knowledge creation (NONAKA, 1994).

While organizations create knowledge, they also do not remember or lose track of acquired knowledge (ARGOTE; BECKMAN; EPPLE, 1990; DARR; ARGOTE; EPPLE, 1995). Thus, *knowledge storage and retrieval*, also referred to as organizational memory, constitute an important aspect of effective organizational KM. Organizational memory includes knowledge in various forms such as written documentation and databases, among others.

*Knowledge sharing* occurs at different levels: between individuals, from individuals to explicit sources, from individuals to groups, between groups, across groups, and from the group to the organization (ALAVI; LEIDNER, 2001). Considering the distributed nature of organizational cognition, the transfer of knowledge to locations where it is needed and can be used is particularly important (ALAVI; LEIDNER, 2001).

The source of competitive advantage in an organization resides in the *knowledge application* rather than in the knowledge itself (ALAVI; LEIDNER, 2001). Therefore, knowledge is especially valuable when applied to tasks, projects, or business processes.

As presented in Figure 3, Ipe (2003) identified four major factors that influence knowledge sharing between individuals in organizations: the nature of knowledge, motivation to share, opportunities to share, and the culture of the work environment.

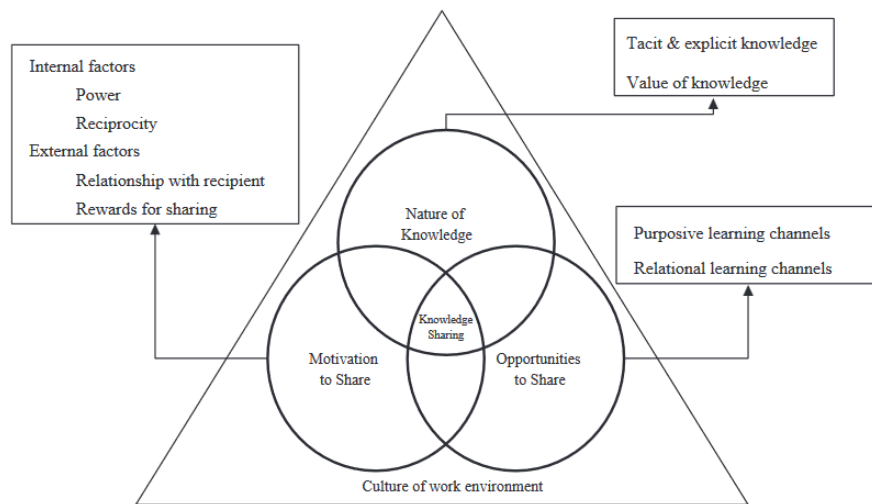


Figure 3 – Factors that influence knowledge sharing between individuals in organizations (IPE, 2003)

Regarding the *nature of knowledge*, the characteristics of tacit knowledge create barriers to sharing, as it requires personal interaction for effective transfer. On the other hand, considering that is codified and stored, explicit knowledge tends to be easily shared between individuals. However, Weiss (1999) argued that the ability to articulate knowledge should not be equated with its availability for use by others in the organization. He introduced the concepts of *rationalized knowledge* and *embedded knowledge*. *Rationalized knowledge* is general and context-independent (e.g., methodologies for conducting consulting projects). In contrast, *embedded knowledge* is context-dependent and narrowly applicable. Therefore, he concluded that *rationalized knowledge* is more likely to be easily shared among individuals if compared with *embedded knowledge*. The value of knowledge also has a significant impact on whether and how individuals share it (WEISS, 1999). When individuals perceive the knowledge they possess as a valuable commodity, knowledge sharing becomes a process mediated by decisions about what knowledge to share, when to

share, and who to share it with (ANDREWS; DELAHAYE, 2000).

*Motivational factors* that influence knowledge sharing between individuals can be divided into *internal factors*, which include the perceived power attached to the knowledge and the reciprocity that results from sharing; and *external factors*, which include relationships with the recipient and rewards for sharing. For example, the relationship between knowledge sharing and incentives has been supported by studies (e.g., (GUPTA; GOVINDARAJAN, 2000) finding that significant changes had to be made in the incentive system to encourage individuals to share their knowledge).

*Opportunities to share* knowledge in organizations can be both formal and informal (IPE, 2003). Formal opportunities include training programs, and structured work teams, among others. On the other hand, informal opportunities include personal relationships and social networks that facilitate learning and the sharing of knowledge. Although formal opportunities play an important role in facilitating knowledge sharing, research indicates that most knowledge is shared in informal settings — through the relational learning channels (JONES; JORDAN, 1998; PAN; SCARBROUGH, 1999; TRURAN, 1998).

The factors above are important for understanding how knowledge is shared between individuals but these factors are influenced by the *culture of the work environment* (IPE, 2003). Long e Fahey (2000) identified certain aspects of organizational culture that influence knowledge sharing—culture shapes assumptions about which knowledge is important, it controls the relationships between the different levels of knowledge (organizational, group, and individual), and it creates the context for social interaction. Culture suggests what to do and what not to do regarding knowledge processing and communication in organizations (DAVENPORT; PRUSAK, 1997).

## 2.2 Expert Finding Systems

Even though there is a huge volume of data available for solving problems, people still seek the services and guidance of an expert. The *expertise seekers* are those who seek knowledge for specific purposes such as problem-solving.

Computer systems that support the process of finding the right expert for a given problem are becoming more feasible, due to the widespread adoption of technology in organizations coupled with the massive amounts of online data available within the organization (BALOG; AZZOPARDI; RIJKE, 2009). The organization's internal and external websites, email, database records, and chats, among others, are all sources of information connecting employees and topics within the organization. These connections can be used to build representations of the candidates' expertise areas to support finding experts. Information retrieval techniques have been used to facilitate the retrieval of appropriate experts who can answer their questions. Such techniques are called **expert**

**finding systems** (EFS) - also known as expert locating systems or expertise retrieval (BALOG; AZZOPARDI; RIJKE, 2009).

According to Husain et al. (2019), expert finding usually involves activities that can be classified into the following five topics:

- i) **Expertise evidence selection:** this involves selecting the expertise-related data and information necessary for discovering experts.
- ii) **Expert representation:** the goal is not only to locate experts but also to assist users in deciding and selecting among relevant experts.
- iii) **Model building:** this consists of pre-processing, indexing, and modeling. Pre-processing and indexing focus on processing available data and identifying references to candidate experts across heterogeneous sources. Modeling and retrieval involve building models that associate candidate experts with queries provided by the expertise seeker.
- iv) **Model evaluation:** this step is responsible for evaluating the accuracy of the list of experts provided by the expert finding system.
- v) **Interaction design:** this concerns the presentation of expert search results to users. A simple list of names is often insufficient, as it may not provide enough information for the expertise seeker to evaluate the relevance of potential experts. Therefore, result pages typically include additional personal data and relevant artifacts that provide evidence of expertise.

For example, an approach is proposed by Menaha e Jayanthi (2024) for finding experts from the community question-answering website Stack Overflow<sup>1</sup> using an exact string-matching algorithm with domain knowledge. The expert profiles are processed from Stack Overflow and represented using a matrix that correlates experts and tags - keywords or labels used to categorize the questions. Based on this data, for each tag, a list of experts with their score in that particular tag is provided. Other approaches are presented in Chapter 3.

## 2.3 Ontologies

Ontologies have been recognized as conceptual tools of great importance in Computer Science since the end of the 1960s, mainly in areas such as Data Modeling (conceptual modeling) and Artificial Intelligence (DAVIS, 1997; MEALY, 1967). They can be used to assign semantics to information items, supporting data and knowledge management based

---

<sup>1</sup> <https://stackoverflow.com/>

on a common and shared understanding and representation (TAHER; VAHDATIKHAKI; HAMMAD, 2022).

In the last two decades, the interest in ontologies has increased in various segments of Computer Science. This has been motivated by the recognition of the importance of the use of ontologies in semantic tasks (e.g., communication, learning, system integration, data integration, and development of knowledge-based systems), in general, and by its role in Semantic Web development, in particular (REGINATO et al., 2019).

An ontology is a formal and explicit specification of a shared conceptualization (STUDER; BENJAMINS; FENSEL, 1998). *Conceptualization* refers to an abstract model of some real world phenomenon that identifies its relevant concepts, *explicit* means that the types of concepts used and the constraints imposed on their use are explicitly defined, *formal* refers to the fact that an ontology should be machine-readable, and *shared* reflects that ontologies must capture consensual knowledge accepted by a community (STUDER; BENJAMINS; FENSEL, 1998). Every knowledge base, knowledge-based system, or knowledge level agent is committed, either explicitly or implicitly, with one conceptualization (STUDER; BENJAMINS; FENSEL, 1998).

Ontologies can be organized in a three-layered architecture that discriminates among foundational ontologies, core ontologies, and domain ontologies (SCHERP et al., 2011). **Foundational ontologies** aim at modeling the very basic and general concepts and relations that make up the world (e.g., objects, events, participation, and parthood). They are generic across any area and are highly reusable in different modeling scenarios. **Core ontologies** provide a refinement to foundational ontologies by adding detailed concepts and relations in a specific area (such as service, process, organizational structure) that still spans across various domains. **Domain ontologies** concern a particular domain in reality, such as a domain-specific medical ontology describing the anatomy of the human body. Domain ontologies can make use of or be based on foundational ontologies and core ontologies by the specialization of their concepts.

Falbo et al. (2013) argue that Scherp, Saathoff, Franz, & Staab (SCHERP et al., 2011) classification should be perceived as a continuum, ranging from pure foundational ontologies (that are totally domain-independent) to domain ontologies (domain-dependent). Thus, there can be different levels of generality in ontologies classified in a certain type. In this sense, considering the continuous nature of the aforementioned classification, some ontologies can be used to support the development of more specific ontologies even within the same level of generality. For instance, there are more general core ontologies, such as UFO-S (NARDI; FALBO; ALMEIDA, 2013b), which addresses services in general, and more specific core ontologies, such as the Software Process Ontology (BRINGUENTE; FALBO; GUIZZARDI, 2011), which addresses core concepts about software and system in the Software Engineering area and spans across several sub-domains in that area, such

as software measurement, software design process, software test process and so on.

Figure 4 illustrates the view of ontology generality levels as a continuum. The dashed arrows show the grounding dependencies between the ontologies at different levels.

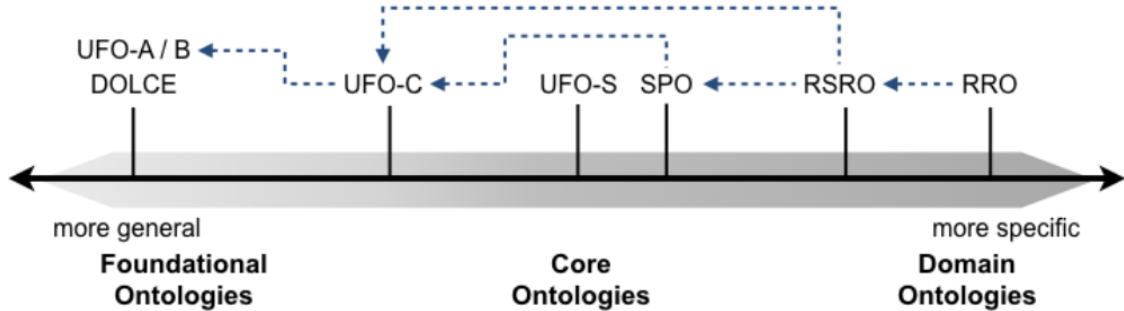


Figure 4 – Ontology generality levels as a continuum (adapted from (FALBO et al., 2013; RUY, 2017))

In the figure, DOLCE (BORGIO; MASOLO, 2009; GANGEMI et al., 2002) and the Unified Foundational Ontology - UFO (GUIZZARDI, 2005) are foundational ontologies. UFO-C (an ontology of social entities) is still considered to be a foundational ontology despite being more specific than UFO-A (an ontology of endurants) (GUIZZARDI, 2005; GUIZZARDI et al., 2018) and UFO-B (an ontology of events) (GUIZZARDI; FALBO; GUIZZARDI, 2008; GUIZZARDI et al., 2013). The *Reference Software Requirements Ontology* (RSRO) (RUY et al., 2016) and *Runtime Requirements Ontology* (RRO) (DUARTE et al., 2016) are domain ontologies, although RRO addresses an even more specific domain than RSRO.

Another important distinction differentiates ontologies as conceptual models, called *reference ontologies*, from ontologies as computational artifacts, called *operational ontologies* (GUIZZARDI, 2007). A reference ontology is constructed with the goal of making the best possible description of the domain in reality, representing a model of consensus within a community, regardless of its computational properties. Operational ontologies, in turn, are designed with the focus on guaranteeing desirable computational properties and, thus, are machine-readable ontologies.

### 2.3.1 Unified Foundational Ontology (UFO) and OntoUML

In this section, we introduce UFO and OntoUML, the foundational ontology and the language used to develop Core-O, the ontology used in this work (it will be presented in Section 2.3.2).

UFO is a foundational ontology that has been developed based on several theories from Formal Ontology, Philosophical Logics, Philosophy of Language, Linguistics, and Cognitive Psychology (GUIZZARDI, 2005). UFO is composed of three main parts: UFO-A,

an ontology of durants (GUIZZARDI, 2005); UFO-B, an ontology of perdurants/events (GUIZZARDI et al., 2015); and UFO-C, an ontology of social entities (both durants and perdurants) built on top of UFO-A and UFO-B (GUIZZARDI; FALBO; GUIZZARDI, 2008).

UFO presents a fundamental distinction between the ontological categories of *Types* and *Individuals*. Types are patterns of features that are repeatable across different entities (e.g., Person) (GUIZZARDI et al., 2022). Individuals are particular entities that cannot be instantiated (e.g., the person Mary). More details about UFO and its applications are available in (GUIZZARDI, 2005; GUIZZARDI; FALBO; GUIZZARDI, 2008; GUIZZARDI et al., 2015; GUIZZARDI et al., 2022).

An important application of UFO is OntoUML (GUIZZARDI, 2005), an ontologically well-founded language for ontology-driven conceptual modeling built as a UML extension based on UFO. Therefore, OntoUML can be used to create class diagrams representing ontologies grounded in UFO - i.e., well-founded reference ontologies. In OntoUML, the UFO classifications are represented as stereotypes, indicating how each concept is classified according to UFO. By using OntoUML, validation rules can be applied to the created model to check consistency and correctness based on UFO conceptualization. We now present the OntoUML stereotypes relevant to this work in Table 1 - class stereotypes are presented in *italic*, and relationship stereotypes are presented in **bold**.

### 2.3.2 Core-O: A Competence Reference Ontology

Core-O is a domain ontology that deals with concepts related to competence such as knowledge, skills, and attitudes, among others (CALHAU et al., 2024). It was created using OntoUML and is formed by two main sub-ontologies: the *Personal Competence Ontology* and the *Competence Type Ontology*. The first focuses on individuals, and the second focuses on the universal modeling of competencies that are not specific to an individual.

Figure 5 shows a fragment of the *Personal Competence Ontology*, containing concepts relevant to this work (the others were omitted). In Core-O conceptual models, the concept stereotype represents its classification according to UFO. In the model description, Core-O concepts are written in **bold**.

A **Human Aspect** refers to a category that includes all intrinsic qualities, characteristics, or experiences that are inherently part of a **Person** and can be evidenced by some **Evidence** - a category that includes certificates, qualifications, project participation, credentials, and other experiences (e.g., John’s Java coding skill is evidenced by his Java course certificate).

**Knowledge** and **Human Capability** are two subtypes of **Human Aspect** defined

Stereotype	Definition
<i>«kind»</i>	It represents a class that supplies a principle of identity for its instances (GUIZZARDI, 2005). The instances are rigid because they will have the same type in every possible world. For example, an instance of Person is necessarily an instance of Person in every possible world.
<i>«category»</i>	It is used to aggregate essential properties to individuals which following different identity principles (GUIZZARDI, 2005). For example, the category Furniture aggregates essential properties of Table and Chair.
<i>«roleMixin»</i>	It aggregates properties which are common to different roles (GUIZZARDI, 2005). For example, a roleMixin Customer aggregates common properties of Private Customer and Corporate Customer.
<i>«event»</i>	It identifies those classes whose instances are events (past occurrences) (ALMEIDA; FALBO; GUIZZARDI, 2019).
<i>«mixin»</i>	It represents properties that are essential to some of its instances and accidental to others (semi-rigidity). An example is the mixin Seatable, which represents a property that can be considered essential to the kinds Chair and Stool, but accidental to Crate, Paper Box or Rock.
<b>«characterization»</b>	It is a relation between a bearer type and its feature. Feature is intrinsic (inherent) moment of its bearer type, and thus existentially dependent on the bearer.
<b>«creation»</b>	A special kind of participation referring to the creation of an endurant. If an endurant is related to an event through an association stereotyped «creation» then that endurant is created in that event.
<b>«manifestation»</b>	It refers to a type of relational property derivation that connects a moment (e.g., a relator or intrinsic mode) to the entity in which it inheres or depends. It represents the ontological dependence of a moment on its bearer.
<b>«participation»</b>	It is used in order to model the participation of endurants in events. An association stereotyped «participation» always relates a class stereotyped with «event» with a class denoting an endurant type. If an endurant and an event are linked through a «participation» association, then, either: (i) the event is a manifestation of a disposition of the participating endurant, or (ii) the event is composed of such a manifestation.

Table 1 – OntoUML stereotypes relevant to this work

in Core-O. A **Human Capability** encompasses all human abilities and is manifested through a **Human Task** - an event related to a unit of work or result. These results include **Artifacts** created or changed by the task: the **Task Outputs**. On the other hand, **Knowledge** is typically associated with internal representations of facts, principles, or theories in a specific domain. A **Skill** is a subtype of **Human Capability** and alludes to

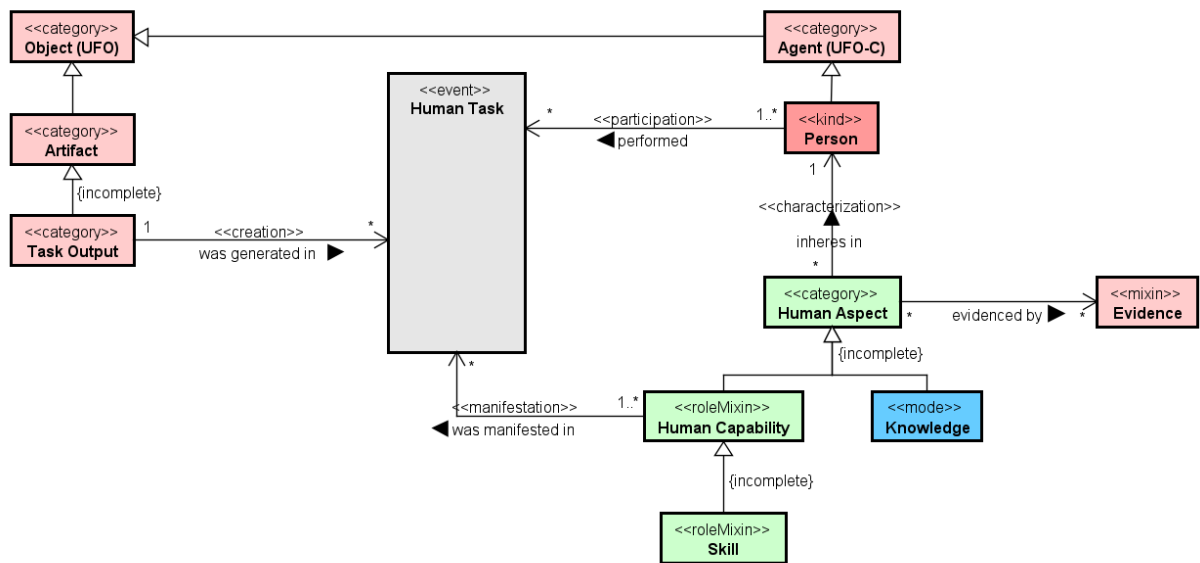


Figure 5 – Fragment of Core-O Personal Competence Ontology (adapted from (CALHAU et al., 2024))

the capability to perform actions. In other words, a **Skill** relates to the ability to apply knowledge to perform a **Human Task**.

In Figure 6, the fragment of the *Competence Type Ontology* relevant to this work is presented. This sub-ontology addresses concepts similar to the ones previously presented. However, it focuses on types (universals). In other words, it describes the types of task, artifact, knowledge, and skill. More details about Core-O can be found in (CALHAU et al., 2024).

## 2.4 Concluding Remarks

This chapter addressed the main background for this work. First, we presented the main notions of KM. After that, we discussed the importance of knowledge in the software development context and introduced the concept of expert-finding systems. Then, we presented basic notions of ontologies and introduced UFO and OntoUML, which were used to develop Core-O, an ontology about competencies that was used in this work and introduced in this chapter.

To better understand how experts have been identified to share knowledge in the software development context, we performed a mapping study, which is presented in the next chapter.

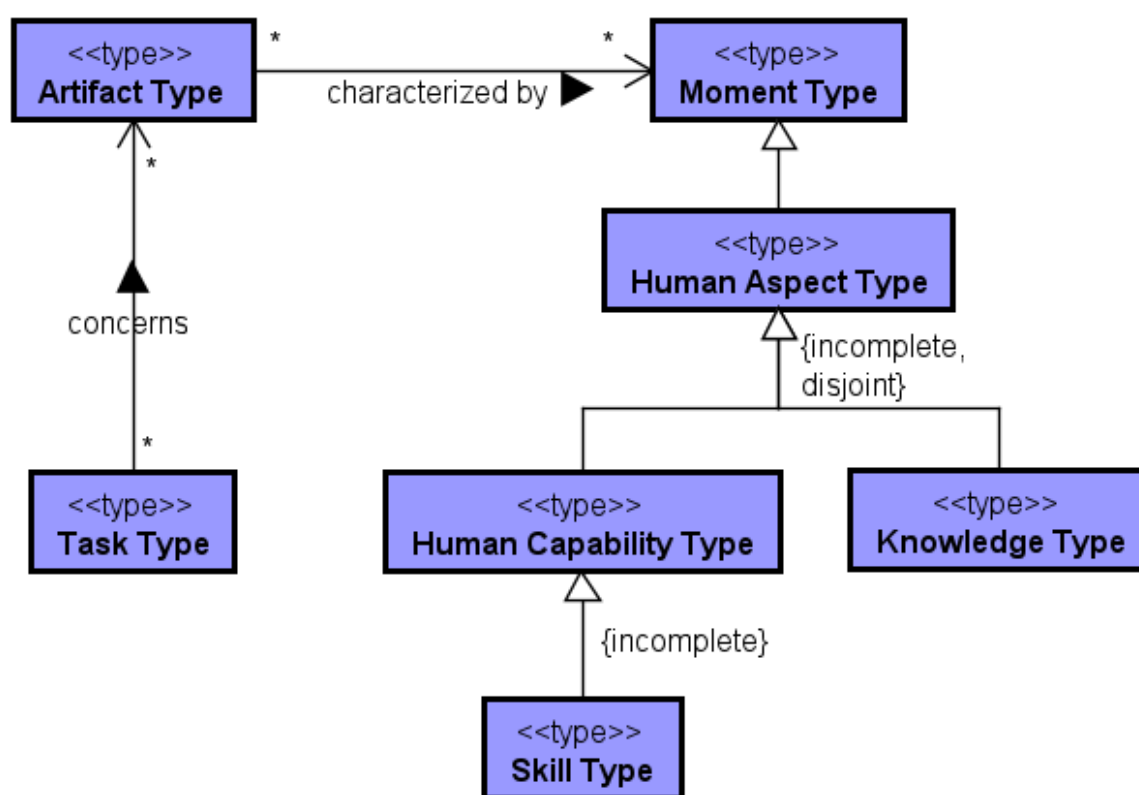


Figure 6 – Fragment of the Core-O Competence Type Ontology (adapted from (CALHAU et al., 2024))

## 3 A Mapping Study about Expert Identification in Software Development

*This chapter presents a mapping study that investigated approaches for identifying experts to share knowledge in the software development context and presents its main results. The study was performed to help us investigate and understand the research problem. Therefore, it is related to the Relevance cycle (Figure 1). Additionally, it is related to the Rigor cycle because it adds knowledge generated by this research to contribute to the knowledge base growth. The study was published in (BRAGA; JR; BARCELLOS, 2023). The chapter is organized as follows: Section 3.1 introduces the chapter; Section 3.2 addresses the research protocol; Section 3.3 summarizes the obtained results; Section 3.4 discusses the results; Section 3.5 presents some of the limitations of the study; and Section 3.6 presents the chapter concluding remarks.*

### 3.1 Introduction

Considering the important role that experts play in software development, we decided to investigate approaches that help identify experts to share knowledge in the software development context. 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. (2019), which investigated expert finding systems in general (i.e., regardless of the application domain). Other works, such as (LIN et al., 2017) and (BALOG et al., 2012), 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 refer to 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.

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 (PETERSEN; VAKKALANKA; KUZNIARZ, 2015). The panorama provided by a mapping study allows identifying issues in the research topic that could be addressed in future research.

We followed the process defined in (KITCHENHAM; CHARTERS, 2007), 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;
- *Reporting*: when the results are recorded and made available to potentially interested parties.

## 3.2 Research Protocol

The study **goal** was to investigate approaches that help identify suitable experts to share knowledge in the software development context. To achieve this goal, we defined the **research questions** presented in Table 2.

Table 2 – 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 (WIERINGA et al., 2006). 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.

Continued on next page

**Table 2 – continued from previous page**

ID	Research Questions	Rationale
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 (STANDARDIZATION et al., 2017) 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 there has been a concern with semantics by verifying whether the approaches have been grounded in any formal conceptualization and the purpose of using it.

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 ((NORAMBUENA; BERGEL, 2021; SHAMI et al., 2007)) 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 an 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 (KITCHENHAM; CHARTERS, 2007), 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<sup>1</sup> 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 mapping was conducted by three researchers. The author of this work performed publication selection and data extraction. A second researcher, a doctoral student, 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

---

<sup>1</sup> <https://bit.ly/3bW3Mo6>

statistical analysis. The third researcher (the supervisor of this research) reviewed the results. Discordances were discussed and resolved. Finally, the three researchers performed qualitative analysis considering the findings, their relation to the research questions, and the study purpose.

### 3.3 Data Extraction and Synthesis

Searches were conducted for the last time in March 2023 and the study considered papers published until 2022. The process followed and the number of publications selected in each step are presented in Figure 7.

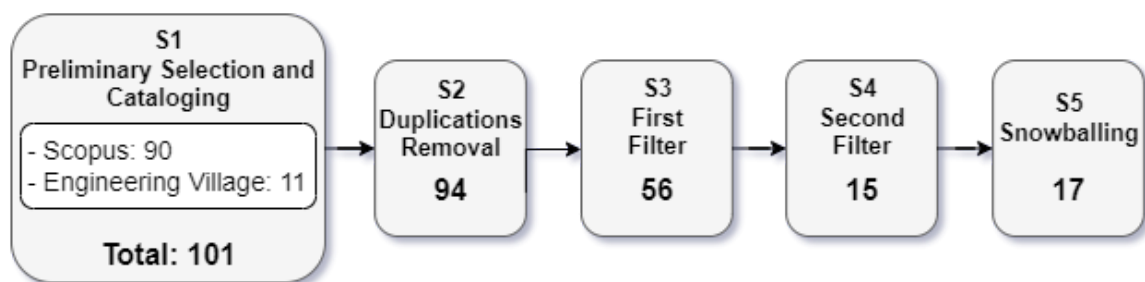


Figure 7 – Publications selection

In the 1<sup>st</sup> 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 2<sup>nd</sup> step (S2), 7 duplicated papers were removed which represents a reduction of approximately 6.9% from the initial number of papers. In the 3<sup>rd</sup> 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 4<sup>th</sup> 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 3<sup>rd</sup> step, and a total reduction of approximately 85% from the initial quantity of papers. Finally, in the 5<sup>th</sup> 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 the study package (BRAGA; JUNIOR; BARCELLOS, 2023).

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, we did not select publications presenting expert identification methods not used in software development or not mentioning knowledge sharing 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 3. In the following, we present the data synthesis for each research question.

Table 3 – 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).	( <a href="#">NORAMBUENA; BERGEL, 2021</a> )
#2	Presents an approach that uses data from Questions & Answers (Q&A) websites to profile users and recommends experts for answering a question.	( <a href="#">HUANG et al., 2020</a> )
#3	Uses knowledge models to help identify which developers (expert candidates) have more knowledge of specific elements of software projects.	( <a href="#">LUCAS et al., 2020</a> )
#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.	( <a href="#">SCHETTINO et al., 2019</a> )
#5	Proposes a mobile application to support expert locations for global software development environments.	( <a href="#">CORDOVA-MORAS; RODRIGUEZ-ELIAS; SERNA-ENCINAS, 2017</a> )
#6	Proposes an expert recommendation method based on knowledge embedding, which recommends appropriate experts for developers.	( <a href="#">FU et al., 2017</a> )
#7	Presents a framework for identifying experts by eliciting developers' expertise in software components using complexity analysis of source code.	( <a href="#">TEUSNER; MATTHIES; GIESE, 2017</a> )
#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.	( <a href="#">MORALES-RAMIREZ et al., 2015</a> )

Continued on next page

**Table 3 – continued from previous page**

ID	Brief Description	Ref.
#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.	(ALLAHO; LEE, 2014)
#10	Introduces a degree-of-knowledge model that identifies experts by computing developer’s knowledge based on developer’s authorship and interaction data.	(FRITZ et al., 2014)
#11	Uses the developers’ mailing list and source code history to support finding experts during coding.	(MORAES et al., 2010)
#12	Presents an approach that identifies experts based on the interaction of individuals with software artifacts.	(JANES; SIL-LITTI; SUCCI, 2008)
#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.	(SHAMI et al., 2007)
#14	Proposes an approach to build expert finder systems, supporting the adaptation of the search strategy according to the organization’s needs.	(HUGHES; CROWDER, 2003)
#15	Presents an approach for identifying experts by using sensitive data and preserving privacy.	(ADAR et al., 2003)
#16	Proposes a tool that allows identifying the most knowledgeable person related to a tool, language, and release version.	(MOCKUS; HERBSLEB, 2002)
#17	Proposes an approach that uses social structures on GitHub (following-followed, watching, and collaboration networks) and programming behavior for identifying experts.	(MO et al., 2015)

**Publication year and type (RQ1):** Figure 8 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 4 shows the number of publications considering their research type according to the classification proposed in (WIERINGA et al., 2006). 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

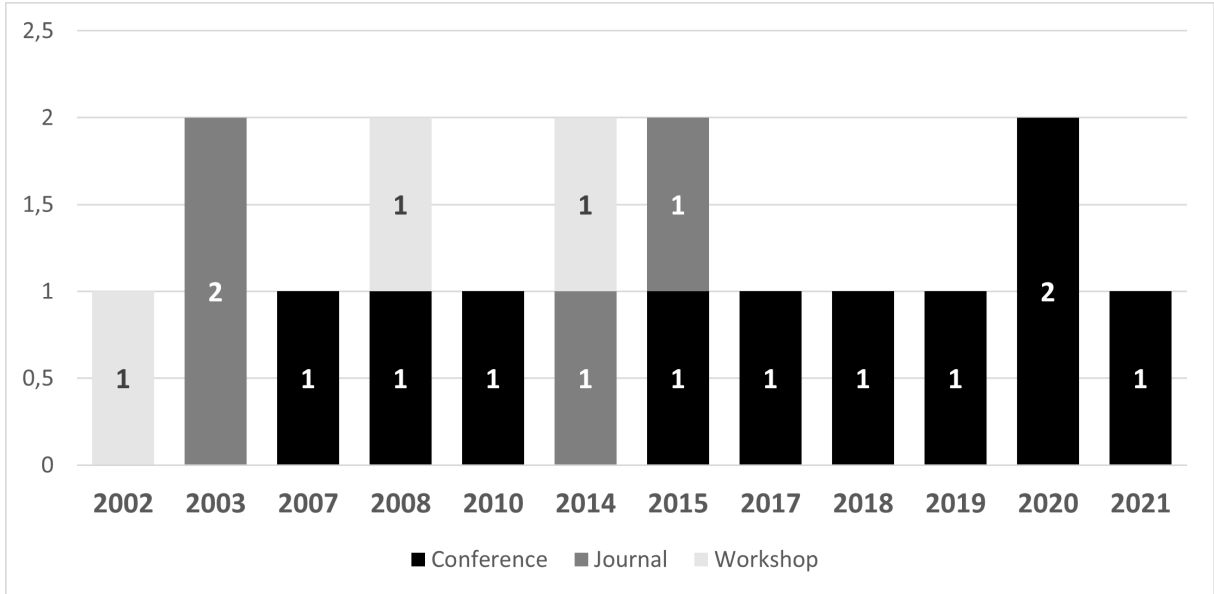


Figure 8 – Publication year and vehicle

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*).

Table 4 – 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%

**Sources for identifying experts (RQ3):** We identified eight sources of evidence used to identify experts, as shown in Figure 9. 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).

**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

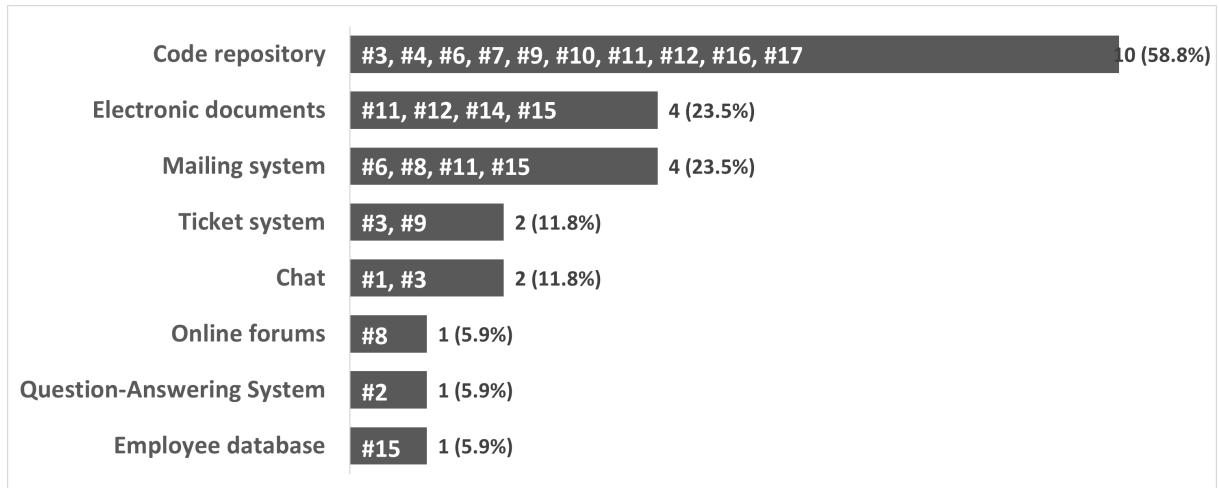


Figure 9 – Sources for identifying experts

not make clear whether there is automated support. Table 5 summarizes the publications considering the automated support aspect.

Table 5 – 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%

**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 6 summarizes these results.

Table 6 – 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%

**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’ needs, while nine publications (52.9%) do not make explicit

the supported roles. Some publications mention more than one supported role. Table 7 indicates the roles supported by each publication.

Table 7 – 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 8 presents the processes/activities (based on (STANDARDIZATION et al., 2017)) 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 8 is higher than the number of investigated approaches because some of them are related to more than one activity.

Table 8 – 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 9 summarizes these results.

Table 9 – 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%

### 3.4 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) (CAMPBELL et al., 2003). 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 explored further. 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) to share 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 who 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 (KRüGER et al., 2018)). 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 (ISRAILIDIS; SIACHOU; KELLY, 2020) 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% 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 (WACHE et al., 2002). Neglecting these semantic conflicts can lead to solutions that fail in achieving their purposes (POKRAEV, 2009). 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 (NARDI; FALBO; ALMEIDA, 2013a). 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.

### 3.5 Threats to Validity

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 (WOHLIN et al., 2012). 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 author of this work. A second researcher (a doctoral student) 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 (LANDIS; KOCH, 1977) and obtained the value 0.8, which, according to Landis e Koch (1977), 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 (WIERINGA et al., 2006) and software process activities (STANDARDIZATION et al., 2017)). 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, the

author of this work together with a doctoral student represented the results using charts and tables, and interpretation was performed by them with the participation of a third researcher (the supervisor of this research) iteratively, considering the research questions. In this way, complementary interpretations were combined and different interpretations were discussed.

### 3.6 Concluding Remarks

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 chapter, 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 to 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 their effectiveness in supporting different roles and processes. Also, knowledge needs of specific roles and processes could be identified 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 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.

# 4 An Ontology-based Approach to Support the Development of Systems to Identify Who Knows What in Software Organizations

*This chapter presents iKnow, an ontology-based approach to support the development of systems to identify who knows what in software organizations. Section 4.1 presents the chapter introduction. Section 4.2 presents iKnow. Section 4.3 discusses related work. Finally, Section 4.4 presents the chapter's concluding remarks. In the DSR context, this chapter is related to the Design cycle (Figure 1) as it presents the main produced artifact.*

## 4.1 Introduction

The literature investigation about expert identification to share knowledge in the software development context presented in Chapter 3 revealed a lack of concern with semantics or a common conceptualization when designing systems for identifying who knows what. This can be problematic because semantic conflicts may happen and lead to the incomplete or incorrect use of data to identify experts, producing less effective results. The use of ontologies can help to address this challenge, by providing a formal and explicit specification of a shared conceptualization (STUDER; BENJAMINS; FENSEL, 1998).

Software organizations generate large amounts of data regarding task execution, produced software artifacts, and the individuals who performed the tasks and created or modified the software artifacts. By uncovering the relations between task types and the corresponding skill types, and by enriching existing data with the appropriate semantics, we can infer the expertise of individuals within the organization based on the tasks they have performed and artifacts they have produced. For example, if Develop Class Diagram is a task type related to the skill type Developing Structural Conceptual Model and John has performed several develop class diagram tasks and produced several class diagrams, we can consider these tasks and artifacts as manifestations of the skill type Developing Structural Conceptual Model and, thus, conclude that John has the developing structural conceptual model skill. Core-O (CALHAU et al., 2024) enhances our understanding of these concepts and their interconnections, contributing to the identification of these relationships and the enrichment of existing data. Core-O was selected because it addresses the concepts necessary in this work and also because it was developed by members of NEMO, the

research group where this work was carried out, facilitating direct communication with the authors to discuss key aspects of its application.

Individuals can manifest their skills through the execution of human tasks, which produce artifacts as a task output. Considering that skills refer to the ability to apply knowledge to perform tasks in an application domain, the execution of tasks serves as evidence of a person’s skill and indicates their knowledge in the domain.

In Figure 10, we present the Core-O extract relevant to this dissertation. It was extracted from the Core-O conceptual model shown in Chapter 2. For simplification reasons, UFO concepts, concepts not directly used in *iKnow* and the general concepts that specialize *Task Output* and *Skill* were omitted. In the figure, concepts referring to types are shown in purple and concepts referring to individuals (i.e., particulars) are represented in green.

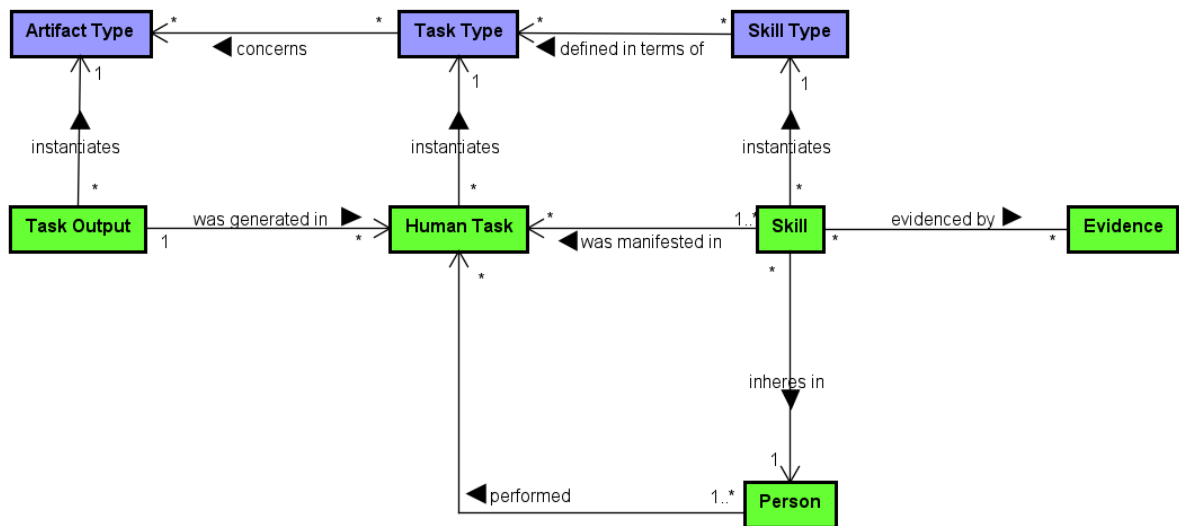


Figure 10 – Core-O extract used in *iKnow*.

As presented in Section 2.3.2, the concepts referring to types (in purple in Figure 10) are not specific to an individual. For example, when referring to the task type Programming Test Application, we mean the general category of task rather than instances of execution of tasks of that type. On the other hand, the concepts referring to individuals (in green in Figure 10) handle specific occurrences of skills, artifacts, and tasks. For example, the execution of the task programming tests of application X (which is a task of the type Programming Test Application) by John on October 22, 2024, at 11:30 a.m.. For clarity purposes, hereafter we adopt the convention presented in Table 10 when mentioning examples of different concepts in the text.

The mapping study also revealed that most proposals (e.g., (LUCAS et al., 2020; MENAHA; JAYANTHI, 2024)) have focused on techniques for identifying experts in a particular skill type based on specific sources (e.g., code base), without evaluating the

Concept	Description	Example
Skill Type	Represented in the gerund form and formatted with underline and italic.	<u><i>Documenting software</i></u>
Task Type	Represented by verbs in the base form and formatted with underline and italic.	<u><i>Document software</i></u>
Artifact Type	Represented by nouns and formatted with underline and italic.	<u><i>Software specification</i></u>
Skill	Represented in the gerund form and formatted with underline and bold.	<b><u>Documenting software</u></b>
Task	Represented by verbs in the base form and formatted with underline and bold.	<b><u>Document software</u></b>
Task Output	Represented by nouns and formatted with underline and bold.	<b><u>Software specification</u></b>

Table 10 – Naming convention

relevance of this skill type to the organization’s needs. We argue that a more comprehensive perspective is necessary to align the organization’s knowledge needs with existing data to identify experts to share knowledge.

In addition, the results revealed a lack of concern with factors other than possessing the desired knowledge in expert identification approaches. This represents an important gap, especially when the goal is to support knowledge-sharing practices within organizations since non-technical factors such as availability should be considered to identify suitable experts.

Therefore, we propose in this chapter an approach based on the presented extract of Core-O (CALHAU et al., 2024), focused on addressing practical needs for developing a system and handling non-technical factors that can influence knowledge sharing.

## 4.2 iKnow: An Ontology-based Approach to Support the Development of Systems to Identify Who Knows What in Software Organizations

The proposed approach leverages the conceptualization defined by the Core-O (CALHAU et al., 2024) to identify who knows what based on performed tasks and produced task outputs (artifacts) that are interpreted as skill manifestations. The *iKnow* approach comprises eight steps. An overview of the approach is presented in Figure 11.

In a nutshell, during the first two steps (*i* and *ii*), our focus is to identify the

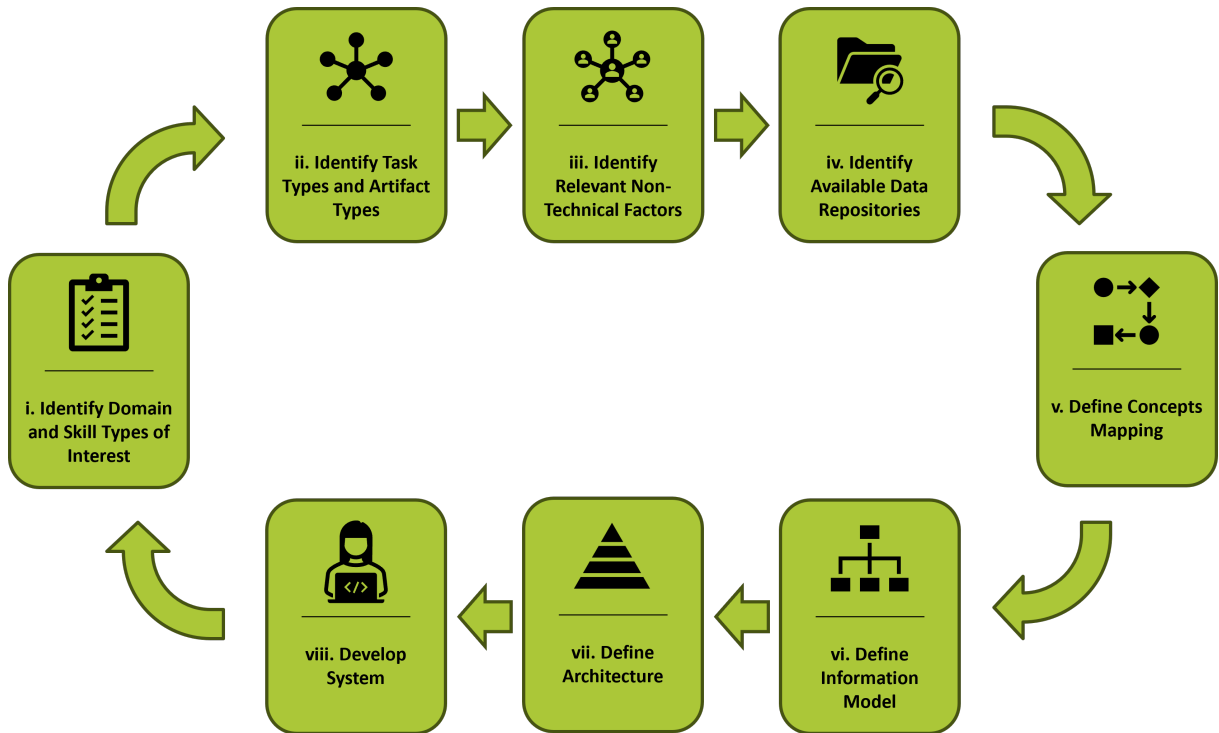


Figure 11 – iKnow steps

domain of application and relevant skill types for the organization, and, thus, identify which task types are related to the identified skill types and the artifact types related to the identified task types. The identified types are utilized in subsequent steps to support the identification of instances of those types in data available in the organization. In the next step (*iii*), considering that many aspects influence knowledge-sharing effectiveness in organizations, the goal is to identify factors (besides having the desired knowledge) that influence the selection of an individual who can share knowledge. These factors are called non-technical factors in this work.

The next three steps (*iv*, *v*, and *vi*) address the identification of tools and data repositories within the organization that contain data related to the previously identified concepts, the assigning of semantics to data by mapping it to the Core-O conceptualization, and the creation of the information model representing the structural conceptual model of the system to be developed.

The next step (*vii*) addresses the definition of the system architecture and presents a proposed reference architecture that can be used. Finally, the last step (*viii*) is related to the development of the system considering the rationale and artifacts produced by the previous steps.

Therefore, as presented in Section 2.2, expert finding usually involves activities that can be classified into the following five topics (HUSAIN et al., 2019): expertise evidence selection; expert representation; model building; model evaluation; and interaction design. In Table 11, we present the correlation between the defined steps and these five topics.

Steps / Expert Finding Topics	i	ii	iii	iv	v	vi	vii	viii
Expertise Evidence Selection	X	X		X	X			
Expert Representation			X					X
Model Building					X	X	X	X
Model Evaluation		X						X
Interaction Design						X	X	X

Table 11 – Correlation between *iKnow* steps and expert-finding topics defined in (HUSAIN et al., 2019)

For simplification, the steps are presented as sequential ones. However, it must be noted that it is possible to have interactions among them. Additionally, organizations are quite dynamic: new expertise may become valuable, and new tools providing different data can be adopted, among other changes that may impact the defined expert finding system. For this reason, the approach is defined as a cycle that allows for the evolution of the system.

It is important to notice that the systems developed by following *iKnow* depend on data to identify who knows what in the organization. Thus, *iKnow* should be used only if the organization collects data during software process executions and there are available data sources.

In the following, we describe each step of *iKnow*.

(i) Identify Domain and Skill Types of Interest

In this step, the goal is to identify the domain to which the needed knowledge is related (i.e., the domain that will be addressed by the expert finding system to be produced) and the skill types that will be considered. According to the organizational needs, the domain can be as general as *Software Engineering* or as narrow as *Java Software Development* or *Unit Testing*. This definition is important since it establishes the possible range of skill types of interest. Once the domain is defined, the skill types to be considered must be identified. Existing standards, such as the Occupation Information Network (O\*Net<sup>1</sup>) and the European Skills, Competences, Qualifications and Occupations (ESCO<sup>2</sup>), provide a list of skill types related to job positions. These standards can serve as a reference catalog for selecting the skill types relevant for the organization and that must be addressed by an expert finding system.

O\*Net, for instance, lists 66 skill types related to the job title Software Developer, including *Programming testing software*, *Web platform development software*, among others

<sup>1</sup> <https://www.onetonline.org/>

<sup>2</sup> <https://esco.ec.europa.eu/>

- additional examples are presented in Figure 12. O\*Net also list technologies related to these skills which may serve as source of definition of finer-grained skill types. For example, *Programming testing software* skill type is related to technologies *JUnit* and *Selenium* and, therefore, it is possible that the skill type of interest in an organization is the *Programming testing software using JUnit* instead of the broader, technology-independent skill type.

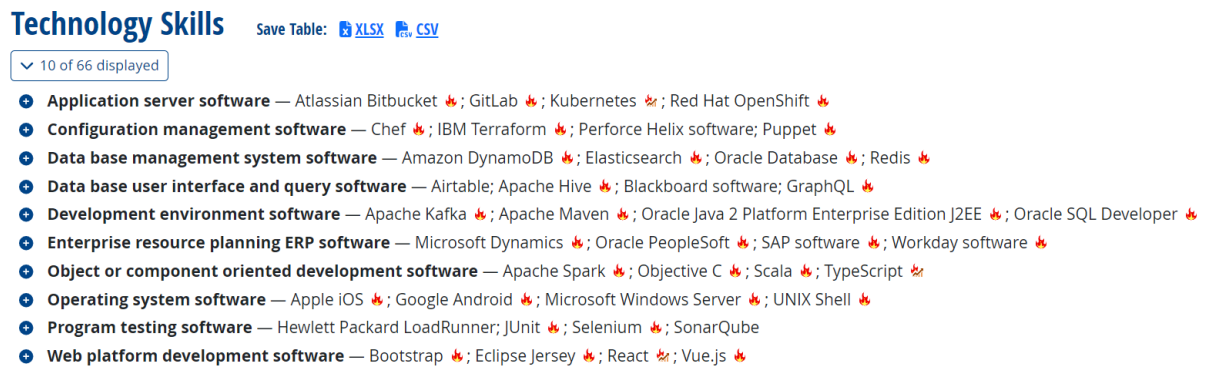


Figure 12 – O\*Net skill types summary for Software Developer

Another valuable source for supporting this step is the existing job title descriptions within the organization, as they may contain occupation-specific information, such as required task types, skill types, and other relevant details. Even though these descriptions can be helpful, it is also important to involve individuals occupying key roles related to the defined domain in the organization. This is important because one skill type may be related to a role in the context, but if the related knowledge is widespread throughout the organization, it may become less valuable. Furthermore, other skill types or more specialized ones may become valuable for the organization and, therefore, addressed by the designed system.

Additionally, it is important to consider the existence of data regarding the defined skill types in the organization. In other words, we should avoid defining skill types of interest that do not possess data to support the identification of experts, or are too abstract and difficult to be related to existing data (e.g., *Solving problems*).

*Example:* To exemplify this and the following steps, suppose that the Project Management Office Manager of *Org*, a large multinational software organization, wants an expert finding system to help software project managers find experts to share knowledge to help them perform their tasks. In this case, the domain of interest would be *Project Management*, and examples of possible skill types of interest could be *Developing a Project Schedule* and *Creating a Project Budget*.

## (ii) Identify Task Types and Artifact Types

This step aims at defining the task types through which the skill types identified in the previous step are manifested and, also, the artifact types produced by these task types. As occurred in the previous step, it is important to involve individuals occupying key roles in the organization. By identifying which task types are linked to the skill types of interest, the system will be able to trace the skill manifestations in the organization and provide a list of expert candidates. It is possible that a skill type is manifested through more than one task type. For example, the *Coding in Java* skill type could be manifested by the task type *Commit changes in Java files* or by the task *Deliver a Java training*.

It is also important to identify the related artifact types, because sometimes the instance of the task type may not be explicit in existing data in the organization. Therefore, the instance of the artifact type can serve as a proxy for the task type that manifests the given skill type. For example, consider the skill type *Programming automated tests*, which is typically manifested through the task type *Write automated test scripts*. The occurrence of this task might not always be explicitly defined in task management tools. In these cases, data regarding artifacts of the artifact type (e.g., *test scripts*) in version control systems provide evidence of the occurrence of instances of the defined task type.

Finally, it is crucial to ensure that the identified task types and artifact types are indeed considered by members of the organization as manifestations of the identified skill types of interest. Thus, the proposed set of skill types, task types, and artifact types should be evaluated (e.g., in meetings or through the application of a survey).

*Example:* Consider that in *Org*, project managers use several tools (e.g., MS Project, MS Excel, Trello, ClockFy, GitLab) to support management tasks and produce project management-related artifacts. Some of them could be related to the skill types previously identified, such as:

- The task type *Develop MS Project Schedule* and the artifact type *MS Project Schedule* could be manifestations of the skill type *Developing a Project Schedule*.

- The task type *Develop Project Costs Spreadsheet* and the artifact type *Project Costs Spreadsheet* could be manifestations of the skill type *Developing a Project Budget*.

## (iii) Identify Relevant Non-Technical Factors

This step aims at identifying non-technical aspects that may influence knowledge sharing, besides the possession of the desired knowledge. Considering that many factors influence knowledge sharing in organizations (IPE, 2003), and one of the key aspects of expert finding systems is to assist users in deciding and selecting among relevant experts, it is important to address which factors are relevant to this decision. In this work, we call

*non-technical factors* any other aspect related to the expert candidates that are important for effective knowledge sharing besides possessing the desired knowledge.

Examples of non-technical aspects that can affect knowledge sharing are, among others:

- *Availability*: An individual within an organization may have the desired knowledge but be unavailable for knowledge sharing (e.g., due to their involvement in high-priority projects).
- *Social bond*: An individual can feel compelled to assist those with whom they share a social bond. On the other hand, the seeker may prefer to ask for help to someone closer (or not) to them.
- *Language*: Knowledge sharing can occur only if the knowledge seeker and the expert can communicate with each other, which requires that both understand the same language.
- *Team or organizational department*: The knowledge seeker can opt for looking for experts that work in same team or organization department because these individuals are more likely to share project's goals.
- *History of effective knowledge sharing*: An individual may select experts with a history of effective knowledge sharing because they are likely to have a collaborative mindset, and a willingness to assist others.
- *Seniority*: It is important because individuals with greater experience often have a deeper understanding of the knowledge relevant to their field.
- *Years of experience*: This aspect provides a similar perspective to seniority but it is more independent of the seniority levels established within the organization.
- *Years of service in the organization*: The amount of time spent working within the organization may influence an individual's understanding of its processes and software patterns, contributing to more effective knowledge sharing.

To select relevant factors, organizational aspects must be considered. For instance, the languages that organization members can communicate may be relevant in a globally distributed scenario, but irrelevant if an organization is only based in one specific country. Furthermore, key stakeholders in the company may be consulted to select the appropriate aspects.

*Example*: *Org* is a large multinational organization. The project managers are distributed worldwide and speak different languages. Some are beginners, others seniors.

Some have a very tight schedule and are often not available to help others. Moreover, the organizational culture affects project management practices, and most of the time the nuances of such influence are in people's minds. Considering this scenario, examples of non-technical factors relevant to *Org* case would be availability, language, seniority, and years of service in *Org*.

#### (iv) Identify Available Data Repositories

This step aims at identifying the available data repositories in the organization that contains data related to the skill types of interest, and related task types and artifact types identified in the previous steps. Tools used in the organization are a good source of the desired data. Task management tools (e.g., JIRA<sup>3</sup>) tend to maintain a lot of information about performed tasks, storing data about their execution, such as the executor and time of execution among others. However, depending on the level of granularity of the defined skill types, the data present in this kind of tool may not be enough. Therefore, other tools may be necessary, such as version control system tools (e.g., Github<sup>4</sup>), cloud monitoring platforms (e.g., Datadog<sup>5</sup>), IT Service Management (ITSM) tools (e.g., ServiceNow<sup>6</sup>), and others. These tools often offer an API that can be used to access and extract data.

*Example:* Considering the results of the previous steps, the system to be developed for *Org* should identify who knows what about *Software Project Management* based on the project managers' skills in *Developing project schedule* and *budget*. For that, the system should consider as evidence of such skills the task of the types *Develop MS Project Schedule* and *Develop Project Costs Spreadsheet* performed by the project managers and the artifact of the types *MS Project Schedule* and *Project Costs Spreadsheet* produced by them. Therefore, in order to identify the project managers with such skills (and, thus, knowledge), the system must consider data related to those types of tasks and artifacts. In other words, the data repositories to be used should provide data about the tasks project managers have performed and the task outputs they have produced. Thus, from the tools previously cited (see Section 4.2), MS Project repository and MS Excel repository would provide useful data do the *Org* case.

#### (v) Define Concepts Mapping

This steps aims at assign semantics to data considering the Core-O extract presented in Section 4.1. In this step, the Core-O extract is used as a reference model to define mappings between its concepts and data available in the data repositories selected in

<sup>3</sup> <https://www.atlassian.com/software/jira>

<sup>4</sup> <https://github.com/>

<sup>5</sup> <https://www.datadoghq.com/>

<sup>6</sup> <https://www.servicenow.com/>

the previous step. These mappings ensure semantic alignment between external data sources and Core-O, facilitating data integration and interoperability. Additionally, this step explicitly documents the rationale behind the extracting and transforming activities, which facilitates the assessment of these mappings and communication among stakeholders. By making the rationale explicit, it facilitates the evolution and validation of the developed system.

To perform the semantic mappings, first it is necessary to capture the conceptual model of the data repositories to be considered. Then, the data repositories concepts equivalent to Core-O concepts must be identified. The semantic mappings will indicate which data will be used to identify who knows what and where they are stored in the repositories. Moreover, if data is stored in different repositories, the semantic mappings will indicate which data from which repositories must be integrated.

Given that the approach primarily focuses on identifying who knows what, and there are works that already address in detail how to carry out semantic mappings, we provide only minimal guidance for this step. The guidelines provided by OBA-SI (CALHAU; FALBO, 2010) can be applied to establish the mappings between the data repositories and Core-O. OBA-SI is an ontology-based approach to system integration that guides the steps - focusing on analysis and modeling at a high level of abstraction - to provide semantic agreement between different tools at the conceptual level, supporting their integration.

Example: Suppose that, in *Org*, data related to the execution of activities is recorded using a spreadsheet, as presented in Figure 13. This spreadsheet includes columns for Activity, Responsible, and Output, capturing the activity performed, the individual responsible, and the resulting output. Therefore, the following mappings can be established between the spreadsheet concepts (**in bold**) and Core-O concepts (*in italic*): **Activity** corresponds to *Human Task*, **Responsible** corresponds to *Person*, and **Output** corresponds to *Task Output*.

<b>Activity</b>	<b>Responsible</b>	<b>Initial Date</b>	<b>Conclusion Date</b>	<b>State</b>	<b>Output</b>
Develop Project C Schedule	Emily	15/01/2024	10/02/2024	Completed	<a href="#">Link to Output</a>
Develop Project A Costs Spreadsheet	Emily	11/02/2024	05/03/2024	Completed	<a href="#">Link to Output</a>
Develop Project B Costs Spreadsheet	Alex	06/03/2024	01/04/2024	Completed	<a href="#">Link to Output</a>

Figure 13 – *Org* spreadsheet

#### (vi) Define Information Model

Using the Core-O extract as a reference model, in this step the conceptual model to be used to structure the system data is developed. The model is based on Core-O extract and it is necessary to make adjustments in the ontological model to turn it into an information model, which is more suitable for implementation. An information model concerns what kind of information may be stored and exchanged considering demands of

specific agents (the “recorded world”), while an ontology model concerns metaphysical aspects of a domain (i.e., it concerns what is considered to exist in the “real world”). Thus, by turning the ontological model into an information model, the resulting model preserves the conceptualization in a structure more suitable for computing demands (CARRARETTO; ALMEIDA, 2012).

In this step, Core-O concepts are turned into classes or attributes. Moreover, the class attributes are identified and new attributes or classes necessary to address the system requirements are defined. Existing approaches in literature can support this step more comprehensively, such as (CARRARETTO; ALMEIDA, 2012). Carraretto e Almeida (2012) provide the initial steps for a principled two-level approach in which a domain ontology addressing ontological concerns is used as a starting point for the definition of an information model, according to a certain information demand.

Example: Suppose that the Core-O extract conceptual model is suitable for the system to be developed to *Org*, but, as informed in *step (iii)* it is necessary to record information about non-technical factors. To do that, new attributes can be added to the Person class to record availability, language, seniority, and years of service to each person. Figure 14 illustrates an information model (hypothetical) to the *Org* case - the added attributes are highlighted with a red rectangle.

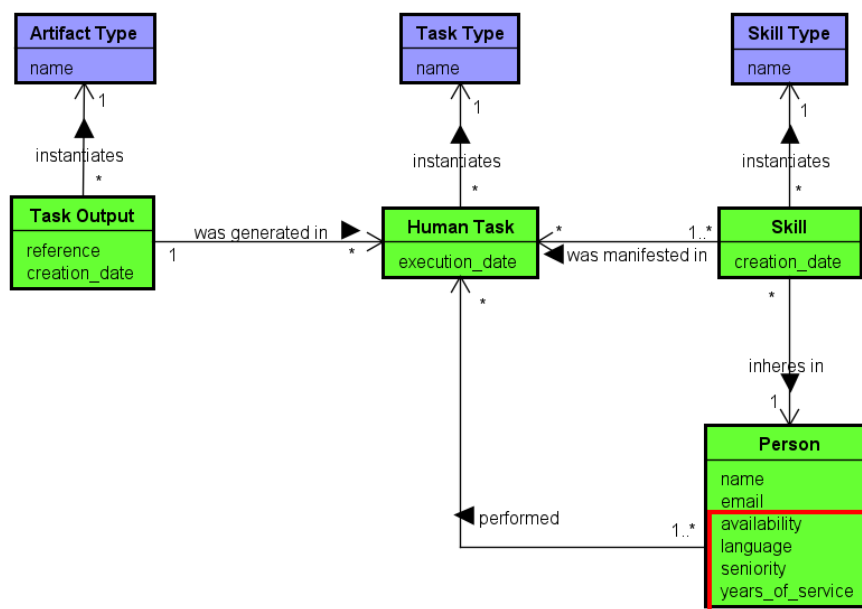


Figure 14 – *Org* hypothetical information model

(vii) Define Architecture

In this step, the system architecture is defined. Bass, Clements e Kazman (2012) cite three reasons for software architecture’s importance: communication among stakeholders;

early design decisions, and transferable abstraction of a system. Software architecture represents a common abstraction of a system that can be used by most stakeholders as a basis for mutual understanding and communication. Furthermore, software architecture manifests early design decisions that have implications in the entire software life-cycle, and this is the first opportunity to analyze how the system will behave and meet its requirements.

A solution for expert identification can be divided into two large contexts: one general that focus on providing a way to access the extract data, and other that is dependent of the identified tools and defined concepts mappings. Therefore, in Figure 15, we propose an architecture that can be used as basis for developing a system for identifying who knows what and contains three modules: Core Competence Module; Data Loading Module; Data Extraction and Transformation Module.

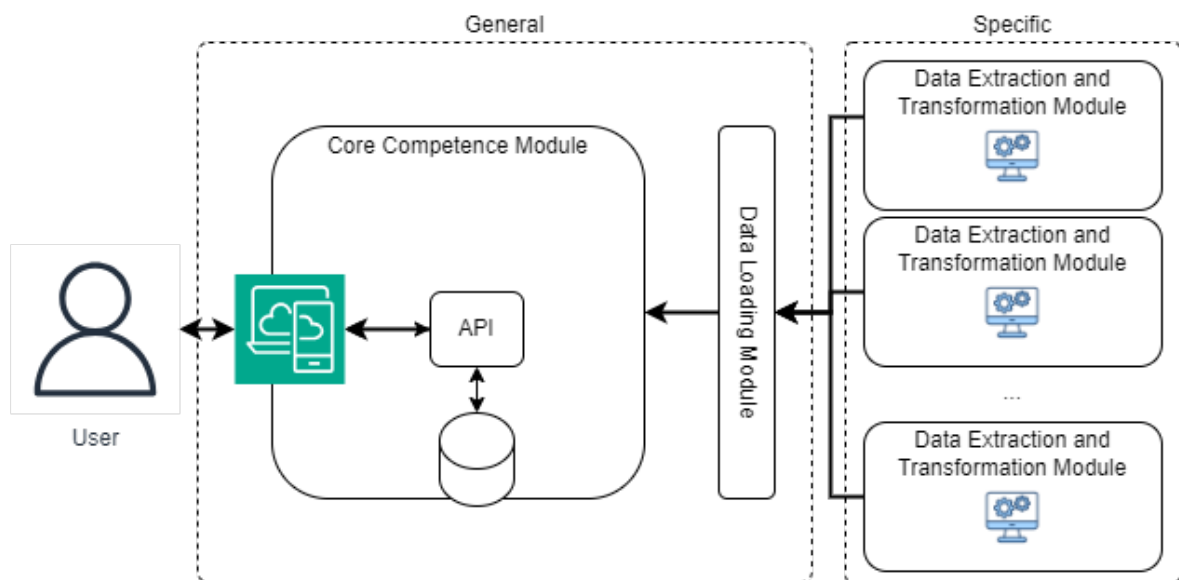


Figure 15 – An architecture for developing applications to support the identification of who knows what

The *Core Competence Module* is responsible for storing information about who knows what and allowing users to access these data through web applications, mobile apps, or other types of applications. The *Data Ingestion Module* provides an application programming interface (API) that can be used for third parties to ingest data from external sources into the platform. These modules have a general purpose - i.e., they handle concepts such as skills, tasks, and so on, and do not refer to any specific domain such as Software Project Management or Software Requirements.

The *Data Extraction Module* is responsible for extracting, transforming, and loading data into the *Core Competence Module* through the *Data Ingestion Module*. This module

must consider the information model from the source and map its concepts to the core concepts. Therefore, this module must address context particularities such as domain concepts, tools used, data available, and so on.

#### (viii) Develop System

According to Sommerville (SOMMERVILLE, 2016) the software development process involves four basic activities: specification, development (design and implementation), validation (deployment), and evolution. The identified skill types of interest and relevant non-technical factors can be seen as special types of requirements. The previous step supports architectural design which is part of the development activity. In this sense, the previous steps support partially the specification and development activities. In this step, design, implementation, and deployment activities are performed. Moreover, new information may be considered to refine the system specification.

*iKnow* is a development process-agnostic approach, meaning the approach does not prescribe the steps related to the system development and is flexible across various software development processes. Therefore, this step can be executed within the preferred software development process.

### 4.3 Related Work

Many studies in the literature have addressed the expert finding systems topic (e.g., (LUCAS et al., 2020; SCHETTINO et al., 2019; TEUSNER; MATTHIES; GIESE, 2017; MORALES-RAMIREZ et al., 2015; HUGHES; CROWDER, 2003)). When analyzing these proposals, we noticed they were not developed to cover all expert-finding systems aspects. Most of them focus on specific sources of expertise evidence and often focus on specific techniques for extracting or consolidating existing data.

Lucas et al. (2020) presented four knowledge-oriented models to represent the developer's knowledge about the following elements of a software project: artifacts, tasks, similar tasks, and the whole software project. These models combine information from interactions between developers and between developers and artifacts. Schettino et al. (2019) proposed a technique that leverages data from existing pull requests in Github to identify collaboration experts. The approach utilizes a network structure to map the influence of an individual over others, with weighted edges representing how influential is a developer compared to others. In Teusner, Matthies e Giese (2017), is proposed a framework to analyze developers' expertise on parts of the system (components) based on code complexity measures. By aggregating these measures, the framework identifies component experts. In Morales-Ramirez et al. (2015), is proposed a process for exploiting online feedback and discussions to compute expertise indicators. This process uses natural language processing

(NLP) and graph-based analysis and mentions the future use of a communication ontology. Hughes e Crowder (2003) proposed an architecture containing modules that, given a search query, produce a ranked list of documents or individuals. Existing results are rescored, and a final score is computed for each document or person. This final ranking is then presented via an interface. This work uses an ontology describing the domain of application. However, the ontology is not grounded on a foundational ontology, it is about the application domain instead of addressing concepts about knowledge and skill.

We defined some aspects to compare the existing approaches with *iKnow*: (i) sources of knowledge evidence; (ii) explicitly considers organizational needs for developing a system; (iii) it is based on a well-founded conceptualization regarding knowledge and/or skills. In Table 12, we present the comparison of the cited works based on these aspects. These studies are presented in the table according to the id defined in Section 3.3 as follows: #3 (LUCAS et al., 2020); #4 (SCHETTINO et al., 2019); #7 (TEUSNER; MATTHIES; GIESE, 2017); #8 (MORALES-RAMIREZ et al., 2015); #14 (HUGHES; CROWDER, 2003).

Aspect	#3	#4	#7	#8	#14	iKnow
<b>(i) Sources of knowledge evidence.</b>	Code repository and ticket system.	Code repository	Code repository.	Code repository.	Electronic documents.	Supports multiple sources.
<b>Explicitly considers organizational needs for developing a system.</b>	No.	No.	No.	No.	No.	Yes.
<b>Based on a well-founded conceptualization regarding knowledge and/or skills.</b>	No.	No.	No.	No.	No.	Yes.

Table 12 – Aspects addressed by expert-finding system approaches

Therefore, none of the analyzed works provide a holistic view of the process of developing expert-finding systems. They typically handle technical concerns and approaches for extracting data from specific sources. In this sense, we suggest that some of the identified approaches can be combined with *iKnow* to support the development of effective systems. This is possible because *iKnow* defines a high-level guide for identifying sources of knowledge evidence and developing expert-finding systems. However, it does not address low-level concerns, such as optimal methods for calculating expertise scores.

## 4.4 Concluding Remarks

This chapter presented *iKnow*, an ontology-based approach to support the development of systems to identify who knows what in software organizations. *iKnow* provides sequential steps anchored in Core-O concepts, considering organizational requirements and leveraging existing data to support the development of expert-finding systems.

To demonstrate that using the proposed approach is feasible and it is useful, *iKnow* was used to develop an expert-finding system for a Brazilian company. The system and its evaluation are presented in the next chapter.

# 5 ExpertFY: A System to Identify Who Knows What in a Software Organization

*This chapter presents ExpertFY, a system developed using iKnow that supports identifying who knows what concerning some aspects of software development (e.g., codification, testing, among others). Section 5.1 introduces the chapter. Section 5.2 presents ExpertFY. Section 5.3 regards ExpertFY evaluation. Section 5.4 discusses limitations and threats to validity. Finally, Section 5.5 presents the chapter's concluding remarks. In the DSR context, this chapter is related to the Design cycle (Figure 1) as it presents the systems developed using the main produced artifact and its evaluation.*

## 5.1 Introduction

To demonstrate the feasibility of using *iKnow* and evaluate its usefulness, it was used to develop a tool called *ExpertFY*, which was applied in a large Brazilian mobility company. In the context of this work, for anonymity reasons, this company will be referred to as BMC.

## 5.2 ExpertFY: *An Expert for You*

### 5.2.1 Study Design

BMC's IT department has approximately 1500 professionals who adopt agile practices and are organized into 24 tribes (or areas) and more than 200 squads. Roles include software engineer, product owner, solution architect, UX designer, data scientist, and data engineer, among others. Given the diversity of skill types and the dynamic nature of these teams, there is a need for collaboration and knowledge sharing across different projects and about several technologies. With many individuals working in different contexts, it can often be challenging for members to identify the right individual to seek help, which can lead to inefficiencies. Therefore, a system that supports the identification of experts based on existing organizational data and non-technical factors could facilitate this process. These aspects motivated the development of ExpertFY for BMC. The organization was selected by convenience (the author of this work had access to people in the company).

The study **goal** was to evaluate if using *iKnow* to develop a system to help identify who knows what in an organization is feasible and if the produced system is useful, which would indicate the usefulness of *iKnow*.

The study involved two phases. In the first one, the author of this work executed the seven first steps of *iKnow* with the help of BMC employees. Then, considering the results produced in those steps, an undergraduate student executed the eighth step of *iKnow* and developed the *ExpertFY* tool, a system to help identify who knows what in BMC. In the second phase, the *ExpertFY* tool was made available for some of the BMC employees, and a survey was performed to collect feedback about their perceptions of the tool.

## 5.2.2 Study Execution and Results

In this section, we present the results produced for each activity of *iKnow*.

### (i) Identify Domain and Skill Types of Interest

At BMC, the focus was to identify who knows what in the *Software Development* domain. The skill types of interest were gathered through unstructured interviews with four BMC Tech Managers. Six skill types were identified: *Optimizing SQL query*, *Monitoring application*, *Programming unit test*, *Using .NET library*, *Designing Rest API*, and *Documenting software*.

To represent these skill types in the context of the conceptualization adopted in *iKnow*, in Figure 16 we present the identified skill types as specializations of Skill Type.

### (ii) Identify Task Types and Artifact Types

By analyzing the organization (mainly its adopted practices and tools), we identified some task types that could represent manifestations of the previously identified skill types. We represented the relation Skill Types, Task Types and Artifact Types as a set of assumptions, as shown in Table 13.

Given that the Task Types indicated as manifestations of Skill Types (and, thus, are indications of knowledge) were identified by this work author, BMC members needed to evaluate them before we proceed to the next step. For that, we performed a survey with 11 BMC software developers. The participants' profile was identified through questions regarding their education level, amount of time working in software development, role, and amount of time working at BMC. The participants were asked about their level of agreement with each assumption. The assumptions were simplified in the survey by omitting the Artifact Types to facilitate participants understanding.

Based on the answers provided by the participants, we calculated the overall score that represents the average level of agreement with each assumption. The resulting score may range from -2 to 2, with negative values indicating that the participants disagree and

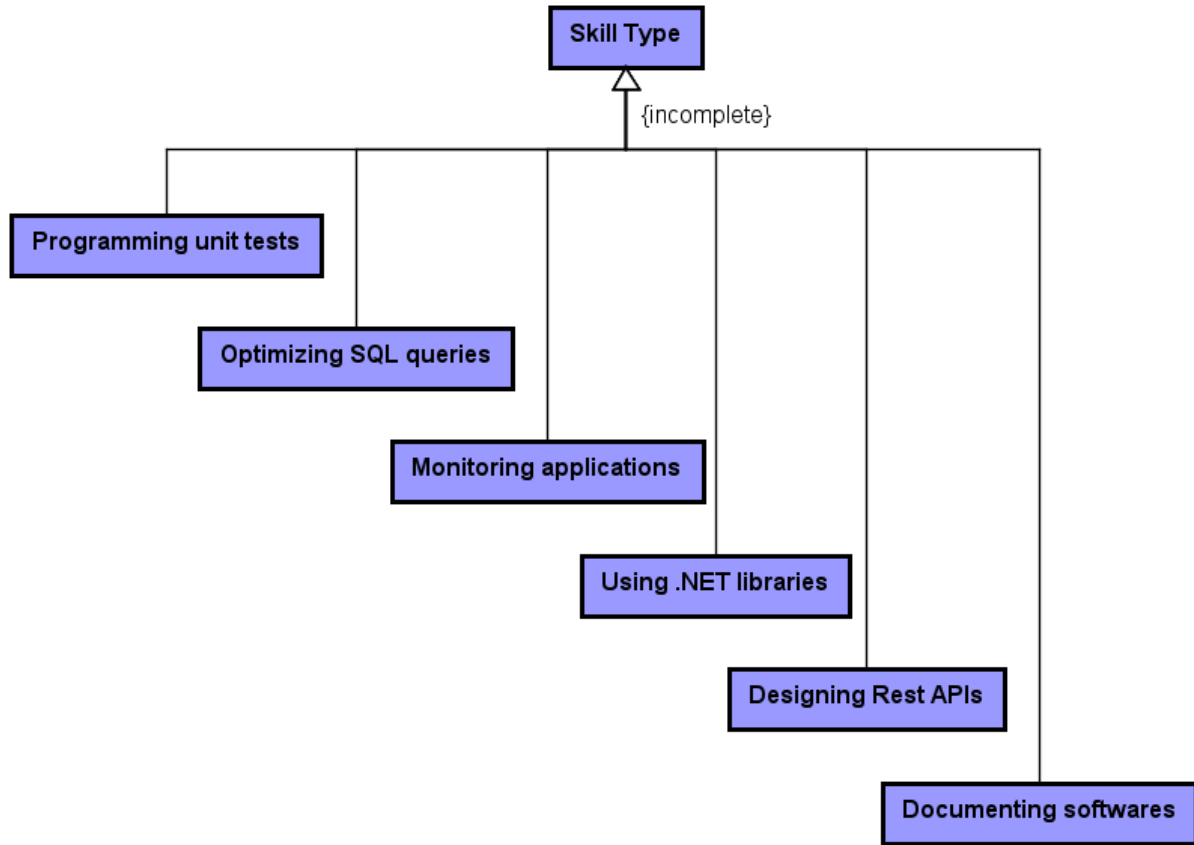


Figure 16 – Skill types identified in BMC

positive values indicating that they agree. Assumption A1 had an overall score of 1.36, A2 had a score of 0.81, A3 had a score of 0, A4 had a score of 0.2, A5 a score of 0.63, A6 0, and A7 a score of 0.18. These results are presented in Table 23 in descending order by score. Additional information about the survey is presented in Appendix A.

It is possible to notice a level of agreement with most assumptions with participants being neutral about the assumptions A3 and A6. Considering that A4 and A5 are assumptions related to the same skill type, we opted for keeping only A5, since it had better favorability.

Considering time constraints, it would not be possible to address all the assumptions in this system to be developed. Thus, we prioritized them and selected the ones to be addressed. We discarded assumptions A3 and A6 because they received neutral ratings from participants, and A7 because presented low favorability. We also discarded A4 because it is associated with the same Skill Type as A5, and participants rated A4 less favorably.

To represent these task types and their relations with skill types in the context of the conceptualization adopted in *iKnow*, in Figure 17 we present an extended fragment of Core-O.

ID	Assumptions
A1	The skill type <u>Optimizing SQL query</u> is manifested by the task type <u>Optimize SQL queries</u> which produces the artifact type <u>Optimized SQL query</u> .
A2	The skill type <u>Monitoring application</u> is manifested by the task type <u>Create or modify alerts</u> which produces the artifact type <u>Application alerts</u> .
A3	The skill type <u>Programming unit test</u> is manifested by the task type <u>Create or modify unit tests</u> which produces the artifact type <u>Unit test cases</u> .
A4	The skill type <u>Using .NET library</u> is manifested by the task type <u>Add the library reference in a code repository</u> which produces the artifact type <u>Library reference in a class</u> .
A5	The skill type <u>Using .NET library</u> is manifested by the task type <u>Create or modify a file referencing the library</u> which produces the artifact type <u>Library referencing application</u> .
A6	The skill type <u>Designing Rest API</u> is manifested by the task type <u>Create or modify a controller class</u> which produces the artifact type <u>Controller class</u> .
A7	The skill type <u>Documenting software</u> is manifested by the task type <u>Create or modify wiki pages</u> which produces the artifact type <u>Software documentation</u> .

Table 13 – Skill type manifestation assumptions

Assumption	Score
A1	1.36
A2	0.82
A5	0.64
A4	0.27
A7	0.18
A3	0.00
A6	0.00

Table 14 – Score level of agreement for assumption

### (iii) Identify Relevant Non-Technical Factors

The survey performed to evaluate the assumptions defined in the previous step also included questions regarding non-technical factors considered relevant to the participants when searching for an expert to share knowledge. The three more relevant aspects according to the 11 participants were: availability; years of experience; and seniority.

### (iv) Identify Available Data Repositories

To provide data related to these task types and artifact types (i.e., data representing performed tasks of those types and produced task outputs of those types), among the tools used by BMC, we identified the data repositories of Azure Boards<sup>1</sup> and Azure Repos<sup>2</sup>

<sup>1</sup> <https://azure.microsoft.com/en-us/products/devops/boards/>

<sup>2</sup> <https://azure.microsoft.com/en-us/products/devops/repos/>

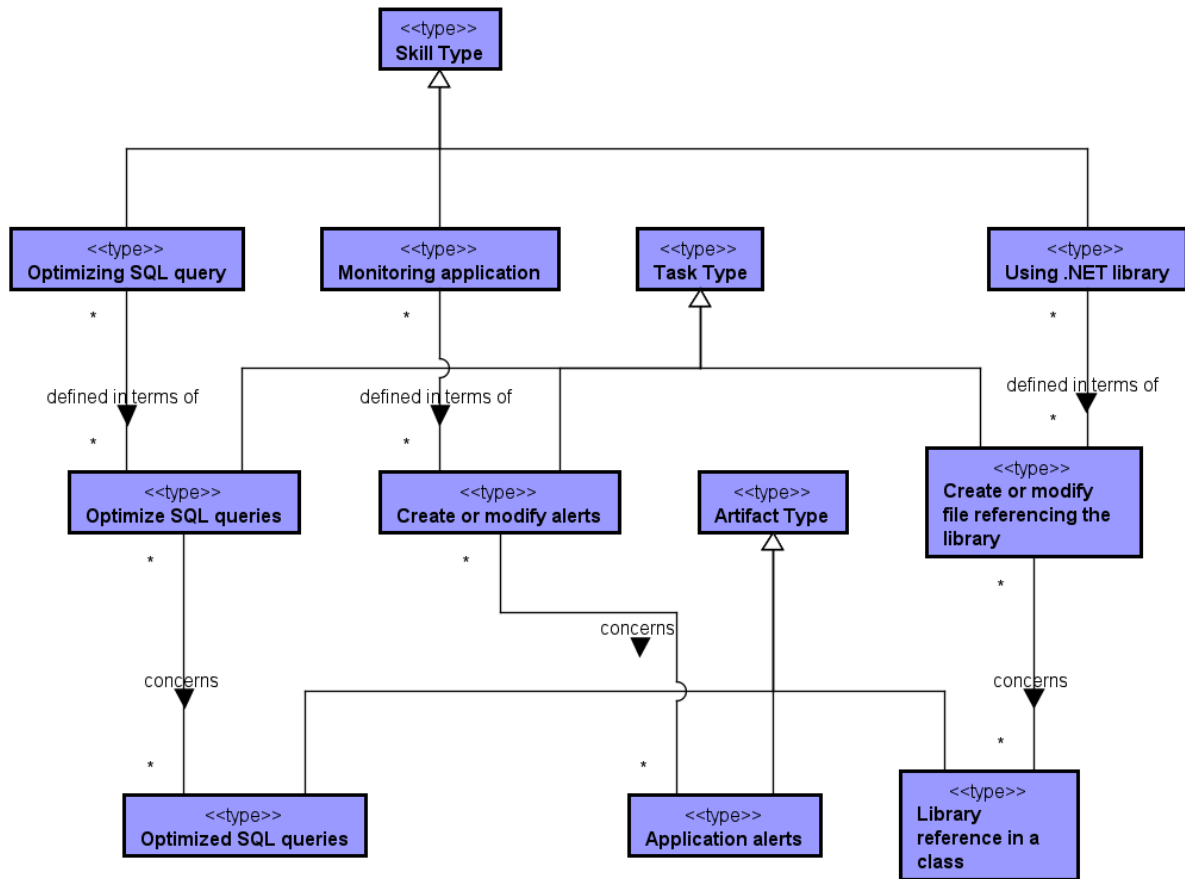


Figure 17 – An extended fragment of Core-O

(modules of Azure DevOps<sup>3</sup>), and Datadog<sup>4</sup>. Next, we provide information about them.

Azure DevOps is a complete set of development tools that allow teams to plan, develop, test, and deliver software efficiently. It is composed of modules, such as Azure Boards, and Azure Repos, among others. Azure Boards is a web-based service that enables teams to plan, track, and discuss work across the entire development process, while it supports agile methodologies. Also, it provides a customizable platform for managing work items, allowing teams to collaborate effectively and streamline their workflow. In that regard, most of the *tasks* executed by software developers are registered and available in Azure Boards including information like who executed the task, the description of the task, related tasks, and others.

Azure Repos is a tool designed to help development teams manage and track changes to their codebase. One of its main features is a version control system (VCS) that allows us to identify who changed the codebase, the changes made, and when they were made. The VCS used in BMC is Git.

Datadog is a cloud-based monitoring and analytics platform designed for IT

<sup>3</sup> <https://azure.microsoft.com/en-us/products/devops>

<sup>4</sup> <https://www.datadoghq.com/>

and DevOps teams. It provides comprehensive monitoring, security, and performance management across various infrastructures, applications, and services, both in the cloud and on-premises.

#### (v) Define Concepts Mapping

In this step, we captured the conceptual models of the tools selected in the previous step (we considered only the portion of interest). In Azure Boards, a WorkItem is a unit of work that tracks a specific task, requirement, bug, or other deliverable within a project. This tool allows assigning a work item to a User, meaning this user is responsible for completing the unit of work. Therefore, we elaborated the conceptual model presented in Figure 18.

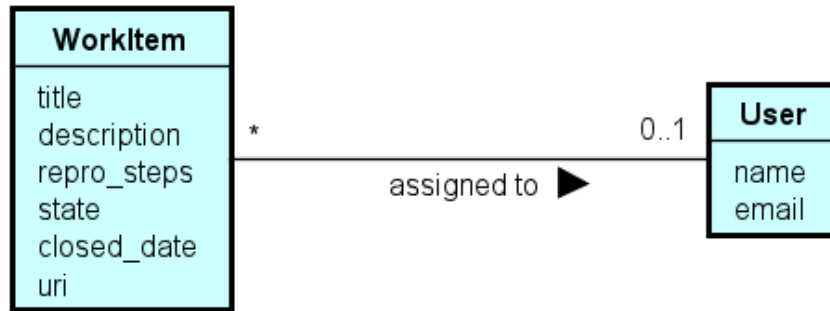


Figure 18 – Azure Boards conceptual model

Then, we identified the semantic mappings between the Azure Boards conceptual model and COre-O extract. The mappings are presented in Table 15. It refers to the identification of Tasks related to the Skill Type *Optimizing SQL Query*. Work Item was mapped to Task when it satisfies the following constraints: (i) At least one of the attributes `title`, `description`, and `repro_steps` containing SQL query optimization textual terms; (ii) The value of attribute `state` equals to `Closed`. Person who performed Task was mapped to User that was assigned to Work Item. Considering that the **optimized SQL query** is not always linked to Work Item, we opted for mapping Work Item itself to Task Output.

Datadog Monitor is a feature within the Datadog platform that allows users to set up monitoring rules (*alerts*) to track the performance and health of their systems, services, and applications. The tool stores the user who created a monitor. In this sense, we elaborated the conceptual model presented in Figure 19.

The semantic mappings between Datadog conceptual model and Core-O are shown in Table 16. It refers to the identification of Tasks related to the Skill Type *Monitoring application*. The Task was implicitly defined by the relation `created by`, and Monitor was mapped to Task Output. Finally, User was mapped to Person.

Core-O	Azure Boards Concept	Description
Task	Work Item	Constraints: (i) At least one of the attributes <code>title</code> , <code>description</code> , and <code>repro_steps</code> containing SQL query optimization textual terms in. (ii) The value of attribute <code>state</code> equals to <code>Closed</code> .
Task Output	Work Item	Constraints: (i) At least one of the attributes <code>title</code> , <code>description</code> , and <code>repro_steps</code> containing SQL query optimization textual terms in. (ii) The value of attribute <code>state</code> equals to <code>Closed</code> .
Person	User	-
Skill Type	"Optimizing SQL query"	Static value.

Table 15 – Azure Boards concepts mapping

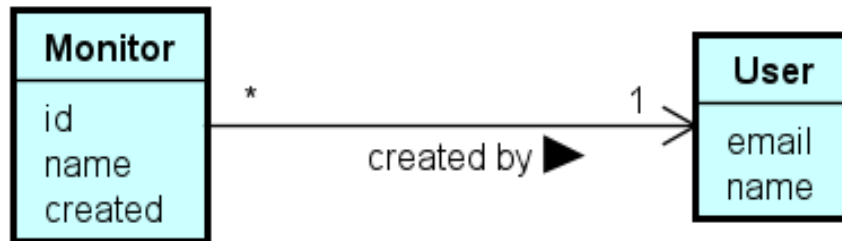


Figure 19 – Datadog conceptual model

Core-O	Datadog Concept	Description
Task	<i>created by</i>	Implicitly defined by the relation <i>created by</i> .
Task Output	Monitor	-
Person	User	-
Skill Type	"Monitoring application"	Static value.

Table 16 – Datadog concepts mapping

In the context of Git (the VCS used in Azure Repos), there are three core concepts: Commit, File, and User. Commit represents a snapshot of changes made to one or more files. File is any piece of data (e.g., code, configuration, or documentation) tracked in the repository that can be modified, added, or deleted. Finally, User is the individual responsible for making changes to files and creating commits - the author of a commit. As a result, we elaborated the conceptual model presented in Figure 20.

In this case, File was mapped to Task Output. However, File must satisfy the following constraint: (i) the instance of File represents a C# class referencing the library MassTransit. The User was mapped to Person, and the Task is implicitly defined by the relation `changed` which captures the task of changing a file. These mappings are presented in Table 17.

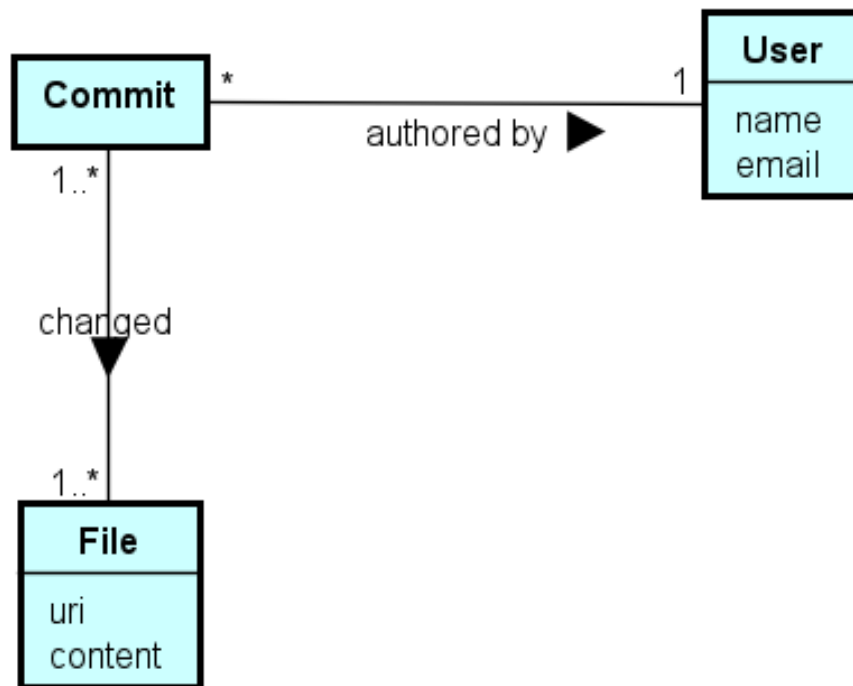


Figure 20 – Git conceptual model

Core-O	Git Concept	Description
Task	<i>changed</i>	Implicitly defined by the relation <i>changed</i> .
Task Output	File	Constraints: (i) File that represents a C# class referencing the library MassTransit.
Person	User	-
Skill Type	"Using MassTransit library"	Static value.

Table 17 – Git concepts mapping

#### (vi) Define Information Model

We made some adjustments to the Core-O conceptual model utilized by *iKnow* to turn it into an information model, which is more suitable for implementation. In summary: (i) Artifact Type concept was represented as an attribute of Task Output; (ii) Task Type concept was represented as an attribute of Human Task; (iii) we changed the multiplicity of relation *was generated in* because we are interested only in Task Outputs related to a Human Task; (iv) we changed the multiplicity of relation *was manifested in* because we consider Human Task as the only mean to manifest a Skill; (v) we changed the multiplicity of relation *performed* for simplicity reasons; (vi) the concept Evidence was replaced by Recommendation - the only mapped subclass of Evidence for ExpertFY; and (vii) we added the necessary attributes to the classes (e.g., to characterize a Person). Figure 21 presents the resulting conceptual model.

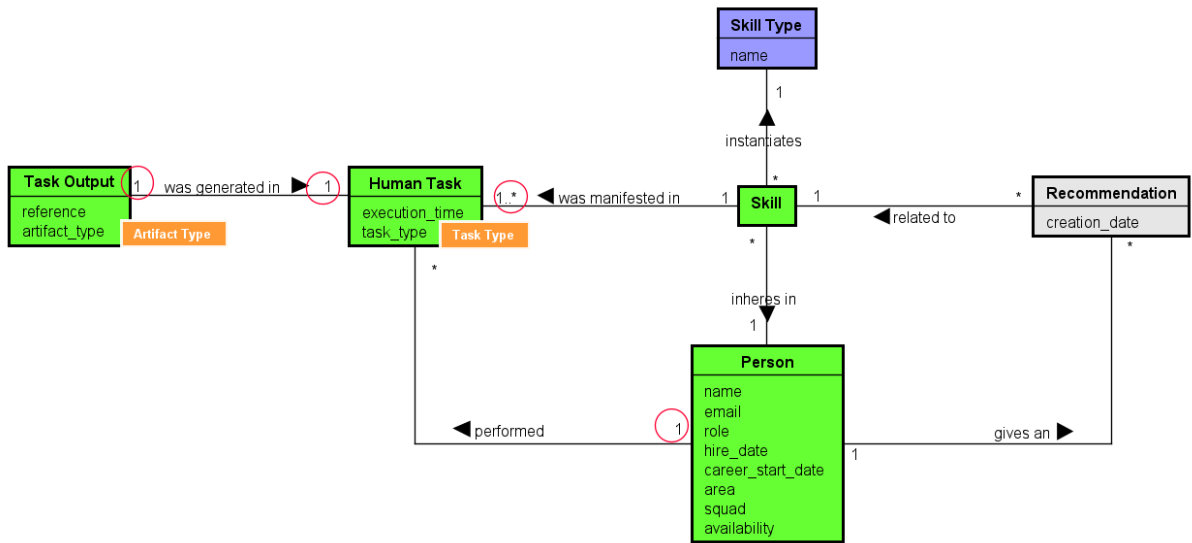


Figure 21 – ExpertFY conceptual model

(vii) Define Architecture

Considering time constraints, we simplified the reference architecture presented in Figure 15. The *Core Competence Module* was the only one fully developed. To insert the necessary data into ExpertFY we implemented various scripts that extracted the data from the available repositories within BMC, transformed the data according to the defined concepts mapping, and loaded the transformed data directly into ExpertFY database. The defined architecture for ExpertFY is presented in Figure 22.

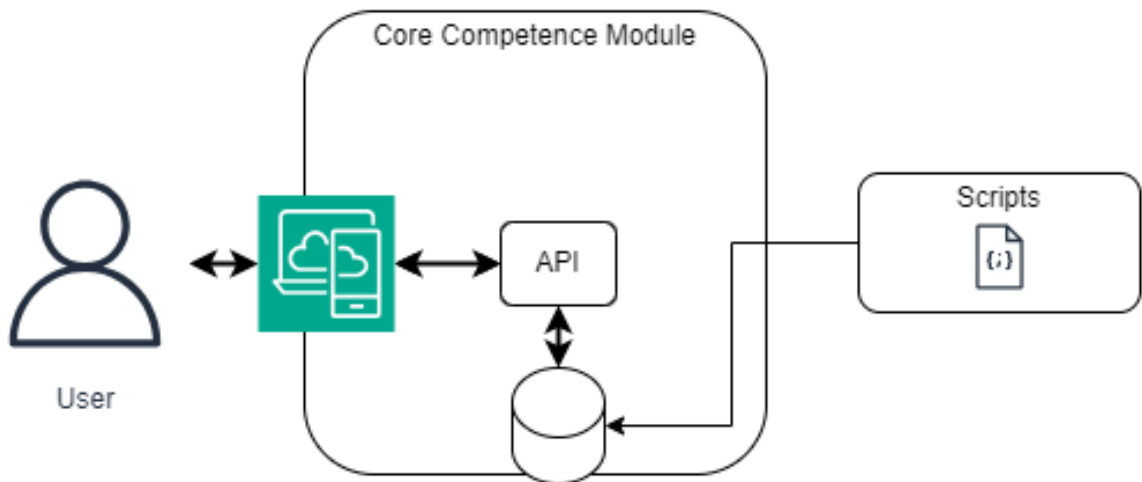


Figure 22 – ExpertFY architecture

(viii) Develop System

In this step, the ExpertFY system was developed by a Computer Science undergraduate student supervised by the author of this work (SANTOS, 2024). We used agile

method with sprints of one week. The main features (use cases) of ExpertFY are presented below. Concepts existing in the extract of Core-O utilized are written in *italic*.

**Search Who Possess a Skill:** this use case aims to allow users searching for a *Person* who possesses *Skill* of a given *Skill Type*. This use case also offers the possibility of using an advanced filter allowing the user to filter the results for area, availability, and minimum years of experience. Figure 23 presents a screen showing the fields that can be used in the search and Figure 24 presents a screen in which the tool presents the results.

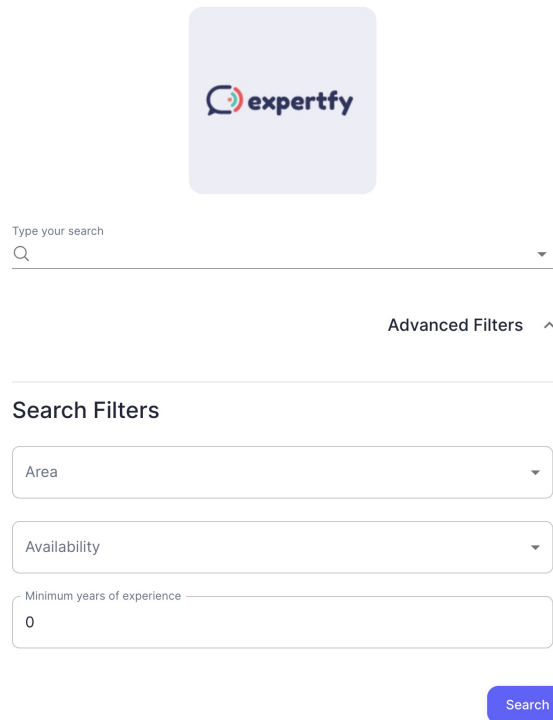


Figure 23 – Search panel with filters in ExpertFY

Name	Email	Role	Years in the Organization	Area	Squad	Years of Experience	Availability	Score	Recommendations
Person 355	person.355@company.com	Software Developer	6 years	Area 23	Squad 901	6 years	Available	10	0
Person 1236	person.1236@company.com	Software Developer	2 years	Area 44	Squad 499	2 years	Available	9	0
Person 374	person.374@company.com	Software Developer	2 years	Area 44	Squad 623	2 years	Available	7	0
Person 1300	person.1300@company.com	Software Developer Specialist	6 years	Area 44	Squad 610	6 years	Available	7	0
Person 53	person.53@company.com	Terceirizado	2 years	Area 44	Squad 145	2 years	Available	6	1

Rows per page: 5 1-5 of 98 < >

Figure 24 – Search results in ExpertFY

**Recommend a Person's Skill:** this use case aims to record recommendations of a *Person's Skill*. The user can recommend only *Skills* previously manifested through *Tasks*. Figure 26 presents a screen showing how a user can recommend a *Person's Skill* - the thumbs up icon in the right side of each *Skill*.

**View Profile:** this use case aims to allow the user to view a *Person's* profile, including their manifested *Skills*, received recommendations, performed *Tasks*, and generated *Task Outputs*. The three sections of the *Person's* profile screen are presented in Figure 25, Figure 26, and Figure 27.

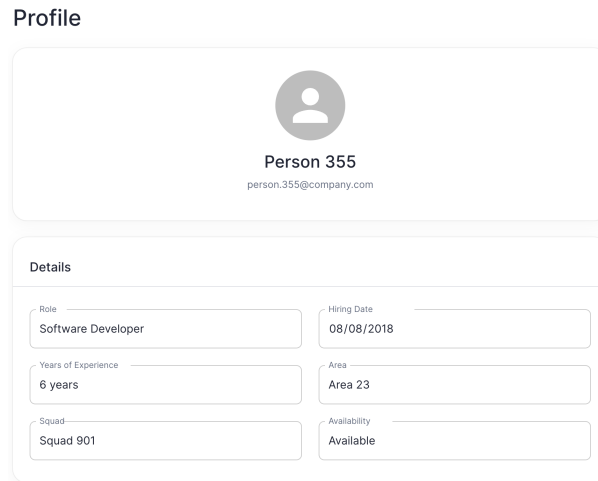


Figure 25 – Profile detail section in ExpertFY

## Skills

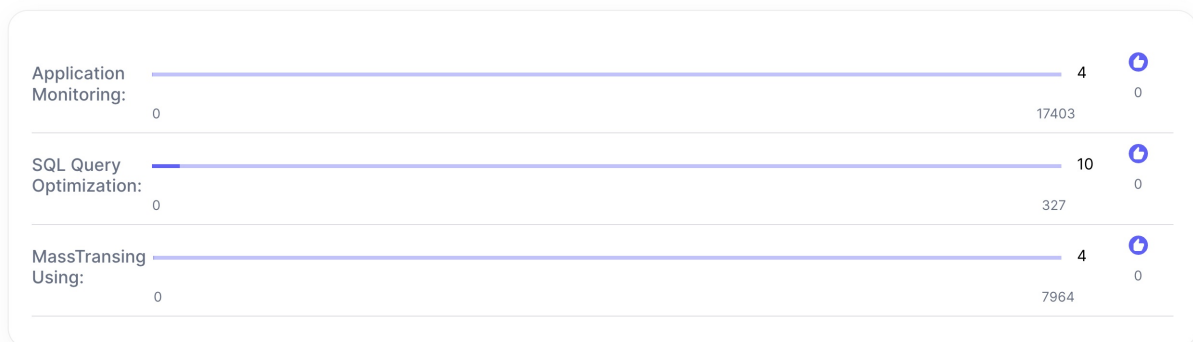


Figure 26 – Profile skills and recommendations section in ExpertFY

After developing ExpertFY, it was made available to five software developers of BMC. Due to time constraints, in the context of this work, it was not possible to release the tool to more people.

## 5.3 A Survey with ExpertFY Users

Performing the steps and producing the results described in the previous section served as a proof of concept to demonstrate that using *iKnow* is feasible. In this section, we present a survey performed with ExpertFY users to collect their perceptions of the tool.

## Tasks

Task execution time	Task Type	Skill Type	Artifact
22/05/2024	Creation of Datadog monitor	Application Monitoring	<a href="#">↔</a>
21/05/2024	Creation of Datadog monitor	Application Monitoring	<a href="#">↔</a>
16/04/2024	Creation or Modification of file referencing MassTransit	MassTransiting Using	<a href="#">↔</a>
29/01/2024	Creation of Datadog monitor	Application Monitoring	<a href="#">↔</a>
21/01/2024	Creation or Modification of file referencing MassTransit	MassTransiting Using	<a href="#">↔</a>

Rows per page: 5 ▾ 1-5 of 18 < >

Figure 27 – Profile tasks in ExpertFY

### 5.3.1 Survey Planning

The study **goal** was to evaluate if ExpertFY is useful to support the identification of who knows what in software organizations and if its use is feasible. By evaluating ExpertFY usefulness, we can indirectly evaluate *iKnow* usefulness (if ExpertFY is useful, this can be understood as a preliminary evidence that *iKnow* is useful, since it was used to develop ExpertFY).

Following the GQM approach (BASILI; CALDIERA; ROMBACH, 1994), this goal is formalized as follows:

*Analyze* ExpertFY

*For the purpose of* evaluating its use to aid in who knows what identification

*With respect to* usefulness and feasibility of use

*From the viewpoint of* software developers

*In the context of* software development.

To analyze the results, the following indicators were considered: usefulness and feasibility. The former was evaluated considering the participants' perceptions about the adequacy of the tool (taking its purpose into account) and how much the tool helped them in who knows what identification. The latter considered the participants' perceptions about ease of use and how much feasible they considered using the tool to aid in who knows what identification. Benefits and drawbacks pointed by the participants were also considered to indicate if the tool is useful and feasible.

The **instrument** used in the study consisted of four artifacts: (i) a document presenting the context of this work, the main functionalities of ExpertFY and the instructions to be considered by the participants when using the tool; (ii) a consent form to participate in the study, which aims to safeguard the participants' rights regarding the study and

its results; (iii) a form to characterize the participants' profile, which aims to obtain information about the participants' knowledge and experience in software development; and (iv) a questionnaire that allows participants to record their perception after using the tool. The forms were prepared with the help of Google Forms and are presented in Appendix B of this dissertation together with the document cited in (i).

The **procedure** adopted in the study consisted of inviting the participants and sending them the document that presents ExpertFY and the study instructions. The author of this dissertation also made a brief presentation of the document and made himself available to assist and answer questions during the study. The participants were chosen considering convenience criteria (i.e., they were invited based on the researchers' relationship network and should be available to participate within the time necessary to conclude this dissertation). Five people were invited and participated in the study. After the ExpertFY presentation made by the author of this work, the participants should use ExpertFY to support them in performing the activity described in the document and then answer a questionnaire in which they should record their perception about the use of ExpertFY in supporting who knows what identification.

The questionnaire included questions aimed to extract the participants perception about the adequacy of ExpertFY in supporting expert identification for knowledge sharing purposes, the utility of the tool in software development context, and the feasibility of using it in practice. The questionnaire consisted of objective and subjective questions. For the objective ones, the participants were asked to justify their answers. There was also a subjective question in which the participants could provide general improvement suggestions to ExpertFY. A fragment of the questionnaire is presented in Figure 28. The complete questionnaire is available in Appendix B.

### 5.3.2 Survey Execution

The participants were five software developers who works at BMC. All of them informed that they consider very important knowledge sharing to support task execution in software development. Two of them informed that they seek help from others to perform their tasks almost daily, two of them seek help from others once or twice a week and one of them seeks help once or twice a month. The participants' characterization is presented in Table 18.

Following the planned procedure, after a presentation made by the researcher, the participants read the document with the instructions and the description of three scenarios (related to the identified skill types) on which it was necessary to identify up to five individuals who could provide knowledge to help on performing a task. In the first stage of the study, the participants should identify the experts without the use of the tool. After that, in the second stage, they used the tool to find the experts for the same

**Feedback**

What is your perception of the usefulness of the tool? \*

Very Useful  
 Useful  
 Somewhat Useful  
 Not Useful

---

**Justification: \***

Your answer \_\_\_\_\_

---

Do you consider that the use of the tool for identifying "who knows what" contributed to any of the following results? \*

	Yes	No
LESS EFFORT employed to identify "who knows what"	<input type="radio"/>	<input type="radio"/>
LESS TIME spent to identify "who knows what"	<input type="radio"/>	<input type="radio"/>
CREATER ACCURACY in "who knows what" identification	<input type="radio"/>	<input type="radio"/>
GREATER COVERAGE of "who knows what"	<input type="radio"/>	<input type="radio"/>

---

**Justification: \***

Your answer \_\_\_\_\_

Figure 28 – Fragment of the feedback questionnaire about the use of ExpertFY

scenarios. Then, they compared the results and answered the questionnaire. The scenarios description and the instructions are presented in Appendix B.

The participants selected 60 individuals in the first stage (ST1) and 58 in the second stage (ST2), with 16 individuals selected in both stages ( $ST1 \cap ST2$ ). It is important to note that some individuals may have been selected in the context of different scenarios by different participants, resulting in their being counted multiple times. A total of 42

Participants / Aspects	P1	P2	P3	P4	P5
<b>Academic Degree</b>	Bachelor's Degree	Specialization	Bachelor's Degree	Bachelor's Degree	Bachelor's Degree
<b>Years of Experience</b>	Between 5 and 10 years	More than 10 years	More than 10 years	Between 3 and 5 years	Between 1 and 3 years
<b>Role</b>	Fullstack Developer	Backend Developer	Backend Developer	Salesforce Developer	Fullstack Developer
<b>Time at Organization</b>	Between 1 and 3 years	Between 5 and 10 years	Between 1 and 3 years	Between 3 and 5 years	Less than 1 year
<b>Importance of Knowledge Sharing</b>	Very important	Very important	Very important	Very important	Very important
<b>Frequency of Knowledge Seeking</b>	Sometimes (1 to 2 times per month)	Always (almost daily)	Always (almost daily)	Frequently (1 to 2 times per week)	Frequently (1 to 2 times per week)

Table 18 – Participants characteristics

individuals were selected exclusively in the second stage ( $ST2 \setminus ST1$ ), while 44 were selected only in the first stage ( $ST1 \setminus ST2$ ). Table 19 presents the data detailed per participant.

The data show a considerable change in the list of select expert candidates from the first stage to the second stage. This can suggest that the information provided by the system led participants to consider that there were more suitable experts available than those they had initially identified from memory.

Participant	ST1	ST2	$ST1 \cap ST2$	$ST1 \setminus ST2$	$ST2 \setminus ST1$
P1	9	9	1	8	8
P2	14	13	2	12	11
P3	15	15	7	8	8
P4	15	15	5	10	10
P5	7	6	1	6	5

Table 19 – Selection data by participant

### 5.3.3 Survey Results

This section summarizes the answers and the comments provided by the participants to each of the questions of the questionnaire.

- a. **ExpertFY's usefulness:** Four participants (P1, P3, P4 and P5) considered the tool *very useful*, and participant P2 considered the tool *useful*. P4 mentioned that

the tool can be especially valuable for new hires, who are still unfamiliar with whom to contact for help. P1 commented that the tool gives confidence that someone previously manifested the desired skill, and without the tool, this search is often based on the seeker's intuition. P5 pointed out that the tool may improve the efficiency of problem-solving.

- b. **Results obtained from the use of ExpertFY:** All five participants considered that ExpertFY reduced the effort required to locate individuals with the necessary knowledge, saved time spent in this search, enhanced the accuracy of identifying who possesses the desired skill, and, expanded the scope of people who they could potentially contact for help within the organization. P1 pointed out that ExpertFY provided insights beyond their social circle, and P4 commented that even those who have worked within the organization for a long time may discover different people who could help them. P3 considered that the tool can help someone who really possesses the desired skill which is more effective than the current practice of looking for help in general-use forums or contacting only the widely recognized carriers of the desired skill.
- c. **ExpertFY's ease of use:** Four participants (P2, P3, P4 and P5) considered the tool *very easy* to use, and P1 considered *easy* to use. Participants P3, P4, and P5 commented that the tool is "intuitive", participants P2 and P4 stated that the tool is "simple", and participants P2 and P5 mentioned that tool is very "straightforward".
- d. **Feasibility of using ExpertFY:** Four participants (P1, P2, P3, and P4) considered the tool *very feasible* to use, and P5 considered *feasible* to use. Participant P3 commented that the tool is very feasible to use because the data used in the tool already exists in the organization, and the tool uses it smartly. Participant P2 mentioned that the tool is like a "virtual person" to whom you could ask who knows about a certain subject, but, unlike a real person, it is more accurate, and offers a wider range of options.
- e. **Recommendation of the use of ExpertFY to other people:** All five participants reported that they would recommend other people to use ExpertFY. Among the mentioned benefits, participant P3 highlighted that the tool is simple and effective. Participant P2 stated that they would mainly recommend the tool for new hires because they still do not possess a recognized social circle within the organization, but they would also recommend for people who work in the organization for more time because they may need help regarding a new topic and do not previously know who they could contact for help. Participants P1 and P2 considered the results that may be obtained from the use of the tool as crucial for its recommendation.

- f. **Stimulus to contact others for obtaining useful knowledge for completing tasks when using ExpertFY:** All five participants answered that they believe that the tool would stimulate them to contact other people to seek help. Participant P1 commented that ExpertFY is especially stimulant when stuck and lacking the desired knowledge in their close circle. Participant P3 noted that the tool provides a simple way to connect those who can help with those in need. Finally, participant 5 believed that the tool encourages forming new relationships with people who may not have been previously engaged.
- g. **Benefits that could be obtained and difficulties that could be faced when using ExpertFY:** All five participants reported that they could obtain benefits from using ExpertFY. The benefit reported by participant P1 is the deconstruction of knowledge silos especially in big companies enabled by the possibility offered by the tool of finding any people in the organization who possesses the desired skill. Additionally, participant P4 informed that the tool may improve the integration between different squads. Participant P2 mentioned that the tool is necessary for new hires because it makes it easier to find knowledgeable people. An increment of agility and accuracy when finding who can help was reported as a benefit of the use by participants P3 and P5. Regarding the difficulties that could be faced when using ExpertFY, only participant P1 did not report any difficulty. Participants P2 and P4 reported that the ranking of knowledgeable people may be a problem. According to them, the better-ranked people may be overloaded with requests for help which may harm their availability. Participant P5 mentioned that it may be a barrier to the adoption of the tool. Finally, participant P3 reported that there may be badly described tasks in the organization that do not offer enough data for identifying the manifested skill.
- h. **Suggestions for the evolution of ExpertFY:** Suggestion about the tool was provided only by one participant. According to participant P1, in order to improve ExpertFY, the tool could include different levels of proficiency in a skill. Participants P3 and P5 did not give any suggestions. Participant P2 mentioned that frequently contacted people could start creating wikis to make explicit part of the knowledge they possess. Finally, participant P4 mentioned how their usage of the tool evolved during the evaluation process. According to participant P4, it would be more valuable search for the most knowledgeable people, rather than limiting their search to those who they already know.

### 5.3.4 Survey Discussion

In this section, we present a discussion about the results previously presented in terms of the indicators defined in the study planning. Questions *(a)*, *(b)*, *(e)*, and *(f)*

referred to usefulness. Questions (c) and (d) were used to analyze ExpertFY feasibility of use. Finally, question (g) provided insights in terms of both usefulness and feasibility (e.g., if there are several benefits and few difficulties in using ExpertFY, it means that the tool is feasible and useful).

Regarding the usefulness, participants considered the tool very useful or useful. Participant P4 highlighted the tool is especially valuable for new hires. This makes sense since people in an organization tend to learn who knows what considering previous experiences and needs. However, different needs may arise, and even those in the organization for a long time may need to consult the tool which was also mentioned by the same participant. In accordance with this, participant P2, who had been in the organization for the longest time among the participants, was the only one who classified the tool as useful and not very useful.

Additionally, participant P5 reported that the tool may improve the efficiency of problem-solving. We believe the tool allows people to be more protagonists and effective in seeking someone who can help them. As mentioned by two participants, they consider this form of search more effective because they know in advance if someone already manifested a skill. Also, it is more effective than looking for help in general-use forums. Another interesting point regards the possibility of finding experts out of participants' social circle. This can help address the problem of knowledge silos - a situation where important expertise is isolated within specific areas or individuals.

Considering these factors, we believe that the usefulness of the tool is directly proportional to the size and complexity of an organization. In an organization with 10 employees, for example, it is easier to discover who knows what if compared with an organization with 1000 employees. Regarding the complexity, the standardization of tools, technologies, and programming languages tends to facilitate the dissemination of relevant expertise in the organization which would make the tool less useful in these kinds of scenarios.

Four out of five participants considered the tool very easy and very feasible to use. As mentioned by participant P3, the tool leverages data existing in the organization. We believe this is crucial for its feasibility since the members do not need to perform extra activities in order to provide data for expert identification. The tasks they already perform leave a trace of knowledge possession in tools, code bases, and chats, among others. Also, the participants defined the tool as "simple", "intuitive", and "straightforward". This suggests a low level of resistance to introducing the tool in real-world scenarios. However, we must consider that although this evaluation was performed with real data, the scenarios of use were hypothetical. We believe that the availability of skill types in the tool is important to its feasibility. For example, if someone rarely finds the desired skill type in the tool, they would be discouraged from keeping using it.

As for the future evolution of ExpertFY, one participant suggested that the tool could include different levels of proficiency in a skill. This relates to the used approach (*iKnow*) for creating ExpertFY since it does not address this scenario. This may be important because someone who manifested a skill the most not necessarily has a high level of proficiency. However, this makes the development of an expert finding system more complex. So, further studies are necessary to evaluate if this path is cost-effective considering the goal of supporting knowledge sharing.

The overall results of the survey indicated that ExpertFY was useful and its use was feasible for identifying who knows what in the organization. These results can be understood as preliminary evidence that *iKnow* is useful, since it was used to develop ExpertFY.

## 5.4 Threats to Validity

As any study, the study performed to evaluate *iKnow* has some limitations that may have threatened the validity of its results. Thus, these limitations must be considered together with the results. In this section, we discuss some threats involved in the study.

An important limitation is that *iKnow* was used by its author. Although this use serves as a proof of concept of the approach's feasibility, it is not enough to ensure that the approach is suitable for use in practical settings.

Another limitation to be considered is the fact that the author of this dissertation works at BMC, which may have threatened the results. On one hand, his knowledge about the organization may have favored the execution of some *iKnow* steps. On the other, the relationship of the survey participants with the author may have influenced their answers. To address these threats, we involved other members of BMC in each *iKnow* step to decrease bias and the participants were told to be free to express their opinion and were advised to be very honest because their feedback would be important to improve the tool.

The participants may also have not understood the tool's features or the motivation for using it. To mitigate this threat, we made a brief presentation about the tool and provided a document describing the tool's main features.

The questions contained in the questionnaire can also be a threat to the results. Some of them can lead to confirmation bias. We minimized this threat by asking the participants to justify their answers so that they could reflect on the given answers instead of only answering yes or no.

Another limitation refers to the fact that the study occurred in a semi-controlled environment that does not reflect a complete real-world scenario, even though we used real data. Moreover, the participants used the tool in a short period of time, and the

evaluation was only based on their perception. The small number of participants is also a threat to the results.

Considering these threats, the study results cannot be generalized and must be understood as preliminary evidence that *iKnow* is useful and its use is feasible to support the development of systems to identify who knows what for knowledge sharing purposes. Analogously, there is only initial evidence that ExpertFY is useful and its use is feasible for identifying who knows what in an organization. Therefore, these results should be complemented by further studies.

## 5.5 Concluding Remarks

This chapter presented ExpertFY, a system developed using *iKnow* as a proof of concept, demonstrating that using the proposed approach is feasible.

The findings indicated that using task outputs and tasks as skill manifestations was effective. Participants considered the suggestions made by the tool more accurate than the ones they had identified on their own. This suggests that incorporating this rationale into expert-finding systems can lead to effective knowledge-sharing activities within the organization.

ExpertFY was evaluated in a study that analyzed the usefulness and feasibility of using it to identify experts for knowledge-sharing purposes in an organization. The study provided preliminary evidence indicating that the tool is useful and its use is feasible. However, the tool must be improved in terms of user interface, security, and scalability in order to be fully prepared for complete practical usage scenarios. The results of ExpertFY evaluation also suggest that *iKnow* is useful, since it supports the development of systems to identify who knows what as the one built in this work.

## 6 Conclusion

*This chapter presents the final considerations (Section 6.1), highlights the work contributions (Section 6.2) and discusses envisioned future work (Section 6.3) to continue and improve the work proposed in this dissertation.*

### 6.1 Final Considerations

Software development is a complex and knowledge-intensive activity. Historically, organizations' knowledge has been often undocumented, being represented through the skills, experience, and knowledge of their professionals, typically tacit knowledge (RUS; LINDVALL, 2002), which makes its use and access limited and difficult (O'LEARY, 1998). In this sense, although there is a huge volume of data and information available today, people still rely on the knowledge of an expert (HUSAIN et al., 2019). However, this activity can be costly, since individuals can seek out a particular knowledge by asking people and following a trail of referrals until they locate the appropriate expert (CAMPBELL et al., 2003). Expert finding systems have been proposed to address this issue. These systems aim to identify candidate experts and rank them based on their expertise in a given subject (BALOG et al., 2012; HUSAIN et al., 2019).

Developing expert-finding systems is challenging due to the involvement of complex concepts such as expertise, skill, knowledge, and other related concepts. Additionally, integrating diverse sources of expertise poses a difficulty (HUSAIN et al., 2019). Ontologies can be helpful in this matter. An ontology is a formal and explicit specification of a shared conceptualization (STUDER; BENJAMINS; FENSEL, 1998). They can be used to capture and organize knowledge based on a common vocabulary to deal with interoperability and knowledge-related problems.

Considering this scenario, this work had the main objective of *exploring ontologies, software development tasks and artifacts, and non-technical factors to help identify who knows what in software organizations*. This main objective was detailed in four specific objectives, and all of them were achieved. Table 20 presents the specific objectives of this work and the main product that serves as evidence that the objective was achieved.

This research presents limitations that should be acknowledged. First, the proposed approach was used only by its author in a case that served as proof of concept. Although the case demonstrated that suing *iKnow* is feasible, this result cannot be extended to its use by other people in practical settings. The evaluation of ExpertFY has also limitations. It was conducted with a limited number of professionals over a short period, thus, without

Objectives	Products
Obtain a panorama of the state of the art about expert identification in the software development context.	Systematic Mapping (Chapter 3)
Propose an ontology-based approach to support identifying who knows what in software organization considering software development tasks, produced artifacts, and non-technical aspects.	<i>iKnow</i> (Chapter 4)
Develop a computational solution to support identifying who knows what based on the proposed approach.	ExpertFY (Section 5.2)
Apply the proposed solution to solve expert identification problems in practical settings.	ExpertFY Evaluation at BMC (Section 5.3)

Table 20 – Specific objectives of this work

capturing the medium and long-term impacts of the tool on knowledge-sharing practices. Additionally, the tool was evaluated within a single company, considering a limited scope, which limits the ability to generalize the findings to other contexts. Furthermore, the tool requires improvements to support a complete practical usage scenario. Due to time constraints, important features for production-ready software, such as authentication, and profile editing, are currently missing.

Concerning the proposed approach, one significant limitation is that *iKnow* does not account for the possibility that a period of reduced or no activity may lead to skill degradation. Additionally, the quality of the available data within the organization can affect the accuracy of the results. For example, if the data wrongly attributes a task to an individual who did not actually perform it, this could lead to misleading conclusions about that person’s skills. Additionally, the approach considers only data existing within the organization. This limitation poses a challenge for new hires, as they may not have had the opportunity to manifest their skills within the organization.

As for the use of data related to skill manifestations to rank expert candidates, even though expert candidates are ranked, it is possible that individuals who are better ranked do not necessarily possess more knowledge on the subject than others. During the case study, we observed the existence of automation scripts for creating application alerts, which contributed to the person responsible for executing the script being ranked higher than most individuals. This raises questions about the accuracy of the ranking depending on the skill manifestations and data considered, as it may not reflect the true level of expertise of the individuals involved.

It must be noted that, although we position our work as supporting knowledge-sharing practices among individuals, a system developed using the proposed approach

can also address other aspects beyond this focus. Given that the system would include information about the skills possessed by individuals, it could be utilized for other purposes related to knowledge, such as identifying knowledge silos within a team or evaluating the impact of losing a team member. Furthermore, data provided by the system could support the performance assessment of individuals.

Finally, it is necessary to reflect on the role expert-finding systems play in the current scenario in which IA-based systems have become more popular. In recent years, the use of large language models (LLMs) has increased significantly. Tools such as ChatGPT and GitHub Copilot are increasingly becoming integral to software organizations and are also subjects of ongoing research. The impact of these tools on knowledge-sharing practices and software activities has yet to be deeper explored. Would they be able to replace expert-finding systems? One important factor to consider is that experts within the organization may possess knowledge on subjects that are specialized for the company, such as organizational software architecture patterns. In this context, the mentioned tools may lack crucial information necessary for generating accurate responses.

## 6.2 Contributions

The main contributions of this work are:

- ***iKnow***, an ontology-based approach to support the development of systems to identify who knows what in software organizations. Practitioners and researchers can use *iKnow* to develop expert-finding systems by leveraging data related to task executions, produced artifacts, and their relationship to skills. In addition, researchers can extend or deepen some of the *iKnow* steps (e.g., to address the quality of the data used), making the approach more comprehensive and potentially enhancing the precision of expert identification.
- **ExpertFY**, the system developed using *iKnow*, which demonstrated that the proposed approach can be used to develop expert-finding systems. ExpertFY allows software developers within BMC to identify who knows what regarding the defined skill types of interest. It can also be evolved to support other skill types. Although ExpertFY was developed considering the specific BMC context, we believe that only minor modifications can be necessary for use in other organizations. Furthermore, the development process of ExpertFY can be used as an example to help other people develop similar systems using *iKnow*.
- The **systematic mapping** that investigated the state of the art and provides a panorama of expert identification in the software development context. Researchers can consider the presented panorama to identify gaps in the literature and explore

research opportunities for advancing this area. The systematic mapping main results were published in (BRAGA; JR; BARCELLOS, 2023).

### 6.3 Future Work

Considering the current stage presented here, some perspectives for future work are presented below.

The **mapping study** could be updated to verify if new works have been published reporting new approaches for expert identification in the software development context. In addition, it is important to explore how the use of generative AI tools (e.g., ChatGPT) impacts the state of the practice and the state of the art on expert-finding systems.

Regarding *iKnow*, the approach could be used for developing systems to cover other subdomains of Software Engineering and even other domains. Although our main focus was to support expert identification in software organizations, we believe that the proposed approach is domain-independent and can support this process on domains other than Software Engineering. In this sense, it is important that the approach could be used by other individuals, and their experience was reported to improve *iKnow*. This requires new studies to evaluate *iKnow*.

Furthermore, we believe that the non-technical factor can be better explored. It is necessary to define a more extensive list of possible factors, according to the literature and evaluations among software practitioners. Moreover, studies can be conducted to evaluate the extension of the impacts of each factor on knowledge sharing.

Additionally, we noticed that the identification of skill types, task types, and artifact types may not be a simple task. Therefore, it would be valuable to create a catalog of skill types in the context of Software Engineering and integrate it with the conceptualization presented in Core-O. Moreover, we believe the relation between some skill types and task types (what we call assumptions) may be independent of a particular organization and could be reused in other contexts. Another important aspect to be explored is the role of task outputs quality when evaluating an individual's skill manifested in a task. As an example, one person P1 can create a unit test containing test smells<sup>1</sup>. In this sense, has P1 still manifested a skill of unit testing applications?

It should also be considered to extend the Core-O extract considered in *iKnow*. Currently, only skill manifestations (tasks and task outputs) are used to find experts. Core-O addresses several other human aspects (e.g., attitude) that could also be considered to find an expert to share knowledge. For example, other sources of assessment, such as performance reviews, individual development plans, and 360-degree reviews, can also be

---

<sup>1</sup> Test smells are indicators of potential problems in test codes.

considered.

Concerning *ExpertFY*, it could be improved with the addition of key features: authentication and profile editing, for example. Furthermore, we recommend developing the modules according to the reference architecture to meet real-usage scenarios. Additionally, we believe that the Core Competence Module of ExpertFY can be reused in other organizations, because it handles mainly concepts related to the competence domain (skill, task, and others). Therefore, it is important to evaluate these tool in other settings. In this context, we recommend the creation of reused components for extracting data from widely used tools (e.g., JIRA, Azure DevOps). We believe this could improve the velocity of the evolution of a tool.

Finally, regarding the use of AI-based systems (particularly LLMs) as a source of knowledge, research can be conducted to investigate its use in knowledge sharing in organizations. Questions such as when to use AI-based systems, when to use expert finding systems, and when to combine both can be objects of investigation.

# Bibliography

ADAR, E. et al. Shock: Aggregating information while preserving privacy. *Information Systems Frontiers*, v. 5, n. 1, p. 15–28, 2003. ISSN 13873326. Disponível em: <<http://link.springer.com/10.1023/A:1022033619551>>. Cited on page 35.

ALAVI, M.; LEIDNER, D. E. Knowledge Management and Knowledge Management Systems: Conceptual Foundations and Research Issues. v. 25, 2001. ISSN 02767783. Cited 3 times on pages 10, 19, and 20.

ALLAHO, M. Y.; LEE, W.-C. Increasing the Responsiveness of Recommended Expert Collaborators for Online Open Projects. In: *Proc. of the 23rd ACM Int. Conf. on Conf. on Information and Knowledge Management*. ACM, 2014. p. 749–758. ISBN 978-1-4503-2598-1. Disponível em: <<https://dl.acm.org/doi/10.1145/2661829.2662032>>. Cited on page 35.

ALMEIDA, J. a. P. A.; FALBO, R. A.; GUIZZARDI, G. Events as entities in ontology-driven conceptual modeling. In: *Conceptual Modeling: 38th International Conference, ER 2019, Salvador, Brazil, November 4–7, 2019, Proceedings*. Berlin, Heidelberg: Springer-Verlag, 2019. p. 469–483. ISBN 978-3-030-33222-8. Disponível em: <[https://doi.org/10.1007/978-3-030-33223-5\\_39](https://doi.org/10.1007/978-3-030-33223-5_39)>. Cited on page 26.

ANDREWS, K. M.; DELAHAYE, B. L. Influences on knowledge processes in organizational learning: The psychosocial filter. *J. Manag. Stud.*, Wiley, v. 37, n. 6, p. 797–810, set. 2000. Cited on page 21.

ARGOTE, L.; BECKMAN, S. L.; EPPLE, D. The persistence and transfer of learning in industrial settings. *Manage. Sci.*, INFORMS, Linthicum, MD, USA, v. 36, n. 2, p. 140–154, fev. 1990. ISSN 0025-1909. Cited on page 19.

ATZBERGER, D. et al. CodeCV: Mining Expertise of GitHub Users from Coding Activities. In: . [S.l.: s.n.], 2022. Cited on page 12.

BALOG, K.; AZZOPARDI, L.; RIJKE, M. de. A language modeling framework for expert finding. *Information Processing & Management*, v. 45, n. 1, 2009. Cited 3 times on pages 12, 21, and 22.

BALOG, K. et al. Expertise retrieval. *Foundations and Trends® in Information Retrieval*, v. 6, n. 2–3, p. 127–256, 2012. ISSN 1554-0669. Disponível em: <<http://dx.doi.org/10.1561/15000000024>>. Cited 3 times on pages 11, 29, and 82.

BASIL, V. R.; CALDIERA, G.; ROMBACH, H. D. The goal question metric approach. In: . [s.n.], 1994. Disponível em: <<https://api.semanticscholar.org/CorpusID:13884048>>. Cited on page 73.

BASS, L.; CLEMENTS, P.; KAZMAN, R. *Software Architecture in Practice*. 3rd. ed. [S.l.]: Addison-Wesley Professional, 2012. ISBN 0321815734. Cited on page 57.

BORGO, S.; MASOLO, C. Foundational choices in dolce. In: *Handbook on Ontologies*. [s.n.], 2009. Disponível em: <<https://api.semanticscholar.org/CorpusID:1073021>>. Cited on page 24.

BRAGA, C.; JR, P. S.; BARCELLOS, M. Help! i need somebody. a mapping study about expert identification in software development. In: *Proceedings of the XXXVII Brazilian Symposium on Software Engineering*. New York, NY, USA: Association for Computing Machinery, 2023. (SBES '23), p. 154–163. ISBN 9798400707872. Disponível em: <<https://doi.org/10.1145/3613372.3613389>>. Cited 4 times on pages 12, 15, 29, and 85.

BRAGA, C. E. C.; JUNIOR, P. S. d. S.; BARCELLOS, M. P. *Supplementary material of the study "Help! I need somebody. A Mapping Study about Expert Identification in Software Development"*. Zenodo, 2023. Disponível em: <<https://doi.org/10.5281/zenodo.8154833>>. Cited on page 33.

BRAGA, C. E. C.; JÚNIOR, P. S. d. S.; BARCELLOS, M. P. *Supplementary material of the study "Help! I need somebody. A Mapping Study about Expert Identification in Software Development"*. Zenodo, 2023. Disponível em: <<https://doi.org/10.5281/zenodo.8154833>>. Cited on page 15.

BRINGUENTE, A.; FALBO, R.; GUIZZARDI, G. Using a foundational ontology for reengineering a software process ontology. *JIDM*, v. 2, p. 511–526, 01 2011. Cited on page 23.

CALHAU, R. et al. Core-o: A competence reference ontology for professional and learning ecosystems. In: \_\_\_\_\_. *Formal Ontology in Information Systems*. [S.l.: s.n.], 2024. (Frontiers in Artificial Intelligence and Applications), p. 16–30. Cited 7 times on pages 5, 14, 25, 27, 28, 47, and 49.

CALHAU, R. F.; FALBO, R. d. A. An ontology-based approach for semantic integration. In: *2010 14th IEEE International Enterprise Distributed Object Computing Conference*. [S.l.: s.n.], 2010. p. 111–120. Cited 2 times on pages 12 and 56.

CAMPBELL, C. S. et al. Expertise identification using email communications. In: *Proc. of the Twelfth Int. Conf. on Information and Knowledge Management*. New York, NY, USA: Association for Computing Machinery, 2003. (CIKM '03), p. 528–531. ISBN 1581137230. Disponível em: <<https://doi-org.ez43.periodicos.capes.gov.br/10.1145/956863.956965>>. Cited 3 times on pages 11, 39, and 82.

CARRARETTO, R.; ALMEIDA, J. Separating Ontological and Informational Concerns: Towards a Two-Level Model-Driven Approach. In: *2012 IEEE 16th International Enterprise Distributed Object Computing Conference Workshops*. [S.l.]: IEEE, 2012. p. 29–37. Cited on page 57.

CORDOVA-MORAS, L. G.; RODRIGUEZ-ELIAS, O. M.; SERNA-ENCINAS, M. T. Expert Location Tool for Global Software Development Environments Based on Knowledge Profile Management: A Mobile Application Approach. In: *2017 5th Int. Conf. in Software Engineering Research and Innovation (CONISOFT)*. Mérida: IEEE, 2017. p. 89–96. ISBN 978-1-5386-3956-6. Disponível em: <<https://ieeexplore.ieee.org/document/8337939/>>. Cited on page 34.

DALKIR, K. *Knowledge Management in Theory and Practice*. [S.l.: s.n.], 2013. ISBN 9780080547367. Cited on page 19.

DARR, E. D.; ARGOTE, L.; EPPLE, D. The acquisition, transfer, and depreciation of

knowledge in service organizations: Productivity in franchises. *Manage. Sci.*, INFORMS, Linthicum, MD, USA, v. 41, n. 11, p. 1750–1762, nov. 1995. ISSN 0025-1909. Disponível em: <<https://doi.org/10.1287/mnsc.41.11.1750>>. Cited on page 19.

DAVENPORT, T. H.; PRUSAK, L. *Information Ecology: Mastering the Information and Knowledge Environment*. 1st. ed. USA: Oxford University Press, Inc., 1997. ISBN 0195111680. Cited on page 21.

DAVENPORT, T. H.; PRUSAK, L. *Working knowledge: how organizations manage what they know*. Boston, Mass: Harvard Business School Press, 1998. ISBN 978-0-87584-655-2. Cited 2 times on pages 10 and 11.

DAVIS, E. Naive physics perplex. *AI Mag.*, v. 19, p. 51–79, 1997. Disponível em: <<https://api.semanticscholar.org/CorpusID:16249928>>. Cited on page 22.

DUARTE, B. B. et al. Towards an ontology of requirements at runtime. In: *Formal Ontology in Information Systems*. [s.n.], 2016. Disponível em: <<https://api.semanticscholar.org/CorpusID:2874044>>. Cited on page 24.

FALBO, R. de A. et al. Organizing ontology design patterns as ontology pattern languages. In: CIMIANO, P. et al. (Ed.). *The Semantic Web: Semantics and Big Data*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. p. 61–75. ISBN 978-3-642-38288-8. Cited 3 times on pages 5, 23, and 24.

FITZGERALD, B.; STOL, K.-J. Continuous software engineering: A roadmap and agenda. *Journal of Systems and Software*, v. 123, p. 176–189, 2017. ISSN 0164-1212. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S0164121215001430>>. Cited on page 12.

FRITZ, T. et al. Degree-of-knowledge: Modeling a developer’s knowledge of code. *ACM Trans. Softw. Eng. Methodol.*, Association for Computing Machinery, New York, NY, USA, v. 23, n. 2, apr 2014. ISSN 1049-331X. Cited on page 35.

FU, C. et al. Expert recommendation in oss projects based on knowledge embedding. In: *2017 Int. Workshop on Complex Systems and Networks (IWCSN)*. Doha: IEEE, 2017. p. 149–155. ISBN 978-1-5386-1890-5. Cited on page 34.

GANGEMI, A. et al. Sweetening ontologies with dolce. In: *Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web*. Berlin, Heidelberg: Springer-Verlag, 2002. (EKAW ’02), p. 166–181. ISBN 3540442685. Cited on page 24.

GUIZZARDI, G. *Ontological foundations for structural conceptual models*. Tese (PhD Thesis - Research UT, graduation UT) — University of Twente, out. 2005. Cited 3 times on pages 24, 25, and 26.

GUIZZARDI, G. On ontology, ontologies, conceptualizations, modeling languages, and (meta)models. In: *Proceedings of the 2007 Conference on Databases and Information Systems IV: Selected Papers from the Seventh International Baltic Conference DBIS’2006*. NLD: IOS Press, 2007. p. 18–39. ISBN 9781586037154. Cited on page 24.

GUIZZARDI, G. et al. Ufo: Unified foundational ontology. *Appl. Ontol.*, IOS Press, NLD, v. 17, n. 1, p. 167–210, jan 2022. ISSN 1570-5838. Disponível em: <<https://doi.org/10.3233/AO-210256>>. Cited on page 25.

GUIZZARDI, G.; FALBO, R. de A.; GUIZZARDI, R. Grounding software domain ontologies in the unified foundational ontology (ufo): The case of the ode software process ontology. In: *Conferencia Iberoamericana de Software Engineering*. [s.n.], 2008. Disponível em: <<https://api.semanticscholar.org/CorpusID:32490124>>. Cited 2 times on pages 24 and 25.

GUIZZARDI, G. et al. Endurant types in ontology-driven conceptual modeling: Towards ontouml 2.0. In: *International Conference on Conceptual Modeling*. [s.n.], 2018. Disponível em: <<https://api.semanticscholar.org/CorpusID:52841941>>. Cited on page 24.

GUIZZARDI, G. et al. Towards ontological foundations for conceptual modeling: The unified foundational ontology (ufo) story. *Appl. Ontology*, v. 10, p. 259–271, 2015. Disponível em: <<https://api.semanticscholar.org/CorpusID:41355553>>. Cited on page 25.

GUIZZARDI, G. et al. Towards ontological foundations for the conceptual modeling of events. In: NG, W.; STOREY, V. C.; TRUJILLO, J. C. (Ed.). *Conceptual Modeling*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. p. 327–341. ISBN 978-3-642-41924-9. Cited on page 24.

GUPTA, A.; GOVINDARAJAN, V. Knowledge management's social dimension: Lessons from nucor steel. *MIT Sloan Management Review*, v. 42, 2000. Disponível em: <<https://api.semanticscholar.org/CorpusID:151067455>>. Cited on page 21.

HANSEN, M. T.; NOHRIA, N.; TIERNEY, T. What's your strategy for managing knowledge? *Harvard business review*, v. 77 2, p. 106–16, 187, 1999. Cited on page 18.

HEVNER, A. A three cycle view of design science research. *Scandinavian Journal of Information Systems*, v. 19, 01 2007. Cited on page 13.

HUANG, C. et al. Software expert discovery via knowledge domain embeddings in a collaborative network. *Pattern Recognition Letters*, v. 130, p. 46–53, fev. 2020. ISSN 01678655. Cited on page 34.

HUGHES, G.; CROWDER, R. Experiences in Designing Highly Adaptable Expertise Finder Systems. In: *Volume 1: 23rd Computers and Information in Engineering Conf., Parts A and B*. Chicago, Illinois, USA: ASMEDC, 2003. p. 451–460. ISBN 978-0-7918-3699-6. Cited 3 times on pages 35, 59, and 60.

HUSAIN, O. et al. Expert finding systems: A systematic review. *Applied Sciences*, v. 9, n. 20, 2019. ISSN 2076-3417. Disponível em: <<https://www.mdpi.com/2076-3417/9/20/4250>>. Cited 8 times on pages 7, 11, 12, 22, 29, 50, 51, and 82.

IPE, M. Knowledge sharing in organizations: A conceptual framework. *Human Resource Development Review*, v. 2, n. 4, p. 337–359, 2003. Disponível em: <<https://doi.org/10.1177/1534484303257985>>. Cited 4 times on pages 5, 20, 21, and 53.

ISRAILIDIS, J.; SIACHOU, E.; KELLY, S. Why organizations fail to share knowledge: an empirical investigation and opportunities for improvement. *Information Technology & People*, ahead-of-print, 09 2020. Cited on page 41.

JANES, A.; SILLITTI, A.; SUCCI, G. Non-invasive Software Process Data Collection for Expert Identification. In: *Int. Conf. on Software Engineering and Knowledge Engineering*. [S.l.: s.n.], 2008. Cited on page 35.

JONES, P.; JORDAN, J. Knowledge orientations and team effectiveness. *Int. J. Technol. Manag.*, Inderscience Publishers, v. 16, n. 1/2/3, p. 152, 1998. Cited on page 21.

KITCHENHAM, B.; CHARTERS, S. *Guidelines for performing systematic literature reviews in software engineering*. [S.l.], 2007. Cited 2 times on pages 29 and 32.

KRÜGER, J. et al. Do you remember this source code? In: *Proc. of the 40th Int. Conf. on Software Engineering*. New York, NY, USA: Association for Computing Machinery, 2018. (ICSE '18), p. 764–775. ISBN 9781450356381. Cited on page 41.

LANDIS, J. R.; KOCH, G. G. The measurement of observer Agreement for Categorical Data. *Biometrics*, v. 33, n. 1, p. 159, mar. 1977. ISSN 0006341X. Cited on page 44.

LIN, S. et al. A survey on expert finding techniques. *Journal of Intelligent Information Systems*, v. 49, 10 2017. Cited 2 times on pages 11 and 29.

LONG, D. W. D.; FAHEY, L. Diagnosing cultural barriers to knowledge management. *Acad. Manag. Perspect.*, Academy of Management, v. 14, n. 4, p. 113–127, nov. 2000. Cited on page 21.

LUCAS, E. M. et al. Knowledge-Oriented Models Based on Developer-Artifact and Developer-Developer Interactions. *IEEE Access*, v. 8, p. 218702–218719, 2020. ISSN 2169-3536. Cited 6 times on pages 11, 12, 34, 48, 59, and 60.

MANGARAVITE, V. et al. The lexr collection for expertise retrieval in academia. *Proc. of the 39th Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, 2016. Cited on page 11.

MCDONALD, D. W.; ACKERMAN, M. S. Just talk to me: A field study of expertise location. In: *Proc. of the 1998 ACM Conf. on Computer Supported Cooperative Work*. Association for Computing Machinery, 1998. (CSCW '98), p. 315–324. ISBN 1581130090. Disponível em: <<https://doi.org/10.1145/289444.289506>>. Cited on page 11.

MEALY, G. H. Another look at data. In: *AFIPS '67 (Fall)*. [s.n.], 1967. Disponível em: <<https://api.semanticscholar.org/CorpusID:18146748>>. Cited on page 22.

MENAHA, R.; JAYANTHI, V. Finding experts in community question answering system using trie string matching algorithm with domain knowledge. *IETE Journal of Research*, v. 70, n. 3, p. 2602 – 2614, 2024. Cited by: 0. Disponível em: <<https://www.scopus.com/inward/record.uri?eid=2-s2.0-85149288001&doi=10.1080%2f03772063.2023.2181233&partnerID=40&md5=d586067abe00602c8626b3fd8b0e8ec0>>. Cited 2 times on pages 22 and 48.

MO, W. et al. Geminer: Mining social and programming behaviors to identify experts in github. *Proc. of the 7th Asia-Pacific Symposium on Internetware*, 2015. Cited on page 35.

MOCKUS, A.; HERBSLEB, J. Expertise browser: a quantitative approach to identifying expertise. In: *Proc. of the 24th Int. Conf. on Software Engineering. ICSE 2002*. [S.l.: s.n.], 2002. p. 503–512. Cited 2 times on pages 12 and 35.

MORAES, A. et al. Recommending experts using communication history. In: *Proc. of the 2nd Int. Workshop on Recommendation Systems for Software Engineering*. Cape Town South Africa: ACM, 2010. p. 41–45. ISBN 978-1-60558-974-9. Disponível em: <<https://dl.acm.org/doi/10.1145/1808920.1808929>>. Cited 2 times on pages 12 and 35.

- MORALES-RAMIREZ, I. et al. Exploiting Online Discussions in Collaborative Distributed Requirements Engineering. In: *Int. i\* Workshop*. [S.l.: s.n.], 2015. Cited 3 times on pages 34, 59, and 60.
- NAEEM, M.; KHAN, M. B.; AFZAL, M. T. Expert discovery: A web mining approach. *Journal of AI and Data Mining*, v. 1, p. 35–47, 2013. Cited on page 11.
- NARDI, J. C.; FALBO, R. de A.; ALMEIDA, J. P. A. Foundational ontologies for semantic integration in eai: A systematic literature review. In: DOULIGERIS, C. et al. (Ed.). *Collaborative, Trusted and Privacy-Aware e/m-Services*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. p. 238–249. ISBN 978-3-642-37437-1. Cited on page 43.
- NARDI, J. C.; FALBO, R. de A.; ALMEIDA, J. P. A. A panorama of the semantic eai initiatives and the adoption of ontologies by these initiatives. In: SINDEREN, M. van et al. (Ed.). *Enterprise Interoperability*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. p. 198–211. ISBN 978-3-642-36796-0. Cited on page 23.
- NONAKA, I. A dynamic theory of organizational knowledge creation. *Organization Science*, v. 5, p. 14–37, 1994. Disponível em: <<https://api.semanticscholar.org/CorpusID:17219859>>. Cited 3 times on pages 5, 18, and 19.
- NONAKA, I.; TAKEUCHI, H. *The knowledge-creating company: how Japanese companies create the dynamics of innovation*. New York: Oxford University Press, 1995. ISBN 978-0-19-509269-1. Cited 2 times on pages 10 and 18.
- NORAMBUENA, I. N.; BERGEL, A. Building a bot for automatic expert retrieval on discord. In: *Proc. of the 5th Int. Workshop on Machine Learning Techniques for Software Quality Evolution*. Athens Greece: ACM, 2021. p. 25–30. ISBN 978-1-4503-8625-8. Cited 2 times on pages 32 and 34.
- O’LEARY, D. Enterprise knowledge management. *Computer*, v. 31, n. 3, p. 54–61, mar. 1998. ISSN 00189162. Disponível em: <<http://ieeexplore.ieee.org/document/660190/>>. Cited 2 times on pages 10 and 82.
- OLIVEIRA, J. et al. Can Source Code Analysis Indicate Programming Skills? A Survey with Developers. In: . Cham: [s.n.], 2022. v. 1621. Cited on page 12.
- PAN, S. L.; SCARBROUGH, H. Knowledge management in practice: An exploratory case study. Informa UK Limited, v. 11, n. 3, p. 359–374, set. 1999. Cited on page 21.
- PEFFERS, K. et al. A design science research methodology for information systems research. *J. Manage. Inf. Syst.*, M. E. Sharpe, Inc., USA, v. 24, n. 3, p. 45–77, dec 2007. ISSN 0742-1222. Disponível em: <<https://doi.org/10.2753/MIS0742-1222240302>>. Cited on page 13.
- PETERSEN, K.; VAKKALANKA, S.; KUZNIARZ, L. Guidelines for conducting systematic mapping studies in software engineering: An update. *Information and Software Technology*, p. 1–18, ago. 2015. ISSN 09505849. Cited on page 29.
- POKRAEV, S. *Model-driven semantic integration of service-oriented applications*. Tese (Doutorado) — University of Twente, out. 2009. Cited on page 43.
- RASDI, R. M.; TANGARAJA, G. Knowledge-sharing behaviour in public service

- organisations: determinants and the roles of affective commitment and normative commitment. *European Journal of Training and Development*, v. 46, n. 3/4, p. 337–355, maio 2022. ISSN 2046-9012, 2046-9012. Disponível em: <<https://www.emerald.com/insight/content/doi/10.1108/EJTD-02-2020-0028/full/html>>. Cited on page 11.
- REGINATO, C. et al. Go-for: A goal-oriented framework for ontology reuse. In: *2019 IEEE 20th International Conference on Information Reuse and Integration for Data Science (IRI)*. [S.l.: s.n.], 2019. p. 99–106. Cited on page 23.
- RUS, I.; LINDVALL, M. Knowledge management in software engineering. *IEEE Software*, v. 19, n. 3, p. 26–38, maio 2002. ISSN 0740-7459. Disponível em: <<http://ieeexplore.ieee.org/document/1003450/>>. Cited 3 times on pages 10, 19, and 82.
- RUY, F. B. *Software Engineering Standards Harmonization: An Ontology-Based Approach*. Tese (Doutorado) — Ph. D. Dissertation. UFES, 2017. Cited 2 times on pages 5 and 24.
- RUY, F. B. et al. Seon: A software engineering ontology network. In: *International Conference Knowledge Engineering and Knowledge Management*. [s.n.], 2016. Disponível em: <<https://api.semanticscholar.org/CorpusID:38873106>>. Cited on page 24.
- SANTOS, A. J. dos. *Um Sistema de Apoio à Identificação de Especialistas para Compartilhamento de Conhecimento no contexto de Desenvolvimento de Software*. [S.l.], 2024. Cited on page 70.
- SCHERP, A. et al. Designing core ontologies. *Appl. Ontol.*, IOS Press, NLD, v. 6, n. 3, p. 177–221, aug 2011. ISSN 1570-5838. Cited on page 23.
- SCHETTINO, V. et al. Towards Community and Expert Detection in Open Source Global Development. In: . [S.l.]: IEEE, 2019. ISBN 978-1-72810-350-1. Cited 3 times on pages 34, 59, and 60.
- SCHNEIDER, K. *Experience and Knowledge Management in Software Engineering*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. ISBN 978-3-540-95879-6 978-3-540-95880-2. Cited on page 10.
- SEID, D. Y.; KOBSA, A. Expert-finding systems for organizations: Problem and domain analysis and the demoir approach. *Journal of Organizational Computing and Electronic Commerce*, v. 13, p. 1 – 24, 2003. Cited on page 11.
- SEMERTZAKI, E. Book. *Special libraries as knowledge management centres / Eva Semertzaki*. [S.l.]: Chandos Oxford, 2011. xxii, 314 p. ; p. ISBN 9781843346135 1843346133. Cited 2 times on pages 10 and 18.
- SHAMI, N. S. et al. 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*. Sanibel Island, Florida, USA: ACM Press, 2007. p. 379. ISBN 978-1-59593-845-9. Cited 2 times on pages 32 and 35.
- SINGH, A.; KUKREJA, V.; KUMAR, M. An empirical study to design an effective agile knowledge management framework. *Multimedia Tools and Applications*, v. 82, n. 8, p. 12191–12209, mar. 2023. ISSN 1380-7501, 1573-7721. Cited on page 10.

SOMMERVILLE, I. *Software Engineering*. [S.l.]: Pearson, 2016. (Always learning). ISBN 9780133943030. Cited on page 59.

STANDARDIZATION, I. O. for et al. *ISO/IEC/IEEE 12207: 2017(E) First Edition 2017-11: ISO/IEC/IEEE Int. Standard - Systems and Software Engineering – Software Life Cycle Processes*. [S.l.]: IEEE, 2017. (IEEE Std). ISBN 9781504442534. Cited 3 times on pages 31, 38, and 44.

STUDER, R.; BENJAMINS, V.; FENSEL, D. Knowledge engineering: Principles and methods. *Data Knowledge Engineering*, v. 25, n. 1, p. 161–197, 1998. ISSN 0169-023X. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S0169023X97000566>>. Cited 4 times on pages 12, 23, 47, and 82.

TAHER, A.; VAHDATIKHAKI, F.; HAMMAD, A. Formalizing knowledge representation in earthwork operations through development of domain ontology. *Engineering, Construction and Architectural Management*, Emerald, v. 29, n. 6, p. 2382–2414, jun. 2022. ISSN 0969-9988. Publisher Copyright: © 2021, Emerald Publishing Limited. Cited on page 23.

TEUSNER, R.; MATTHIES, C.; GIESE, P. 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)*. Prague, Czech Republic: IEEE, 2017. p. 418–425. ISBN 978-1-5386-0592-9. Cited 3 times on pages 34, 59, and 60.

TRURAN, W. R. Pathways for knowledge: How companies learn through people. *Eng. Manag. J.*, Informa UK Limited, v. 10, n. 4, p. 15–20, dez. 1998. Cited on page 21.

WACHE, H. et al. 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*. Cited on page 43.

Wasko; Faraj. Why Should I Share? Examining Social Capital and Knowledge Contribution in Electronic Networks of Practice. *MIS Quarterly*, v. 29, n. 1, p. 35, 2005. ISSN 02767783. Disponível em: <<https://www.jstor.org/stable/10.2307/25148667>>. Cited on page 11.

WEISS, D.; SHANTEAU, J. Empirical assessment of expertise. *Human factors*, v. 45, p. 104–16, 02 2003. Cited on page 11.

WEISS, L. M. Collection and connection: The anatomy of knowledge sharing in professional service firms. *Acad. Manag. Proc.*, Academy of Management, v. 1999, n. 1, p. A1–A6, ago. 1999. Cited on page 20.

WIERINGA, R. et al. Requirements engineering paper classification and evaluation criteria: A proposal and a discussion. *Requir. Eng.*, v. 11, p. 102–107, mar. 2006. Cited 3 times on pages 30, 35, and 44.

WOHLIN, C. et al. *Experimentation in Software Engineering*. [S.l.]: Springer Publishing Company, Incorporated, 2012. ISBN 3642290434. Cited on page 43.

# Appendix

ID	Assumptions
A1	The skill type <u>Optimizing SQL query</u> is manifested by the task type <u>Optimize SQL queries</u> which produces the artifact type <u>Optimized SQL query</u> .
A2	The skill type <u>Monitoring application</u> is manifested by the task type <u>Create or modify alerts</u> which produces the artifact type <u>Application alerts</u> .
A3	The skill type <u>Programming unit test</u> is manifested by the task type <u>Create or modify unit tests</u> which produces the artifact type <u>Unit test cases</u> .
A4	The skill type <u>Using .NET library</u> is manifested by the task type <u>Add the library reference in a code repository</u> which produces the artifact type <u>Library reference in a class</u> .
A5	The skill type <u>Using .NET library</u> is manifested by the task type <u>Create or modify a file referencing the library</u> which produces the artifact type <u>Library referencing application</u> .
A6	The skill type <u>Designing Rest API</u> is manifested by the task type <u>Create or modify a controller class</u> which produces the artifact type <u>Controller class</u> .
A7	The skill type <u>Documenting software</u> is manifested by the task type <u>Create or modify wiki pages</u> which produces the artifact type <u>Software documentation</u> .

Table 21 – Skill type manifestation assumptions

## A Artifacts used in Application of iKnow

### A.1 Introduction

This appendix presents information about the survey performed to evaluate seven assumptions that represent task types as manifestations of skill types (and, thus, are indications of knowledge). The survey additionally aimed to evaluate non-technical factors that software developers within the BMC consider relevant. The participants were 11 BMC software developers. The participants' profile was identified through questions regarding their education level, amount of time working in software development, role, and amount of time working at BMC. The participants were asked about their level of agreement with each assumption.

### A.2 Assumptions and Non-Technical Factors

The assumptions evaluated in BMC are presented in Table 21. The assumptions were simplified in the survey forms by omitting the Artifact Types to facilitate participants' understanding.

Additionally, the participants had to check which of these options they considered as relevant non-technical factors:

- Availability;

- Social bond;
- Language;
- Team or organizational department;
- History of effective knowledge sharing;
- Seniority;
- Years of experience;
- Years of service in the organization.

The applied form is presented in figures 29, 30, 31, and 32.

### A.3 Results

Nine (81.8%) developers had bachelor's degrees and two (18.2%) had also a specialization course. Regarding the experience working with software development, two (18.2%) had between 1 and 3 years of experience, two (18.2%) had between 3 and 5 years of experience, four (36.4%) had between 5 and 10 years of experience, and 3 (27.3%) had more than 10 years of experience. Considering the role they play, five (45.5%) considered themselves backend developers, two (18.2%) considered themselves frontend developers, and four (36.4%) considered themselves fullstack developers. Two (18.2%) of the participants had worked at BMC for less than 1 year, five (45.5%) had worked at BMC for between 1 and 3 years, two (18.2%) had worked at BMC for between 5 and 10 years, and two (18.2%) had worked at BMC for more than 10 years.

For each assumption, participants responded if they *Strongly Agree*, *Agree*, *Disagree*, *Strongly Disagree* or *Don't Know*. For assumption A1, four (36.4%) participants *Strongly Agree*, and seven (36.6%) *Agree*. Regarding A2, three (27.3%) participants *Strongly Agree*, six (54.5%) *Agree*, one (9.1%) *Disagree*, and one (9.1%) *Strongly Disagree*. When evaluating A3, three (27.3%) participants *Strongly Agree*, two (18.2%) *Agree*, four (36.4%) *Disagree*, and two (18.2%) *Strongly Disagree*. For assumption A4, three (27.3%) participants *Strongly Agree*, three (27.3%) *Agree*, four (36.4%) *Disagree*, and one (9.1%) *Strongly Disagree*. Regarding assumption A5, two (18.2%) participants *Strongly Agree*, six (54.5%) *Agree*, and three (27.3%) *Disagree*. Evaluating A6, one (9.1%) participant *Strongly Agree*, five (45.5%) *Agree*, three (27.3%) *Disagree*, and two (18.2%) *Strongly Disagree*. At last, regarding A7, two (18.2%) participants had answered *Strongly Agree*, four (36.4%) *Agree*, four (36.4%) *Disagree*, and one (9.1%) *Strongly Disagree*. The summarized results are presented in Figure 33.

In order to calculate an overall score for each assumption, we assigned a weight for each answer as presented in Table 22, and calculated the average score for each assumption



## Who knows what identification and knowledge sharing

This study is being conducted as part of the master's research of student Carlos Eduardo Correa Braga, carried out within the Graduate Program in Informatics at the Federal University of Espírito Santo (UFES), advised by the Professor Monalessa Perini Barcellos.

The purpose of this study is to investigate aspects (e.g., data and characteristics) considered relevant for supporting the identification of who knows what within an organization and facilitating knowledge sharing on Software Engineering topics.

Thank you.

Carlos Eduardo Correa Braga

[Faça login no Google](#) para salvar o que você já preencheu. [Saiba mais](#)

\* Indica uma pergunta obrigatória

### Email

Identification is not mandatory. However, by identifying yourself, you contribute to the management of the study, enabling the addressing of any questions if necessary.

Sua resposta

### Academic degree \*

Choose your highest academic degree among the options below.

- Elementary school
- High school
- Bachelor's degree
- Specialization
- Masters' degree
- Ph.D. degree

### How many years of experience do you have in software projects? \*

- Less than 1 year
- Between 1 and 3 years
- Between 3 and 5 years
- Between 5 and 10 years
- More than 10 years

Marque o seu nível de concordância com cada uma das assertivas abaixo. \*

In case you do not know, please select the option "Do not know".

With this question, we aim to verify whether the execution of a certain task serves as an indication that the individual performing it possesses knowledge about a given topic. For example, whether the number of class diagrams created by a person is indicative of their knowledge of class diagram modeling. The statements below consider data that can be extracted from repositories of tools used by developers throughout the software development process.

	Strongly Agree	Agree	Disagree	Strongly disagree	Do not know
The number of tasks of type Optimize SQL queries performed by an individual serves as an indicator of their knowledge on Optimizing SQL query.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The number of tasks of type Create or modify alerts performed by an individual serves as an indicator of their knowledge on Monitoring application.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The number of tasks of type Create or modify unit tests performed by an individual serves as an indicator of their knowledge on Programming unit test.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The number of tasks of type Add the library reference in a code repository performed by an individual serves as an indicator of their knowledge on Using .NET library.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The number of					

**Other Evidences**

Below, for each of the topics addressed in the previous question, identify what additional data, beyond those already mentioned, you believe could serve as an indicator of how much knowledge a person has about the topic in question.

Optimizing SQL query. \*

I do not know

Outro: \_\_\_\_\_

Monitoring application. \*

I do not know

Outro: \_\_\_\_\_

Programming unit test. \*

I do not know

Outro: \_\_\_\_\_

Using .NET library. \*

I do not know

Outro: \_\_\_\_\_

Designing Rest API. \*

I do not know

Outro: \_\_\_\_\_

Documenting software. \*

I do not know

Outro: \_\_\_\_\_

Figure 31 – Assumptions Evaluation Form (part 2)

**Characteristics that influence Knowledge Sharing**

In this section, we aim to identify characteristics that are important for knowledge sharing among individuals. For example, a person with extensive knowledge of class diagram modeling may struggle to share that knowledge with someone who does not understand their language. In this sense, in addition to knowledge of a given topic, language would be a relevant characteristic for facilitating knowledge transfer.

What characteristics do you consider relevant for a person to be able to share knowledge with others?

Availability

Social bond

Language

Team or organizational department

History of effective knowledge sharing

Seniority

Years of experience

Years of service in the organization

Outro: \_\_\_\_\_

Use the question below to record any comments you find relevant (for example, regarding any responses you provided to a question) if you find it necessary.

Sua resposta \_\_\_\_\_

Figure 32 – Non-Technical Factor Selection Form

Option	Score
Strongly Agree	2
Agree	1
Don't Know	0
Disagree	-1
Strongly Agree	-2

Table 22 – Weighted Options

and answer. Therefore, the result may range from -2 to 2 with negative values indicating that the participants disagree and positive values indicating that they agree. Assumption A1 had an overall score of 1.36, A2 had a score of 0.81, A3 had a score of 0, A4 had a score of 0.2, A5 a score of 0.63, A6 0, and A7 a score of 0.18. These results are presented in Table 23 in descending order by score.

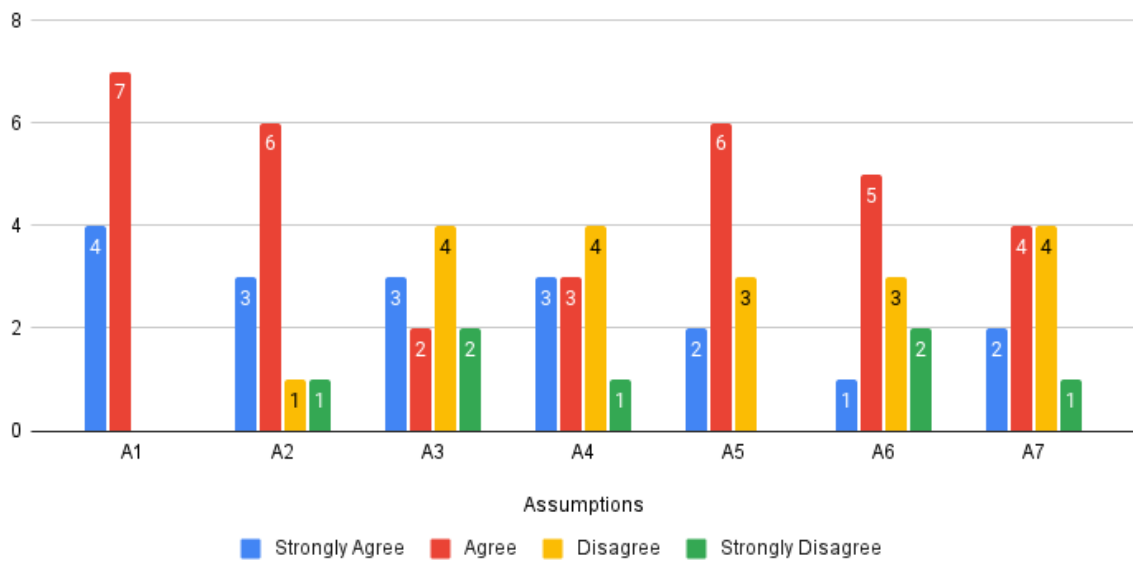


Figure 33 – Summarized results of assumptions evaluation



Assumption	Score
A1	1.36
A2	0.82
A5	0.64
A4	0.27
A7	0.18
A3	0.00
A6	0.00

Table 23 – Assumptions score sorted descending by score

Regarding the non-technical factors, the survey included various factors, with the following number of participants for each: seven participants identified "Availability", six participants "Years of experience", and six also mentioned "Seniority". Four participants reported "Years of service in the organization", while four identified "Language". Additionally, four participants mentioned "History of effective knowledge sharing". Only one participant selected "Social bond," and one indicated "Team or organizational department". Two participants specified other relevant factors.

## B Artifacts used in ExpertFY Evaluation Study

### B.1 Instructions Document

	<p>UFES (Universidade Federal do Espírito Santo)          NEMO (Núcleo de Estudos em Modelagem Conceitual e Ontologias)</p> <p><i>Evaluation study of a system to support who knows what identification within a software organization.</i></p>	
---	---	---

**Student:** Carlos Eduardo Correa Braga

**Advisor:** Monalessa Perini Barcellos (UFES)

#### Consent Form

This study is being conducted as part of the master's research of student Carlos Eduardo Correa Braga, carried out within the Graduate Program in Informatics at the Federal University of Espírito Santo (UFES), advised by the Professor Monalessa Perini Barcellos. The aim of the study is to evaluate the use of a computational tool to support the identification of who knows what in an organization, in the context of software development.

After accepting this consent form, a few questions will be presented. The confidentiality of the individual data provided in the study is guaranteed. The data will be used solely for research purposes and will not be used for personal or professional evaluation. The name of the participants (optional) and email address are requested only for contact purposes, should it be necessary to clarify any of the responses.

Despite being invited, participation in this study is voluntary, being entitled not to want to participate or abandon the study at any time.

#### Introduction

Knowledge is the combination of experience, values, information with context, and expert insights that form a foundation for assessing and incorporating new information. It can be either explicit or tacit. Explicit knowledge is structured and, therefore, easy to share. In contrast, tacit knowledge is difficult to formalize and communicate as it resides in people's minds and is the result of individual and subjective experiences. Knowledge management (KM) in an organization is essential for effectively capturing, distributing, and utilizing existing knowledge, promoting a culture that supports information creation and sharing.

In software development, finding experts with the necessary knowledge is crucial but can be challenging due to the vast amount of available data and the subjective nature of expert knowledge. In this context, within the scope of the research in which this study is being conducted, a computational tool has been proposed to support the identification of who knows what in an organization by using tasks performed and artifacts created as evidence. That is, experts are identified by considering the number of tasks completed and artifacts produced related to the topic of interest, which are seen as evidence that the person possesses a certain skill and, therefore, has knowledge on the subject.

The tool allows members of an organization to search for people for whom there is evidence that they possess a particular skill. The tool's features are presented below.

**Search Who Possess a Skill:** this use case aims to allow users searching for a *Person* who possess a given *Skill*. This use case also offers the possibility of using an advanced filter allowing the user to filter the results for specific *Attitudes* or other characteristics - such as seniority, area, and others.

**Endorse Someone's Skill:** this use case aims to record endorsements of a Person's Skill. The user can endorse Skills previously manifested through Tasks. Endorsements work as Evidence of Skills.

**View Profile:** this use case aims to allow the user to view someone's profile, including their manifested skills, received endorsements, performed tasks, and related artifacts.

## Instructions

The objective of this study is to evaluate the use of a computational tool to support the identification of who knows what in an organization within the context of software development, considering its feasibility and usefulness. It is important to emphasize that this study does NOT focus on evaluating usability aspects of the tool, but rather on how well the tool supports identifying who knows what. To conduct the study, consider the following scenarios:

Scenario 1: You need to create an alert to identify when an application under your responsibility is behaving anomalously. However, you do not have the necessary knowledge to perform this task. Scenario 2: You need to optimize an SQL database query that has been causing performance issues in an application for which you are responsible. However, after several attempts, you have still not achieved the expected optimization result. Scenario 3: You have been experiencing an issue when using the MassTransit library. Despite several attempts and online searches, you have not been able to resolve the problem.

The following steps outline the process to be followed. This order is crucial and essential for the validity of the study. Therefore, do not perform the steps out of the

proposed order.

### Step 1

For each scenario, using only your prior knowledge, you must identify a maximum of 5 people in your organization whom you believe can provide you with useful knowledge to meet your needs. To assist you, please fill out the following table.

	<b>Scenario 1</b>	<b>Scenario 2</b>	<b>Scenario 3</b>
1			
2			
3			
4			
5			

### Step 2

Now, for each scenario, using the presented tool, you must identify a maximum of 5 people in your organization whom you believe can provide you with useful knowledge to meet your needs.

	<b>Scenario 1</b>	<b>Scenario 2</b>	<b>Scenario 3</b>
1			
2			
3			
4			
5			

### Step 3


After completing the previous steps, you should fill out the questionnaire available at <https://forms.gle/E8yYkCPzdamfZ92V6>.


*Thank you.*

## B.2 Consent Form

### Evaluation Study

Study evaluating the use of a computational tool to support the identification of 'who knows what' in an organization, in the context of software development.

caducbraga@gmail.com [Switch account](#) 

 Not shared

#### Consent Form

This study is being conducted as part of the master's research of student Carlos Eduardo Correa Braga, carried out within the Graduate Program in Informatics at the Federal University of Espírito Santo (UFES), advised by the Professor Monalessa Perini Barcellos. The aim of the study is to evaluate the use of a computational tool to support the identification of 'who knows what' in an organization, in the context of software development.

After accepting this consent form, a few questions will be presented. The confidentiality of the individual data provided in the study is guaranteed. The data will be used solely for research purposes and will not be used for personal or professional evaluation. The name of the participants (optional) and email address are requested only for contact purposes, should it be necessary to clarify any of the responses.

Despite being invited, participation in this study is voluntary, being entitled not to want to participate or abandon the study at any time.

By clicking "Next", I declare that I am over 18 years old and voluntarily participate in this study, having read the information contained in this form before participating

### B.3 Participants Profile Form

**Participant Profile**

Name (optional)

Your answer \_\_\_\_\_

Email \*

da@moa.com \_\_\_\_\_

Academic degree \*

Choose your highest academic degree among the options below

Elementary school

High school

Bachelor's degree

Specialization

Masters' degree

Ph.D. degree

How many years of experience do you have in software projects? \*

Less than 1 year

Between 1 and 3 years

Between 3 and 5 years

Between 5 and 10 years

More than 10 years

What is your current job position? \*

- Backend developer
- Frontend developer
- Fullstack developer
- Solutions architect
- Other: \_\_\_\_\_

How long have you been working at your current organization? \*

- Less than 1 year
- Between 1 and 3 years
- Between 3 and 5 years
- Between 5 and 10 years
- More than 10 years

How important do you consider knowledge sharing among individuals to assist in software development tasks? \*

- Very important
- Important
- Somewhat important
- Not important

How often do you seek out others for useful knowledge to complete your tasks? \*

- Always (almost daily)
- Frequently (1 to 2 times a week)
- Sometimes (1 to 2 times a month)
- Rarely (less than once a month)
- Never

## B.4 Feedback Questionnaire

**Feedback**

What is your perception of the usefulness of the tool? \*

- Very Useful
- Useful
- Somewhat Useful
- Not Useful

Justification: \*

Your answer

Do you consider that the use of the tool for identifying "who knows what" contributed to any of the following results? \*

	Yes	No
LESS EFFORT employed to identify "who knows what"	<input type="radio"/>	<input type="radio"/>
LESS TIME spent to identify "who knows what"	<input type="radio"/>	<input type="radio"/>
CREATER ACCURACY in "who knows what" identification	<input type="radio"/>	<input type="radio"/>
GREATER COVERAGE of "who knows what"	<input type="radio"/>	<input type="radio"/>

Justification: \*

Your answer

What was your perception of the tool's ease of use? \*

- Very easy
- Easy
- Hard
- Very hard

Justification: \*

Your answer \_\_\_\_\_

What was your perception regarding the feasibility of using the tool? \*

- Very feasible
- Feasible
- Infeasible
- Very infeasible

Justification: \*

Your answer \_\_\_\_\_

Would you recommend the use of the tool to other people? \*

- Yes
- No

Justification: \*

Your answer \_\_\_\_\_

Do you believe that using the proposed tool would encourage you to reach out to others in order to obtain useful knowledge to complete your tasks? \*

Yes

No

Justification: \*

Your answer

Based on your experience using the tool, what BENEFITS do you believe it would bring if used in your organization? \*

Your answer

Based on your experience using the tool, what DIFFICULTIES do you believe it would bring if used in your organization? \*

Your answer

Please share your suggestions for improvements or further development of the tool, as well as any other comments you consider relevant.

Your answer