An Ontology-based Approach for Developing Systems to Identify Who Knows What in Software Organizations

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

Abraão Jesus dos Santos
Ontology & Conceptual Modeling Research Group (NEMO), Computer Science Department
Federal University of Espírito Santo
Vitoria, ES, Brazil
abraao.dev.pub@gmail.com

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

ABSTRACT

Context: Software development is a knowledge-intensive process, 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. Thus, 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's 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 required knowledge. Expert-finding systems address this issue by identifying and ranking experts. Developing such systems is challenging and involves complex concepts such as expertise, skill, and knowledge, among others. 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. *Objective:* We aim to help organizations develop expert-finding systems to identify who knows what in a software organization considering data stored in the organization repositories and non-technical factors that may affect knowledge sharing. *Method:* We developed *iKnow*, an ontology-based approach that supports developing systems that use data on skill manifestations to identify who knows what. *Results: iKnow* was applied to develop the *ExpertFY* system, which was used by members of a software organization. *ExpertFY* helped identify experts the members had not identified themselves. *Conclusion:* the results suggest that *iKnow* is a useful and promising approach to developing expert-finding systems and *ExpertFY* improves the identification of who knows what.

KEYWORDS

Expert-Finding System, Skill, Knowledge Sharing, Ontology

1 Introduction

Knowledge is a human specialty stored in people's minds, acquired through experience and interaction with their environment [32]. Historically, organizations' knowledge has often been undocumented and represented through the skills, experience, and knowledge of their professionals, typically tacit knowledge [30]. This kind of knowledge is highly personal and, thus, hard to formalize and communicate to others [28], which makes its use and access limited and difficult [29].

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 [1]. In particular, knowledge sharing (or knowledge transfer) is the process of transferring knowledge between individuals, groups, or organizations [1].

Software development is a complex and knowledge-intensive activity. Agile methods, with their focus on tacit knowledge, have increased the importance of knowledge resources and knowledge sharing. Sharing experiences and knowledge can improve the ability of teams to handle uncertain and ambiguous situations, which is essential for the success of software projects [35]. 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 has knowledge and displays expertise in a specific domain/topic 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 [40]; an individual who has a great deal of knowledge and displays comprehensive skills in a specific area [18]; and a person who has the best knowledge and experience in a particular topic [26]. In this work, we consider experts to be people who have knowledge of a topic and may have different levels of expertise. Moreover, in 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 [11, 17].

Although nowadays there is a huge volume of data and information available, people often seek the knowledge and guidance of an expert and require comprehensive information regarding experts who can answer their questions [17]. Moreover, some cultural aspects are usually not incorporated into the data and information provided by automated tools, and interaction among people contributes to getting them closer and to a better organizational climate. This kind of knowledge sharing depends on finding a suitable expert, i.e., a person who possesses the desired knowledge and is willing to share it. Finding experts is an important task and has attracted much attention [18].

Identifying experts in software organizations is challenging, especially in the context of large-scale distributed software projects [20, 21, 33]. Individuals can seek particular knowledge by asking people and following a trail of referrals until they locate the appropriate expert [8]. This approach can incur significant costs, such as the effort repeated by different people looking for the same answers, and miscommunication that leads to taking the advice of not-so-experts who are found quickly [8]. Moreover, knowledge possession is not the only factor that influences expert responsiveness. Other factors are also important, such as availability [23], communication skills, and interest in building a reputation [39].

Data overload and the specific nature of expert knowledge can hinder the identification of suitable experts. Expert-finding systems aim to address this challenge. However, developing such systems is complex and involves numerous abstract concepts such as expertise, skill, and knowledge. Additionally, integrating diverse sources containing data related to expertise poses a difficulty [17]. The sources are often heterogeneous, with little concern with sharing and integration aspects, leading to several semantic conflicts [7].

Ontologies can be helpful in this matter. An ontology is a formal and explicit specification of a shared conceptualization [36]. 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 context of expert search, they can help by providing a common understanding of concepts related to knowledge and experts. Moreover, they can be used as an interlingua to map the concepts used by different sources, enabling data understanding and integration [7].

Some works have been proposed to support finding experts, such as [19, 22, 24]. 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 used to identify experts, as well as difficulties in integrating data from different sources. The latter may cause the identification of experts who possess the desired knowledge but are not the most suitable ones to share it with a particular seeker [4].

To bridge these gaps, we proposed *iKnow*, an ontology-based approach to support the development of systems to identify who knows what in software organizations. *iKnow* considers non-technical factors and different artifacts and tasks as skill manifestations (i.e., evidence of that skill) that help identify experts. It was used to develop the *ExpertFY* (Expert For You) system. The results suggest that *iKnow* is useful and its use is feasible, and *ExpertFY* helps people identify who knows what in an organization.

This paper introduces *iKnow*. Section 2 provides a brief background for the paper, Section 3 presents *iKnow*, Section 4 regards the application of *iKnow* in a practical setting, Section 5 discusses related work, and Section 6 presents our final considerations.

2 Background

Knowledge and Knowledge Management (KM). KM is the deliberate and systematic coordination of the organization's people, technology, processes, and structure to add value through knowledge reuse and innovation [10]. KM enables the organization to *know how, know where, know who, know what, know when, and know why* [30]. It involves both explicit and tacit knowledge.

Explicit knowledge is structured and formal [34], and it can be easily communicated and shared. Tacit knowledge, in turn, is hard to formalize and communicate to others since it is highly personal [28]. Knowledge is created through a continuous conversion between tacit and explicit knowledge, which can occur in four different modes [27]: socialization (conversion of tacit knowledge into tacit knowledge), externalization (conversion of tacit knowledge into explicit knowledge), combination (conversion of explicit knowledge into explicit knowledge), internalization (conversion of explicit knowledge into tacit knowledge).

Externalization and socialization support knowledge sharing. The former focuses on explicit knowledge, aiming to systematize and store knowledge. The latter, in contrast, focuses on tacit knowledge and involves creating a repository of information on knowledge sources (e.g., Yellow Pages) to facilitate knowledge sharing within the organization [14]. As the name suggests, socialization relies on personal interaction. It requires identifying who has specific knowledge and promoting knowledge sharing between the person who possesses the desired knowledge (the expert) and the person interested in that particular knowledge (the seeker).

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. Expert-finding systems (EFS) are information retrieval systems that identify candidate experts and rank them based on their expertise in a given subject [17]. 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 [2]. These connections can be used to build representations of expertise areas and expert candidates and, thus, support finding experts.

Ontologies. An ontology is a formal and explicit specification of a shared conceptualization [36]. Every knowledge base, knowledge-based system, or knowledge level agent is committed, either explicitly or implicitly, to one conceptualization [36]. Ontologies can be used to assign semantics to information items, supporting data and knowledge management based on a common and shared understanding and representation [37].

An important distinction differentiates ontologies as conceptual models, called *reference ontologies*, from ontologies as computational artifacts, called *operational ontologies* [12, 13]. A reference ontology is constructed to make the best possible description of the domain in reality, 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. Both reference and operational ontologies have been used to aid software development. The former is suitable for supporting the description of the application domain itself and is applied in development time, a.k.a., *ontology-driven development* (ODD). The latter is appropriate for use as primary artifacts in run-time and plays a major role in application logic, a.k.a., *ontology-based architecture* (OBA) [15].

In this work, we used *Core-O*, a domain reference ontology that deals with concepts related to human competence such as knowledge, skills, and attitudes, among others [6]. In our approach, the ontology is used in development time (i.e., ontology-driven development). Core-O is composed of two sub-ontologies: the *Personal Competence Ontology*, which focuses on individuals and their competencies, addressing concepts such as *Human Task*, *Artifact*, and *Skill*; and the *Competence Type Ontology*, which regards competencies that are not specific to an individual - i.e., it describes the types of task, artifact, knowledge, and skill, among others. Details about Core-O can be found in [6]. Figure 1 presents the simplified Core-O fragment relevant to this paper. Concepts referring to types are shown in purple, and concepts referring to individuals are in green.

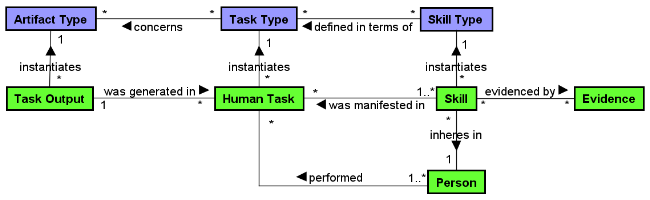


Figure 1: Core-O extract (simplified) relevant to this paper

A *Skill* is a quality that is inherently part of a *Person* and alludes to the capability to perform actions. In other words, it relates to the ability to apply knowledge to perform a *Human Task*, which can produce artifacts called *Task Outputs*. Thus, a *Human Task* is a manifestation of a *Skill* inherent in the *Person* who performed that task. For example, the execution of the task programming unit tests (*Human Task*) by John (*Person*) on October 22, 2024, at 11:30 am, producing the unit test 256 (*Task Output*), is a manifestation of John's skill in programming unit tests (*Skill*). *Skills* can be evidenced by some *Evidence*, such as certificates, qualifications, project participations, credentials, and other experiences. For example, John's Java coding skill is evidenced by his Java course certificate and his experiences coding Java. *Artifact Type*, *Task Type*, and *Skill Type* are concepts similar to those previously presented. However, they focus on types, i.e., refer to the types of task, artifact, and skill. For example, when referring to the task type Programming Unit Test, we mean the general task category rather than instances of execution of tasks of that type.

3 The *iKnow* Approach

iKnow aids in the development of systems (particularly in the phases that involve conceptual definition and modeling) to identify who knows what in software organizations. It is based on Core-O [6] conceptualization to identify who knows what based on performed tasks and produced task outputs (artifacts), which are interpreted as skill manifestations. 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.

Software organizations produce large amounts of data regarding tasks, 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 *Structural Model* is an Artifact Type related to the *Develop Class Diagram* Task Type, and *John* (Person) has performed several *develop class diagram* tasks (Human Tasks of the *Develop Class Diagram* Task Type) and produced several *UML class diagrams* (Task Output of the *Structural Model* Artifact Type), we can consider these tasks and artifacts as manifestations (Evidence) of the Skill Type *Developing Structural Conceptual Model* and, thus, conclude that *John* has the *developing structural conceptual* model Skill (i.e., a skill of the *Developing Structural Conceptual Model* type). Core-O enhances our understanding of these concepts and interconnections, contributing to identifying their relationships and enriching existing data.

An overview of *iKnow* is presented in Figure 2. It comprises eight steps. In a nutshell, during the first two steps (*i* and *ii*), the focus is on identifying the application domain and relevant skill types for the organization, and, thus, identifying 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) – here called non-technical factors – that influence the selection of an individual who can share knowledge. 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*) concerns defining the system architecture. Finally, the last step (*viii*) is related to the development of the system considering the results produced in the previous steps.

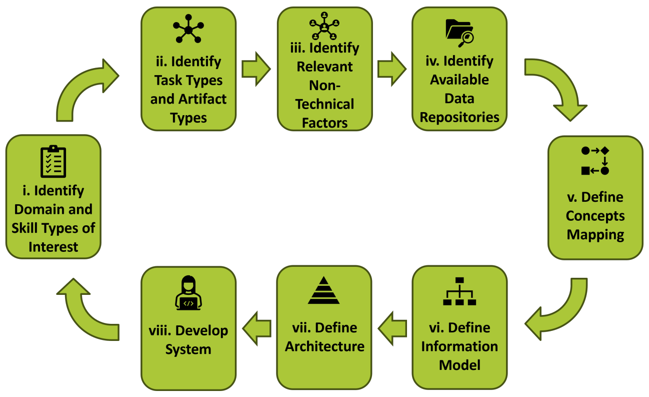

i. Identify Domain and Skill Types of Interest
ii. Identify Task Types and Artifact Types
iii. Identify Relevant Non-Technical Factors
iv. Identify Available Data Repositories
v. Define Concepts Mapping
vi. Define Information Model
vii. Define Architecture
viii. Develop System

Figure 2: *iKnow* overview

For simplification, *iKnow* steps are presented as sequential ones. However, 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 developed system. For this reason, the approach is defined as a cycle that allows for the evolution of the system through iterations. In the following, we briefly describe each step of *iKnow*.

(i) Identify domain and skill types of interest. This step aims to identify the domain to be addressed by the expert-finding system to be developed and the skill types to be considered. According to the organization's needs, the domain can be as general as *Software Engineering* or as narrow as *Java Software Development* or *Unit Testing*. Defining the domain is important because it establishes the possible range of skill types of interest and will later help identify suitable data sources. Once the domain is defined, the skill types to be considered must be identified.

Existing standards, such as the Occupation Information Network (O*Net[1]) and the European Skills, Competences, Qualifications and Occupations (ESCO[2]), 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. 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 playing 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 valuable for the organization may be identified by its employees.

When identifying the skill types, it is important to ensure that there is data related to them in the organization (e.g., in tools repositories and spreadsheets). Skill types without related data to support the identification of experts should not be considered at this point. Moreover, skill types should not be too general and difficult to relate to existing data (e.g., *Solving problems*).

As an example (we will use it here and in the following steps), suppose that the Project Management Office Manager of a large multinational software organization (*Org*) wants a system to help software project managers find experts to share knowledge to help them perform their tasks. In this case, the domain of interest could be *Project Management*, and examples of possible skill types of interest could be *Developing a Project Schedule* and *Developing 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 playing key roles in the organization. By identifying which task types relate 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. A skill type may be manifested through more than one task type. For example, the *Coding in Java* skill type could be manifested by the task types *Commit changes in Java files* or *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 an artifact of that 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 could be manifested through the task type *Write automated test scripts*. The occurrence of tasks of this type might not always be explicitly defined in task management tools. In these cases, data regarding artifacts of the artifact type (e.g., *test scripts*) recorded in version control systems could 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 a survey).

In the example presented in the previous step, consider that 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 *Create Project Schedule* and the artifact type *Project Schedule* could be manifestations of the skill type *Developing a Project Schedule*; and the task type *Create 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 and the decision on the most suitable expert to satisfy the seeker's needs. 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 deeper 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

[1]https://www.onetonline.org/
[2]https://esco.ec.europa.eu/

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.

In *Org* example, suppose that 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* would be availability, language, seniority, and years of service in *Org*.

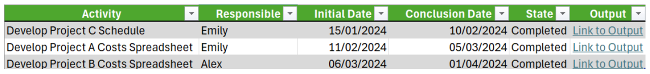
(iv) Identify available data repositories. This step aims at identifying the available data repositories in the organization that contain data related to the skill types of interest, and related task types and artifact types identified in the previous steps. Tools used in the organization are a good source of the desired data. Task management tools (e.g., JIRA[3]) 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[4]), cloud monitoring platforms (e.g., Datadog[5]), IT Service Management (ITSM) tools (e.g., ServiceNow[6]), and others. These tools often offer an API that can be used to access and extract data.

In *Org* example, considering the results of the previous steps, the system to be developed 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 tasks of the types *Develop Project Schedule* and *Develop Project Costs Spreadsheet* performed by the project managers, and the artifacts of the types *Project Schedule* and *Project Costs Spreadsheet* produced by them. Therefore, 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 cited in step (ii), MS Project and MS Excel could provide useful data.

(v) Define concepts mapping. This step aims to assign semantics to data considering the Core-O extract previously presented. 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 the previous step. These mappings ensure semantic alignment between the 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, communication among stakeholders, and the evaluation and evolution 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. Guidance to perform semantic mappings is provided in [7].

In *Org* case, suppose that data related to the execution of activities is recorded using a spreadsheet as illustrated in Figure 3. This spreadsheet includes columns for Activity, Responsible, and Output, capturing the performed activity, the responsible individual, and the resulting output. 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*. Suppose that in some projects, this data is recorded in MS Project. The semantic mapping between the MS Project conceptual model and Core-O concepts should also be performed. By doing so, it would be possible to identify which data from the different sources represent the same concept (e.g., **Activity** in the Excel Spreadsheet and **Task** in MS Project are equivalent to *Human Task* in Core-O) and should be integrated to support identifying who knows what.



Activity	Responsible	Initial Date	Conclusion Date	State	Output
Develop Project C Schedule	Emily	15/01/2024	10/02/2024	Completed	Link to Output
Develop Project A Costs Spreadsheet	Emily	11/02/2024	05/03/2024	Completed	Link to Output
Develop Project B Costs Spreadsheet	Alex	06/03/2024	01/04/2024	Completed	Link to Output

Figure 3: Fragment of a spreadsheet used in *Org*

(vi) Define the information model. Using the Core-O extract as a reference model, in this step, the conceptual model 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 the 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 [9]. 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. In [9] there are guidelines on how to develop the information model.

In *Org* case, consider that the Core-O extract conceptual model is suitable for the system to be developed and, as informed in step *(iii)*, it is necessary to record information about the selected non-technical factors. To do that, new attributes can be added to the Person class to record availability, language, seniority, and years of service of each person. Figure 4 illustrates a (hypothetical) information model that could be used to develop the system for *Org*.

[3]https://www.atlassian.com/software/jira
[4]https://github.com/
[5]https://www.datadoghq.com/
[6]https://www.servicenow.com/

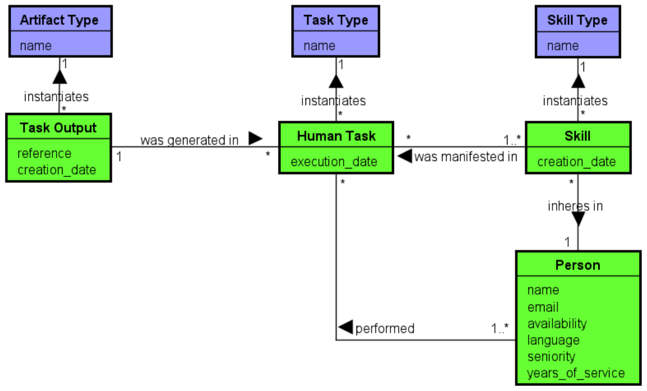


Figure 4: Hypothetical information model for the system

(vii) Define the architecture. In this step, the system architecture is defined. A solution for expert identification can be divided into two major contexts: one general, which focuses on providing a way to access the extracted data, and another dependent on the identified tools and concepts mappings. Figure 5 presents an overview of the proposed architecture that can be used as a basis to develop a system for identifying who knows what. It contains three modules: Core Competence Module, Data Loading Module, and Data Extraction and Transformation Module.

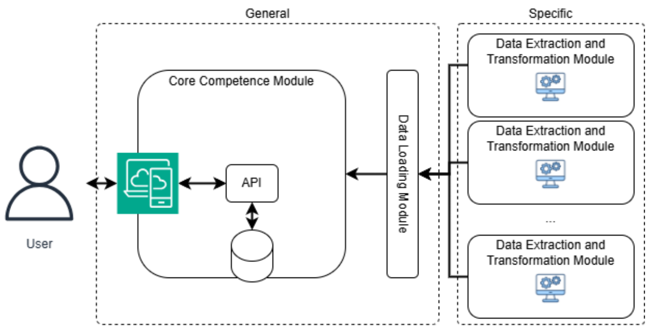

General
Specific
Core Competence Module
API
Data Loading Module
Data Extraction and Transformation Module
Data Extraction and Transformation Module
...
Data Extraction and Transformation Module
User

Figure 5: Proposed architecture

The *Core Competence Module* is responsible for storing information about who knows what and allowing users to access this 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 by third parties to ingest data from external sources into the platform. These modules have a general purpose, i.e., they handle concepts related to skills, tasks, etc., and do not refer to any specific domain such as Software Project Management or Software Requirements. The *Data Extraction Module*, in turn, is responsible for extracting, transforming, and loading data into the *Core Competence Module* through the *Data Ingestion Module*. Therefore, it addresses context particularities such as domain concepts, used tools, and available data.

(viii) Develop the system. The results of the previous steps are related to the system requirements specification and design. In this step, they can be refined and complemented to properly address other system requirements (e.g., desired features and usability, among others). Design, implementation, testing, and deployment activities are also performed. Therefore, this step encapsulates several software development activities. *iKnow* focuses on the conceptual aspects necessary to develop a system to identify who knows what considering skill manifestations as evidence of knowledge. It does not detail other steps of system development because they can follow well-known software development activities. Hence, in this step, the other activities needed to develop the system are performed according to the software development process defined by the software engineer who uses *iKnow*.

4 Applying *iKnow*

To demonstrate the feasibility of using *iKnow* and evaluate its usefulness, it was used to develop a system called *ExpertFY* (Expert For You), 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*.

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. 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 and topics. With many members working in different contexts, it is challenging for a knowledge seeker to identify the right person to get help. The first author works at BMC and has experienced this difficulty and observed other members face it. This scenario motivated the use of *iKnow* to develop *ExpertFY* for *BMC*.

The study involved two phases. In the first phase, the first author executed the seven first steps of *iKnow* with the participation of *BMC* employees and the second author. Then, considering the results produced in these steps, the second author executed the eighth step of *iKnow* and developed *ExpertFY*, a system to help identify who knows what in *BMC*. The successful use of *iKnow* to develop a system to help identify who knows what in an organization demonstrates that using *iKnow* is feasible. In the second phase, *ExpertFY* was used by some *BMC* employees, and a survey was performed to collect their perceptions of the system. The system's usefulness serves as an indirect indication of *iKnow*'s usefulness.

4.1 Developing *ExpertFY*

In this section, we provide an overview of *ExpertFY* development using *iKnow*. A detailed description of the results produced at each step of *iKnow* when developing *ExpertFY* as well as further information about *ExpertFY* can be found in our study package [5].

In the first step to developing *ExpertFY*, to identify the domain and skill types to be addressed by the system, the first author performed unstructured interviews with BMC tech managers. The domain of interest was *Software Development*, and examples of skill types of interest are *Optimizing SQL Queries* and *Monitoring applications*. In the second step, to identify task types and artifact types that could be manifestations of the identified skill types, the first author elaborated a set of assumptions. For example, "the skill type *Monitoring application* is manifested by the task type *Create or modify alerts* which produces the artifact type *Application alerts*". To evaluate the assumptions, the first author performed a survey with 11 BMC software developers. The assumptions were prioritized based on the developers' agreement level. The survey also asked the developers to identify the non-technical factors they judged more relevant to select experts to share knowledge in *BMC*. As a result, in the third step, the following non-technical factors were selected: availability, years of experience, and seniority.

In the fourth step, the following data repositories were selected to provide data related to the identified skill types, task types, and artifact types: Azure Boards[7], Azure Repos[8], and Datadog[9]. For each data repository, in the fifth step, we captured its conceptual model and defined the semantic mappings between the conceptual model and Core-O concepts. By doing that, data from Azure Boards, Azure Repos, and Datadog will be extracted and stored in *ExpertFY* database according to the established semantic mappings. For example, Figure 6 presents a fragment of the Datadog conceptual model. In the semantic mappings, we mapped Monitor to Task Output, User to Person, and the relation `created by` was used to (indirectly) represent Human Task, indicating that a user performed a task (e.g., create monitor) and created a monitor.

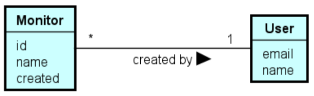


Figure 6: Fragment of Datadog conceptual model

In the sixth step, we made some adjustments to the Core-O conceptual model and transformed it into an information model, which is more suitable for implementation. This included simplifications and design decisions, such as constraining multiplicities, and embedding concepts into attributes. Moreover, given that a person's recommendation regarding another individual's skill can serve as evidence of that skill, we introduced the concept Recommendation as a specialization of Evidence. However, since Recommendation would be the only specialization of Evidence relevant to the system, we omitted the Evidence class in the information model for simplification. Figure 7 depicts the information model.

In the seventh step, we adopted the reference architecture proposed in *iKnow* but we needed to make some adjustments due to time constraints. The *Core Competence Module* was fully developed. To extract data from the repositories and store it in the *ExpertFY* database, we created some scripts. Finally, in the eighth step, the second author developed the system using the results produced in the previous steps and considering additional requirements about the features expected in the system. The main features (use cases) of *ExpertFY* are: *Search Who Possesses a Skill*, *Recommend a Person's Skill*, and *View Profile*. The repository containing its source code and instructions on how to run the application is available at https://bit.ly/4lH5SMz.

[7]https://azure.microsoft.com/en-us/products/devops/boards/
[8]https://azure.microsoft.com/en-us/products/devops/repos/
[9]https://www.datadoghq.com/

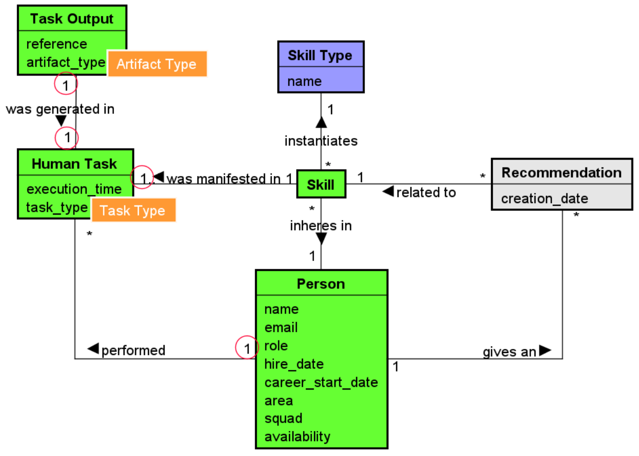


Figure 7: *ExpertFY* conceptual model

4.2 Evaluating *ExpertFY*

After developing *ExpertFY*, it was made available and five software developers of *BMC* used it. Thus, we performed a survey to collect their perceptions of the system.

Study Design. Our goal was to evaluate whether *ExpertFY* is useful to support the identification of who knows what in a software organization 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 indirect evidence that *iKnow* is useful, since it was used to develop *ExpertFY*).

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

The instruments consisted of four artifacts: (i) a document presenting the study context, the main functionalities of *ExpertFY*, and the instructions to participate in the study; (ii) a consent form to safeguard the participants' rights; (iii) a form to characterize the participants' profile and 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 system. The forms were prepared with the help of Google Forms and are available in our study package [5].

The adopted procedure consisted of inviting the participants and, for those who accepted the invitation, sending the documents cited above. The participants were chosen considering convenience criteria. The first author made a brief presentation and made himself available in case of doubts. After the presentation, the participants should use *ExpertFY* following the instructions described in document (i). The document presented the description of three scenarios, common in *BMC*, in which the participants should identify up to five individuals who could provide knowledge to help them perform a task. First (first stage), the participants should identify the experts without using *ExpertFY* (i.e., based on their knowledge of who knows what in BMC). After that (second stage), they should use *ExpertFY* to find experts for the same scenarios. Then, they should answer the questionnaire (iv).

The questionnaire included questions to collect the participants' perceptions of the adequacy of *ExpertFY* in supporting expert identification for knowledge-sharing purposes, the usefulness of the system, 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 comments and suggestions. The collected data is available in the study package.

Study Execution and Results. The participants were five software developers who work at *BMC*. Regarding their current roles, two participants declared that they worked as Fullstack Developers, two as Backend Developers, and one as a Salesforce Developer. In terms of academic background, four participants had an undergraduate degree and one had a post-graduate specialization. Participants reported different levels of experience in software projects: two declared to have more than 10 years of experience, one between 5 and 10 years, one between 3 and 5 years, and one between 1 and 3 years. Concerning the time they have worked in their current organization, one participant declared 5 to 10 years, one 3 to 5 years, two 1 to 3 years, and one less than 1 year.

All of them informed that considered very important knowledge sharing to support task execution in software development. Two of them declared that they usually 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. Following the planned procedure, after a presentation made by the researcher, the participants read the document with the instructions, followed them, and answered the questionnaire. Next, we summarize the results considering the main evaluated aspects:
- *ExpertFY's usefulness:* Four participants considered the system *very useful*, and one participant considered the system *useful*.
- *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 range of people they could potentially contact for help within the organization.
- *ExpertFY's ease of use:* Four participants considered the system *very easy* to use, and one considered it *easy* to use.
- *Feasibility of using ExpertFY:* Four participants considered that the system is*very feasible* to use in practice, and one considered it *feasible*.
- *Recommendation of ExpertFY to other people:* All five participants reported that they would recommend other people to use *ExpertFY*.
- *Stimulus to contact other people:* All participants answered that the system would stimulate them to contact other people to seek help.
- *Benefits and difficulties when using ExpertFY:* All five participants reported benefits from using *ExpertFY*. For example, one participants reported the deconstruction of knowledge silos especially in big companies enabled by the possibility offered by the system of finding any people in the organization who possesses the desired skill. Regarding the difficulties that could be faced when using *ExpertFY*, two participants mentioned that ranking knowledgeable people could lead to too many help requests for those at the top, which could affect their availability. Others pointed out that some task types may be poorly defined, making it difficult to identify the skills involved.
- *Improvement suggestions:* Only one participant identified improvement opportunities. According to them, the system could address different levels of proficiency in a skill.

Results Interpretation. Here, we discuss the results in terms of the indicators previously defined: usefulness and feasibility of use. We also present some comments made by the participants (here identified as P1 to P5).

Regarding the usefulness, participants considered the system very useful or useful. Participant P3 mentioned that, before using the system, they often relied on people considered technical references (e.g., solutions architects or tech leads) even if they were not sure these individuals had the knowledge they needed and if they were willing to share it. The system improved the identification of who knows what by amplifying the range of experts to be reached and considering real data to identify them.

Participant P4 highlighted that the system would be especially valuable for new hires. We think that this makes sense because people in an organization tend to learn who knows what considering their previous experiences and needs. Thus, the longer a person works in the organization, the higher their knowledge of who knows what (probably). However, different needs may arise, and even those in the organization for a long time may need to consult the system, which was also mentioned by the same participant. Aligned with this, participant P2, who had been in the organization for the longest time among the participants, was the only one who classified the system as useful (instead of very useful) because P2 believes that it can be more useful for newcomers.

Participant P5 reported that the system may improve problem-solving efficiency, because the system 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 adopted in the system more effective because they know in advance if someone has 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 outside the participants' social circle in the organization. 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 system is directly related to the size and complexity of the organization. In an organization with 10 employees, for example, it is easier to discover who knows what if compared to an organization with 1000 employees. Regarding the complexity, the standardization of tools, technologies, and programming languages adopted in the organization tends to facilitate knowledge dissemination in the organization, which could make the system less useful in such scenarios.

Four out of five participants considered the system very easy and very feasible to use. As mentioned by participant P3, the system leverages data existing in the organization. We believe this is crucial for its feasibility since the members do not need to perform extra activities to enable 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 system as "simple", "intuitive", and "straightforward". This suggests a low level of resistance to introducing the system in real-world scenarios. However, we must consider that although the evaluation was performed with real data, it is limited. We believe that the availability of skill types in the system is important to its feasibility. For example, if someone rarely finds the desired skill type in the system, they would be discouraged from using it.

As for improvements in *ExpertFY*, one participant suggested that the system could address different levels of skill proficiency. This limitation is caused by *iKnow*, which does not address such levels. Core-O addresses the Proficiency concept, and *iKnow* could be extended to address it as well. This concept is indeed important because someone who has the highest number of manifestations of a skill does not necessarily have a high level of proficiency. However, addressing this issue makes the development of an expert-finding system more complex. So, we believe that further studies are necessary to properly address this issue.

The results provided preliminary evidence that the system is useful and its use is feasible. They also suggest that *iKnow* is useful, since it supports the development of systems to identify who knows what as the one developed in this work. Furthermore, the results indicated that using task outputs and tasks as skill manifestations, as proposed in *iKnow*, was effective. Participants considered the suggestions made by the system more accurate than the ones they had identified on their own. This suggests that incorporating this rationale into expert-finding systems can contribute to knowledge-sharing within the organization.

4.3 Threats to Validity

Like 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.

An important limitation is that *iKnow* was used by its author. To minimize this limitation, BMC members and the second author, who did not participate in the creation of *iKnow* were involved. Even so, this use serves as proof of concept of the approach's feasibility, and it is necessary to perform other studies to ensure that the approach is suitable for use by other people in practical settings.

Another limitation to be considered is the fact that the first author works at *BMC*. On one hand, his knowledge of 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 opinions and were advised to be very honest because they would not be evaluated and their feedback would be important to improve the system.

The participants may also not have understood the system's features or the motivation for using it. To mitigate this threat, we made a brief presentation about the system and provided a document describing its 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 addressed 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 may not reflect a complete real-world scenario, even though we used real data. Moreover, the participants used the system for a short period of time, and the evaluation was based only 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 should 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. Therefore, these results should be complemented by further studies.

5 Related Work

Before developing *iKnow*, we carried out a systematic mapping study investigating approaches that help identify experts who can share knowledge in software development [4]. The mapping study results showed us that most approaches have relied on code repositories as a source of evidence for identifying experts and, consequently, focus on supporting developers in the codification activity. Therefore, there is a need to explore different elements as sources of evidence of expertise and increase activities supported by expert identification results. *iKnow* does that by enabling the use of different artifacts and tasks as skill manifestations that provide evidence of expertise. In addition, *iKnow* is application-domain and data source independent. When using *iKnow*, the software engineer defines the domains of interest and the data sources to be used.

The mapping results also revealed that there has been a lack of concern with non-technical factors that influence reaching the most suitable expert for a specific situation, and semantics has not been a concern. *iKnow* includes a step devoted to identifying non-technical factors that can influence expert identification in an organization and provides a (not exhaustive) set of non-technical factors that can be considered. To address semantics, *iKnow* relies on Core-O conceptualization to use skill manifestations as knowledge evidence and support expert identification. Moreover, the conceptualization is used to assign semantics to data, enabling the semantic integration of data from different sources. In the following, we cite some works and compare them with *iKnow*.

Lucas et al. [19] present 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. The models combine information from interactions between developers and between developers and artifacts. Schettino et al. [31], in turn, propose a technique that uses 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 a developer is compared to others. In Teusner et al. [38], a framework is proposed to analyze developers' expertise on the system's components based on code complexity measures. By aggregating these measures, the framework identifies experts on the components. In Morales-Ramirez et al. [25], a process is proposed for exploiting online feedback and discussions to compute expertise indicators. This process uses natural language processing (NLP) and graph-based analysis. Hughes and Crowder [16] proposed an architecture containing modules that, given a search query, produce a ranked list of documents or individuals. This work uses an ontology describing the application domain, i.e., it does not address general concepts related to expertise, such as knowledge or skill.

Aligned with our previous discussion, when analyzing these proposals, we noticed that most of them focus on specific sources of evidence of expertise and often propose solutions to specific repositories. As a consequence, although the works present solutions for using existing data to identify experts, they do not guide how to build other solutions (e.g., exploring different data sources or domains of interest). They typically handle technical concerns and approaches for extracting data from specific sources. We believe that some of the identified approaches could be combined with *iKnow*, since it provides a high-level guide for identifying expertise evidence and developing expert-finding systems, but does not address low-level concerns, such as optimal methods for calculating expertise scores, which could be covered by other approaches.

Table 1, summarizes some aspects comparing *iKnow* and the cited works. The analyzed aspects were: (*A1*) the data sources used for expert finding, (*A2*) whether the approach is domain-specific or general, (*A3*) whether it provides guidelines for developing expert-finding systems, (*A4*) whether it considers non-technical factors, and (*A5*) whether it addresses semantic aspects.

	A1	**A2**	**A3**	**A4**	**A5**
[19]	Code repository and ticket system	General	No	No	No
[31]	Code repository	Specific	No	No	No
[38]	Code repository	Specific	No	No	No
[25]	Code repository	Specific	No	No	No
[16]	Electronic documents	General	No	No	Yes
iKnow	**Multiple sources**	**General**	**Yes**	**Yes**	**Yes**

Table 1: Aspects addressed by expert-finding system approaches

6 Final Considerations

Software development is knowledge-intensive, and although there is a large volume of information, people still rely on experts to access specific knowledge [17]. Expert-finding systems help by identifying and ranking experts [3, 17], but their development is complex due to abstract concepts and diverse data sources [17]. Ontologies can support this process by organizing knowledge through a shared vocabulary [36].

In this paper, we presented *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 relationships 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.

iKnow was used to develop *ExpertFY*, which served as a demonstration that using *iKnow* is feasible and useful. *ExpertFY* enabled software developers of the *BMC* organization to identify who knows what and it was considered useful and feasible. These results can be understood as initial evidence that *iKnow* is a promising approach. Although *ExpertFY* was developed considering the specific *BMC* context, we believe that only minor modifications would be necessary for use in other organizations. Furthermore, *ExpertFY* development can be used as an example to help other people develop similar systems using *iKnow*. The studies performed so far to evaluate *iKnow* involved the participation of the first author. Thus, new studies involving the *iKnow* use by other people are necessary to provide new evidence.

As future work, we believe that non-technical factors can be further explored. A more extensive list of possible factors can be defined based on the literature and feedback from 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 could be valuable to create a catalog of skill types in the context of Software Engineering and integrate it with Core-O conceptualization. Furthermore, we believe the relation between some skill types and task types (what we called assumptions) may be independent of a particular organization and could be reused in other contexts. Moreover, it is important to evaluate the accuracy of the assumptions regarding expert identification, and the approach could be evolved to address different proficiency levels.

Another important aspect to be explored is the influence of task output quality when evaluating an individual's skill manifested through a performed task and produced artifact. For example, a person can create a unit test containing test smells. In this case, has that person still manifested a skill of writing unit tests?

Finally, considering the increasing use of AI-based systems (particularly LLMs) as a source of knowledge, research should be conducted to investigate their 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.

ARTIFACT AVAILABILITY

A document describing *iKnow* and its use to develop *ExperFY*, and the package of the study performed to evaluate *ExpertFY* are available in [5].

ACKNOWLEDGMENTS

This research is supported by Coordination for the Improvement of Higher Education Personnel - Brazil (CAPES) - Finance Code 001, and Espírito Santo Research and Innovation Support Foundation (FAPES) - Processes 2023-5L1FC, 2021-GL60J, 2022-NGKM5, and T.O. 1022/2022.

REFERENCES

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

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

[3] Krisztian Balog, Yi Fang, Maarten de Rijke, Pavel Serdyukov, and Luo Si. 2012. Expertise Retrieval. *Foundations and Trends® in Information Retrieval* 6, 2–3 (2012), 127–256. doi:10.1561/1500000024

[4] Carlos Braga, Paulo Santos Jr, and Monalessa Barcellos. 2023. Help! I need somebody. A Mapping Study about Expert Identification in Software Development. In *Proceedings of the XXXVII Brazilian Symposium on Software Engineering* (Campo Grande, Brazil) *(SBES '23)*. Association for Computing Machinery, New York, NY, USA, 154–163. doi:10.1145/3613372.3613389

[5] Carlos Eduardo Correa Braga, Abraão Jesus dos Santos, and Monalessa P. Barcellos. 2025. *Supplementary material of the study "An Ontology-based Approach for Developing Systems to Identify Who Knows What in Software Organizations"*. doi:10.5281/zenodo.15768954

[6] Rodrigo F. Calhau, João Paulo A. Almeida, Tiago Prince Sales, Pedro Paulo F. Barcelos, and Giancarlo Guizzardi. 2024. *Core-O: A Competence Reference Ontology for Professional and Learning Ecosystems.* 16–30. doi:10.3233/FAIA241289

[7] Rodrigo Fernandes Calhau and Ricardo de Almeida Falbo. 2010. An Ontology-Based Approach for Semantic Integration. In *2010 14th IEEE International Enterprise Distributed Object Computing Conference.* 111–120. doi:10.1109/EDOC.2010.32

[8] Christopher S. Campbell, Paul P. Maglio, Alex Cozzi, and Byron Dom. 2003. Expertise Identification Using Email Communications. In *Proc. of the Twelfth Int. Conf. on Information and Knowledge Management* (New Orleans, LA, USA) *(CIKM '03)*. Association for Computing Machinery, New York, NY, USA, 528–531. doi:10.1145/956863.956965

[9] R. Carraretto and J.P.A. Almeida. 2012. Separating Ontological and Informational Concerns: Towards a Two-Level Model-Driven Approach. In *2012 IEEE 16th International Enterprise Distributed Object Computing Conference Workshops.* IEEE, 29–37. doi:10.1109/EDOCW.2012.14

[10] Kimiz Dalkir. 2013. *Knowledge Management in Theory and Practice.* doi:10.4324/9780080547367

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

[12] Nicola Guarino, Daniel Oberle, and Steffen Staab. 2009. *What Is an Ontology?* Springer Berlin Heidelberg, 1–17. doi:10.1007/978-3-540-92673-3_0

[13] Giancarlo Guizzardi. 2007. 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.* IOS Press, NLD, 18–39.

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

[15] Hans-Jörg Happel and Stefan Seedorf. 2006. Applications of ontologies in software engineering. In *Proc. of Workshop on Sematic Web Enabled Software Engineering"(SWESE) on the ISWC.* Citeseer, 5–9.

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

[17] Omayma Husain, Naomie Salim, Rose Alinda Alias, Samah Abdelsalam, and Alzubair Hassan. 2019. Expert Finding Systems: A Systematic Review. *Applied Sciences* 9, 20 (2019). doi:10.3390/app9204250

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

[19] Edson M. Lucas, Toacy C. Oliveira, Daniel Schneider, and Paulo S. C. Alencar. 2020. Knowledge-Oriented Models Based on Developer-Artifact and Developer-Developer Interactions. *IEEE Access* 8 (2020), 218702–218719.

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

[21] David W. McDonald and Mark S. Ackerman. 1998. Just Talk to Me: A Field Study of Expertise Location. In *Proc. of the 1998 ACM Conf. on Computer Supported Cooperative Work* (Seattle, Washington, USA) *(CSCW '98)*. Association for Computing Machinery, 315–324. doi:10.1145/289444.289506

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

[23] Roziah Mohd Rasdi and Gangeswari Tangaraja. 2022. Knowledge-sharing behaviour in public service organisations: determinants and the roles of affective commitment and normative commitment. *European Journal of Training and Development* 46, 3/4 (May 2022), 337–355. doi:10.1108/EJTD-02-2020-0028

[24] Alan Moraes, Eduardo Silva, Cleyton da Trindade, Yuri Barbosa, and Silvio Meira. 2010. Recommending experts using communication history. In *Proc. of the 2nd Int. Workshop on Recommendation Systems for Software Engineering.* ACM, Cape Town South Africa, 41–45. doi:10.1145/1808920.1808929

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

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

[27] Ikujiro Nonaka. 1994. A Dynamic Theory of Organizational Knowledge Creation. *Organization Science* 5 (1994), 14–37. https://api.semanticscholar.org/CorpusID:17219859

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

[29] D.E. O'Leary. 1998. Enterprise knowledge management. *Computer* 31, 3 (March 1998), 54–61. doi:10.1109/2.660190

[30] I. Rus and M. Lindvall. 2002. Knowledge management in software engineering. *IEEE Software* 19, 3 (May 2002), 26–38. doi:10.1109/MS.2002.1003450

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

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

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

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

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

[36] Rudi Studer, V.Richard Benjamins, and Dieter Fensel. 1998. Knowledge engineering: Principles and methods. *Data Knowledge Engineering* 25, 1 (1998), 161–197. doi:10.1016/S0169-023X(97)00056-6

[37] Alhusain Taher, Faridaddin Vahdatikhaki, and Amin Hammad. 2022. Formalizing knowledge representation in earthwork operations through development of domain ontology. *Engineering, Construction and Architectural Management* 29, 6 (24 June 2022), 2382–2414. doi:10.1108/ECAM-10-2020-0810 Publisher Copyright: © 2021, Emerald Publishing Limited..

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

[39] Wasko and Faraj. 2005. Why Should I Share? Examining Social Capital and Knowledge Contribution in Electronic Networks of Practice. *MIS Quarterly* 29, 1 (2005), 35. doi:10.2307/25148667

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