Supporting the recruitment of software development experts: aligning technical knowledge to an industry domain

Vitor de Campos[1], **José Maria N. David**[1], **Victor Ströele**[1], **Regina Braga**[1]

[1]Computer Science Postgraduate Program – Federal University of Juiz de Fora
Juiz de Fora – MG – Brazil

vitor.queiroz@ice.ufjf.br, jose.david@ufjf.br

victor.stroele@ufjf.br, regina.braga@ufjf.br

Abstract. *Finding experts that meet specific technical skills, combined with expertise in an industry domain, is essential in software development environments. However, this may be a complex task once different information about software developers is scattered among diverse databases. This work aims to detect experts and assemble a list of recommended experts regarding technologies and industry domains of interest. Data from LinkedIn, GitHub, and Topcoder platforms were used to achieve this goal. Our approach matches data using semantic and syntactic techniques and infers non-obvious information through an ontology. The information regarding the recommended software developers has the potential to support decision-makers and recruiters.*

1. Introduction

The speed of change has become one of the main differences between the fourth and the previous industrial revolutions [Frey and Osborne 2015] when a large volume of new information available follows the fast-paced technological evolution. Whereas the previous industrial revolutions have developed at a linear pace, on the other hand, it is said that the fourth industrial revolution has developed at an exponential rate, as such, it will affect all countries and industries at the same time [Schwab 2016]. As part of a crucial industry evolution, software development companies have found detecting specialists in specific software development domains difficult. Due to the high demand for a specialist workforce and the scarcity of such professionals, companies have hired professionals at the initial levels of their careers since it has not been possible to find specialists to fulfill such roles [Hyrynsalmi et al. 2021].

Finding collaborators with the desired technical knowledge, such as specific programming languages or software development standards, aligned with the experience of working in specific industry domains is of high interest to companies and institutions that carry out software development in collaborative environments. As an example, suppose that a company from the food and beverages industry is looking for software developers to compose its staff. This company is looking for a collaborator with technical knowledge in the JavaScript stack, who has already worked with agile project development, and who already has experience in the food industry. Thus, this new employee would not only have the necessary technical knowledge to work in the open position but would also have knowledge about the business rules and classic problems faced by companies in this industry domain. So, the combination of these skills makes this professional valuable for the job.

In order to support the search for collaborators, Recommendation Systems (RS) can be used to provide suggestions for feasible candidates based on data analysis. An RS is a software system that provides suggestions on items of interest from the analysis of data [Ricci et al. 2022], using different approaches such as algorithms for analyzing network metrics [Knoke and Yang 2008], machine-learning [Al-Taie et al. 2018] and semantic approaches [Hoehndorf et al. 2016]. One of the advantages of using semantic techniques in this scenario is the possibility of inferring non-obvious data from the syntax and semantics of data, extracted from databases [Guarino et al. 2009]. As a semantic approach, an ontology can be described as an explicit representation of the relationships and concepts of knowledge domains. Ontologies can be used to discover new information from semantic models and inference rules. [Guarino et al. 2009, Herre 2010]

This work proposes an RS approach that could support the search for suitable software developers for high-specialty contexts aligning technical knowledge in software development with knowledge in specific industry domains. By extracting data from LinkedIn[1], TopCoder[2] and GitHub[3] databases, data is converted into a canonical format, saved into a Global Schema Database and, finally, non-obvious inferences are made from the extracted information through a proposed ontology considering semantic and syntactic analysis. In order to guarantee the quality and reliability of the data, the proposed approach considers data provenance [Buneman et al. 2001]. By doing so, the different data extracted from diverse databases are better identified and categorized.

As a result, collaborators in the context of global software development are presented. The final recommendation considers factors that align technical skills (hard skills and soft skills) to the specific domain of interest in the industry. As the main contribution, this work presents the first stages of an approach that considers: i) the alignment between industry domains and desired technologies, ii) the extraction and integration of three different relevant bases, and iii) an ontology that considers collaboration aspects of the specialists found.

Thus, the remainder of this article is structured as follows: section 2 discusses related work, section 3 presents the RS approach, section 4 presents the results, and section 5 addresses final considerations and related work.

2. Related Work

Recommendation systems have been used in several scenarios in software development. Whether assisting in accessing available information on software projects [Beecham et al. 2012], for guidance on specifications in agile software development [Ghaisas 2010], in the allocation of development teams [Pereira et al. 2010] or even finding qualified developers to open source projects [Zhang et al. 2017]. Regarding recommendation systems of collaborators, studies have explored topic-oriented [Lin et al. 2017] and expertise-oriented [Alarfaj et al. 2012] approaches. Other studies have explored graph metrics algorithms, machine-learning [Al-Taie et al. 2018], and even semantic approaches [Cifariello et al. 2019].

[1]link 1 - https://www.linkedin.com
[2]link 2 - https://www.topcoder.com
[3]link 3 - https://github.com

Regarding semantics, the work by [Cifariello et al. 2019] proposes a search engine combining language modeling techniques for expert finding in academia. The study results demonstrate the effectiveness of the semantic approach, being able to compute the expertise of authors. The study, however, is limited to the context of the scientific academy and the detection of authors, not presenting a solution to collaboration software development environments. In [Martínez-García et al. 2020], the authors propose the description of an ontology development process following the Methontology Framework in order to reduce architectural knowledge loss and support expertise location. The work, however, does not consider the use of ontologies for collaborators discovery and recommendation. In order to propose a software expert recommendation system, [Pourheidari et al. 2018] develop a system exploring two social networks: LinkedIn and Twitter. The work aims to develop a system that considers the relationships between the data of the two providers and results in recommendations not being possible within just one site. However, the authors do not consider using an ontological approach in their solution in order to recommend software developers.

For the three of the discussed studies, [Martínez-García et al. 2020], [Pourheidari et al. 2018] and [Neira et al. 2018], the advantage of using a semantic approach, such as ontologies, would represent a different correlation of information centered on inferring non-obvious correlations, which a regular database technique could not take advantage of. A data extraction method combining inferences from ontologies using data from GitHub repositories has been proposed [Lopes et al. 2021]. The work analyses both semantic and syntactic aspects of the data in order to extract topics from key terms. Nonetheless, the work does not consider the alignment of target technologies to industry domains from different data providers.

Therefore, the relevance of this area of research is clear. However, it is notable the lack of studies that use semantic approaches allied to the extraction of multiple bases to generate knowledge to recommend collaborators in the context of global software development by analyzing factors that align technical competencies (hard skills and soft skills) to the specific domain of interest in the industry.

3. Approach Proposal

As a way to gather data, Figure 1 presents an overview of the approach phases. The extraction process begins with API connections with the Local Conceptual Schemas (LCSs) [Özsu and Valduriez 1999]. The extraction service then saves the raw data to a data lake [Miloslavskaya and Tolstoy 2016]. In a data lake, saving the data returned from the extraction in an unchanged form is prior in order to guarantee the usage of the information in the following steps. The model of integration with the databases is a logical model. In this model, the information that will be extracted from the databases is pre-established and, at each query, only the relevant information established in the project is extracted.

Three main databases are listed to obtain data from developers that will be processed through the proposed solution: i) LinkedIn, ii) GitHub, iii) Topcoder. The first, LinkedIn, was chosen due to the information from professionals it brings to the proposed solution regarding the context of their professional positions, work experience, industry sectors, time of experience, and education. The second, GitHub, aggregates information specific to the software development context, being able to provide information such as

programming languages used, projects carried out, companies or institutions, and time of experience. The third, Topcoder, is a crowdsourcing platform that has been used to solve challenges, in which developers are rewarded for the completion of tasks, focusing on collaboration. From this database, it is possible to extract information such as the challenges involved by the users, the programming languages used, and general information about developers, such as work experience or education.

With data saved in the data lake, the translations take place. The translation process aims to convert the data coming in specific formats of each base to the canonized format of the solution. At the end of the translation process, the canonical representation of the data will be ready. Figure 1 demonstrates the described extraction layer. In Figure 1, the canonical forms of the data are represented by the acronym InS (Intermediary Schema).

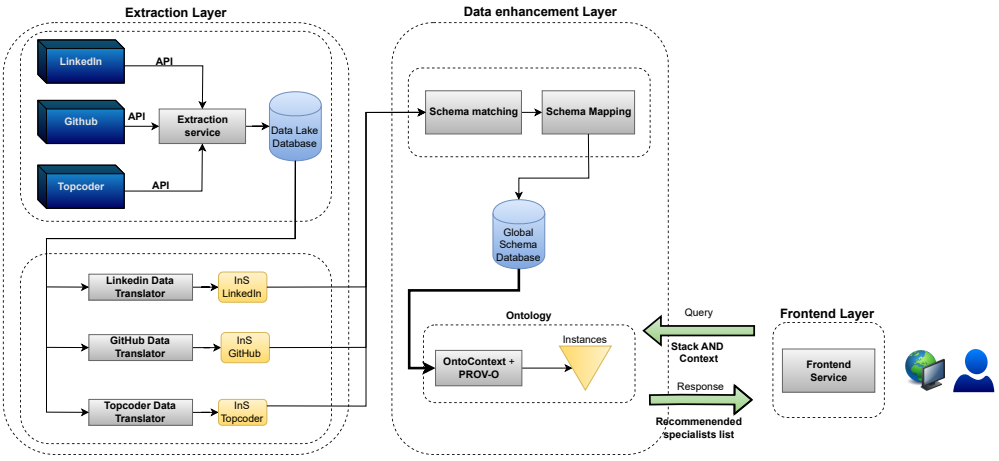

Figure 1. Overview of the proposed approach

With the data from each database in its canonical representation at the end of the extraction layer, the data flows into the enhancement layer. It is in this layer that the data will be aggregated, and non-obvious inferences will be made from the ontology.

The first step of this layer is the mapping. In this step, Schema Matching and Schema Mapping are performed. In Schema Matching, the syntactic and semantic correspondences between the elements will be determined. In the Schema Mapping, the way in which each element of the LCSs will be mapped to the Global Conceptual Schema (GCS) [Özsu and Valduriez 1999] is determined. Future studies will further detail the matching and mapping processes.

The database with the GCS will have its model defined in advance, so the results of queries prompted by the system user will be restricted to the set of objects defined in the global model. This is a Global As View (GAV) model [Özsu and Valduriez 1999].

The ontology step is the last step in the data enhancement layer. Non-obvious inferences are made from the data contained in the canonized base, which is an important intelligence step in the process. We propose a novel ontology[4] considering the GSC schema and the recommendations on provenance standards [Buneman et al. 2001].

[4]link 4 - https://www.github.com/vitorqcq/devFinderOntology

In order to represent the personal attributes of developers, hereby called soft skills, the list of such skills compiled by [Matturro et al. 2019] and described in their systematic mapping is used in this work. When querying the databases, our approach searches for such soft skills and assigns them to the related developers. The considered soft skills in this project are: Communication skills, Conflict Management, Customer orientation, Teamwork, Analytical skills Organizational and Planning skills, Interpersonal skills, Problem-solving skills, Autonomy, Decision-making, Initiative, Change management, Commitment/Responsibility, Ethics, Results orientation, Innovation, Critical thinking, Listening skills, Fast learner, Methodical.

As a way of guaranteeing the traceability of the developers' information generated from the proposed ontology and thus guaranteeing quality and reliability factors for the final results, the proposed model uses the PROV-O provenance model[4]. Three types of data stand out in the model: i) agents, ii) entities, iii) activities. The entities of the model are: Developer, Company, and Industry. The entities are: Hardskill, Softskill, Location and Role. The activities: LinkedIn-Experience, GithubProject and Topcoder-Challenge, as shown in Figure 2.

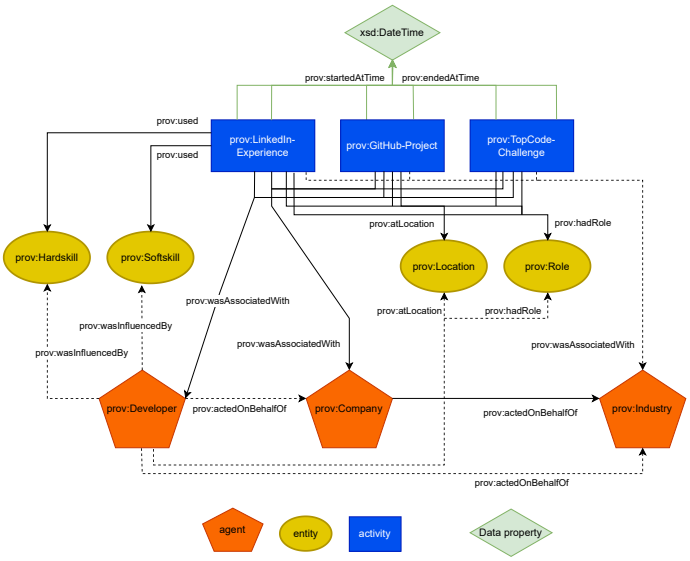

Figure 2. Proposed ontology representation

The relationships between entities are also proposed by the PROV-O model and link the information extracted in the bases to their correspondences in the ontology. The purpose of using an ontology lies in providing intelligence to the system by inferring non-obvious information from the data it is possessed. For that, we used Semantic Web Rule Language (SWRL). These are the rules that bring intelligence and inferences to this solution that traditional databases may not be able to express. In Figure 3 the relationships inferred by the ontology are represented by the dotted lines. The defined SWRL rules were:

- **Rule 1:** $Developer(?d)$ ^$influenced(?d, ?w)$ ^$Company(?c)$ ^$influenced(?c, ?w)$ →actedOnBehalfOf(?d, ?c)

[4]link 4 - https://www.w3.org/TR/prov-o/

- **Rule 2:** $Company(?c)$ ^$actedOnBehalfOf(?c, ?i)$ ^$Industry(?i)$ ^$influenced(?i, ?c)$ ^$Developer(?d)$ ^$actedOnBehalfOf(?d, ?c)$ →actedOnBehalfOf(?d, ?i)
- **Rule 3:** $Work(?w)$ ^$used(?w, ?s)$ ^$Skill(?s)$ ^$influenced(?s, ?w)$ ^$Developer(?d)$ ^ $influenced(?d, ?w)$ →wasInfluencedBy(?d, ?s)
- **Rule 4:** $Location(?p)$ ^$atLocation(?w, ?p)$ ^$Work(?w)$ ^ $wasAssociatedWith(?w, ?d)$ ^$Developer(?d)$ →atLocation(?d, ?p)
- **Rule 5:** $Role(?r)$ ^$hadRole(?w, ?r)$ ^$Work(?w)$ ^$wasAssociatedWith(?w, ?d)$ ^ $Developer(?d)$ →hadRole(?d, ?r)
- **Rule 6:** $Work(?w)$ ^$wasAssociatedWith(?w, ?c)$ ^$Company(?c)$ ^ $actedOnBehalfOf(?c, ?i)$ ^$Industry(?i)$ →wasAssociatedWith(?w, ?i)

Rule 1 aims to relate the developer who is linked to a work activity with the company related to this work. Rule 2 relates the developer to the industry to which their work experiences are linked. Rule 3 expresses that if the developer used a skill (skill) in a work experience, then the developer has that skill. Rule 4 relates the location of a work experience to the developer who is connected to it. Rule 5 relates the role title that a developer has had in some work experience to the developer itself. Rule 6 associates industries with an activity entity.

Figure 3 demonstrates the classes view (on the left) and the OntoGraf tab (on the right) of the Protegè system. Through the figure, it is possible to perceive the organization of the proposed classes as activities, agents, and entities in addition to the Location and Role classes, inherited from the PROV-O provenance model.

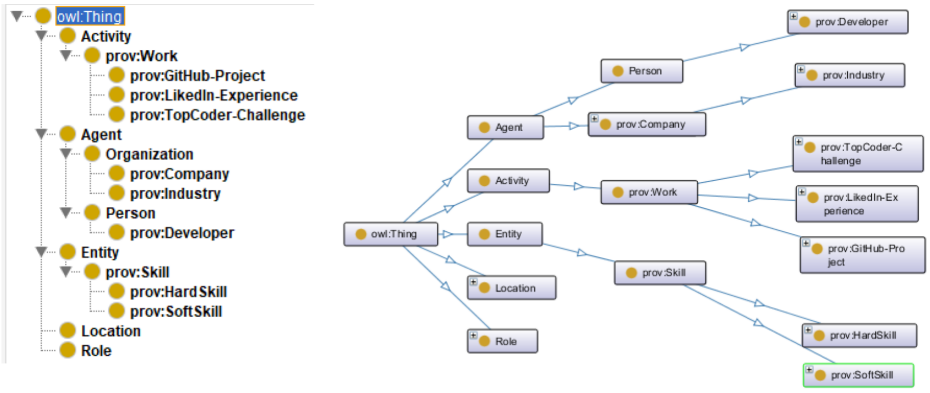

Figure 3. Representation of the proposed ontology in the Protègè system

4. Results

For the purposes of a better understanding, we will return to the example mentioned in the Introduction section. Consider a company from the food and beverages industry that is looking for software developers to compose its staff. In its search for software developers using the proposed solution, the company enters the following data:

```
{
    "technologies" : ["JavaScript", "HTML", "CSS", "React", "Node"],
    "industries": ["Food", "Food and Beverage"]
}
```

Prompting the data of the desired developers to be recommended, the system will search the three databases. Then, following the described flow, the system will go through the translation, mapping, and, finally, the ontological stage. In this step, suppose that three potential developers have been found: Elisa, Robert, and Nick.

Now it is possible to explore one of the returned developers so that their work experiences will be demonstrated. Due to space restrictions, only the information contained in the developer Robert will be discussed. However, similar information could be found in the other two developers.

By exploring the links of Robert it can be seen that the system identified a challenge in the TopCoder database, a repository on Github, and a job description on LinkedIn involving the search terms. All these work experiences relate to the same developer (Robert) and the same company, Food4u. This information can be easily understood by the name of each experiment, which follows the logic *"prov: Developer name - Company name - Database"*, as seen in Figure 4.

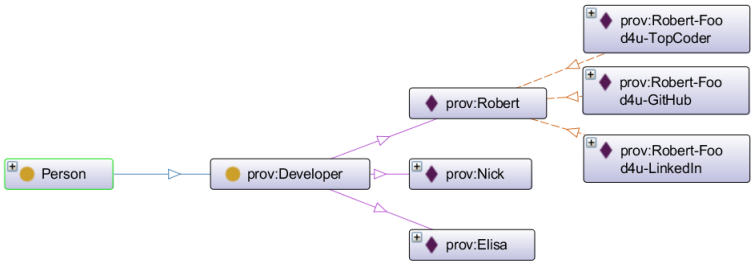

Figure 4. The work experiences of Robert

It is now desired to expand each of the experiences of Robert. Thus, it is possible to visualize which technical knowledge, and soft skills, in which companies the developer worked and which industries Robert was involved in. Figure 5 shows the clustered experiences returned from each database used in the solution.

It is possible to see that in each database the developer used different technologies, such as Android in the Topcoder base, which is not present in the other experiences. However, some of the technical competencies were used in different experiences, as in the case of JavaScript, having been used both in the TopCoder experience and in the GitHub experience.

Additionally, note that the developer's soft skill competencies are tied to the LinkedIn experience. The LinkedIn and GitHub experiences are associated with the *New-York-USA* location. Also, it is possible to find the roles held in these experiences, such as the *prov:Project_Owner* linked to the LinkedIn experience. It is also possible to see that all 3 experiences are linked to the same company: *Food4you*.

Nevertheless, from the point of view of the *Robert* instance in the resulting owl file (W3C Web Ontology Language), it is necessary to highlight the information inferred by the ontology and compose the final representation of the developer. There is a unified construction of all skills (hard skills and soft skills), workplaces, positions, and the industry that this developer is related to.

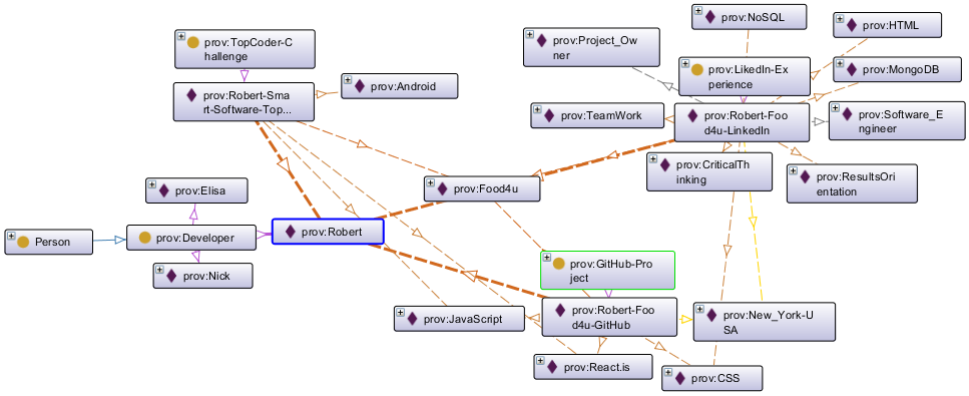

Figure 5. Expanded Experiences of Robert

Thus, the company of the example holds valuable information about the developers recommended by the system. With data on the technical skills and industry of the professionals, the company has greater autonomy to integrate new software developers to collaborate with its team.

5. Conclusions and future work

In a scenario where collaborators in software development are increasingly needed, finding such professionals who meet the specific needs of companies from different domains of the industry can be a challenge. In this sense, we propose a recommendation system for collaborators in software development that considers technical skills and areas of activity in the industry in order to support the search and selection of technology professionals. To this end, three context-relevant databases are analyzed in a process of extraction, treatment, and inference of non-obvious data from an ontology enhanced by data provenance.

As a contribution, this research presents the first results of a broader project that proposes an approach meant to be capable of extracting and processing information from data obtained from different databases to infer new information as means of supporting decision-makers in their struggle to find suitable software developers specialists to collaborate on projects. Based on the recommendations of this system, companies and institutions are supported in a task that may be complex and increasingly necessary.

The limitations in the current version of the proposed system lie in the correlation between the software developers returned from the different databases. Beyond that, APIs for communicating with databases are limited to a maximum number of requests, which prevents the system from working faster when extracting large volumes of data. In these cases of a large volume of data, the processing time of the mappings and the ontology also increases considerably.

In future work, we consider further analyses of the mapping and matching between the software developers found in the different databases, the integration between the layers not yet integrated, a better understanding of the temporal effects, the comparison of the proposed approach to other published related studies, and the construction of a layer that turns the solution available to interested end users.

Acknowledgement

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001, CNPq/Brazil (grant: 307194/2022-1), and FAPEMIG/Brazil (grant: APQ-02685-17), (grant: APQ-02194-18).

References

Al-Taie, M. Z., Kadry, S., and Obasa, A. I. (2018). Understanding expert finding systems: domains and techniques. *Social Network Analysis and Mining*, 8(1):57.

Alarfaj, F., Kruschwitz, U., Hunter, D., and Fox, C. (2012). Finding the right supervisor: Expert-finding in a university domain. pages 1–6.

Beecham, S., Carroll, N., and Noll, J. (2012). A decision support system for global team management: Expert evaluation remidi.

Buneman, P., Khanna, S., and Wang-Chiew, T. (2001). Why and where: A characterization of data provenance. In *International conference on database theory*, pages 316–330. Springer.

Cifariello, P., Ferragina, P., and Ponza, M. (2019). Wiser: A semantic approach for expert finding in academia based on entity linking. *Information Systems*, 82:1–16.

Frey, C. and Osborne, M. (2015). *Technology at Work: The Future of Innovation and Employment*.

Ghaisas, S. (U.S. Patent 9262126B2, 2010). Recommendation system for agile software development.

Guarino, N., Oberle, D., and Staab, S. (2009). *What Is an Ontology?*, pages 1–17. Springer Berlin Heidelberg, Berlin, Heidelberg.

Herre, H. (2010). *General Formal Ontology (GFO): A Foundational Ontology for Conceptual Modelling*, pages 297–345. Springer Netherlands, Dordrecht.

Hoehndorf, R., Gkoutos, G. V., and Schofield, P. N. (2016). Datamining with ontologies. *Methods Mol Biol*, 1415:385–397.

Hyrynsalmi, S. M., Rantanen, M. M., and Hyrynsalmi, S. (2021). The war for talent in software business - how are finnish software companies perceiving and coping with the labor shortage? In *2021 IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC)*, pages 1–10.

Knoke, D. and Yang, S. (2008). *Social Network Analysis*, volume 154.

Lin, S., Hong, W., Wang, D., and Li, T. (2017). A survey on expert finding techniques. *Journal of Intelligent Information Systems*, 49(2):255–279.

Lopes, T., Ströele, V., Braga, R., David, J. M. N., and Bauer, M. (2021). Identifying and recommending experts using a syntactic-semantic analysis approach. In *2021 IEEE 24th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pages 739–744.

Martínez-García, J. R., Castillo-Barrera, F.-E., Palacio, R. R., Borrego, G., and Cuevas-Tello, J. C. (2020). Ontology for knowledge condensation to support expertise location in the code phase during software development process. *IET Software*, 14(3):234–241.

Matturro, G., Raschetti, F., and Fontán, C. (2019). A systematic mapping study on soft skills in software engineering. *Journal of Universal Computer Science*, 25:16–41.

Miloslavskaya, N. and Tolstoy, A. (2016). Big data, fast data and data lake concepts. *Procedia Computer Science*, 88:300–305.

Neira, A., Steinmacher, I., and Wiese, I. (2018). Characterizing the hyperspecialists in the context of crowdsourcing software development. *Journal of the Brazilian Computer Society*, 24:17.

Özsu, M. T. and Valduriez, P. (1999). *Principles of distributed database systems*, volume 2. Springer.

Pereira, T. A. B., dos Santos, V. S., Ribeiro, B. L., and Elias, G. (2010). A recommendation framework for allocating global software teams in software product line projects. In *Proceedings of the 2nd International Workshop on Recommendation Systems for Software Engineering*, RSSE '10, page 36–40, New York, NY, USA. Association for Computing Machinery.

Pourheidari, V., Mollashahi, E. S., Vassileva, J., and Deters, R. (2018). Recommender system based on extracted data from different social media. a study of twitter and linkedin. In *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pages 215–222.

Ricci, F., Rokach, L., and Shapira, B. (2022). *Recommender Systems: Techniques, Applications, and Challenges*, pages 1–35. Springer US, New York, NY.

Schwab, K. (2016). The fourth industrial revolution.

Zhang, X., Wang, T., Yin, G., Yang, C., Yu, Y., and Wang, H. (2017). Devrec: A developer recommendation system for open source repositories. In Botterweck, G. and Werner, C., editors, *Mastering Scale and Complexity in Software Reuse*, pages 3–11, Cham. Springer International Publishing.