UNIVERSIDADE FEDERAL DE JUIZ DE FORA
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Vitor Queiroz de Campos

DevFinder: An Architecture for Recommending Software Development Specialists with Industry-Specific Expertise in Desired Technologies.

Juiz de Fora
2024

Vitor Queiroz de Campos

DevFinder: An Architecture for Recommending Software Development Specialists with Industry-Specific Expertise in Desired Technologies.

Dissertação apresentada ao Programa de Pós-graduação em Ciência da Computação da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do título de Mestre em Ciência da Computação.

Orientador: Prof. Dr. José Maria Nazar David

Coorientador: Prof. Dr. Victor Ströele de Andrade Menezes

Juiz de Fora

2024

Ficha catalográfica elaborada através do Modelo Latex do CDC da UFJF
com os dados fornecidos pelo(a) autor(a)

De Campos, Vitor.
DevFinder: An Architecture for Recommending Software Development Specialists with Industry-Specific Expertise in Desired Technologies. / Vitor Queiroz de Campos. – 2024.
116 f. : il.

Orientador: José Maria Nazar David
Coorientador: Victor Ströele de Andrade Menezes
Dissertação (Mestrado) – Universidade Federal de Juiz de Fora, Instituto de Ciências Exatas. Programa de Pós-graduação em Ciência da Computação, 2024.

1. specialized talet. 2. recommendation. 3. complex networks. 4. large language models. I. David, José Maria Nazar, orient. II. Menezes, Victor Ströele de Andrade , coorient. III. DevFinder: An Architecture for Recommending Software Development Specialists with Industry-Specific Expertise in Desired Technologies.

Vitor Queiroz de Campos

DevFinder: An Architecture for Recommending Software Development Specialists with Industry-Specific Expertise in Desired Technologies

Dissertação apresentada ao Programa de Pós-graduação em Ciência da Computação da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do título de Mestre em Ciência da Computação. Área de concentração: Ciência da Computação.

Aprovada em 06 de setembro de 2024.

BANCA EXAMINADORA

Prof. Dr. José Maria Nazar David - Orientador
Universidade Federal de Juiz de Fora

Prof. Dr. Victor Ströele de Andrade Menezes - Coorientador
Universidade Federal de Juiz de Fora

Profª. Dra. Regina Maria Maciel Braga
Universidade Federal de Juiz de Fora

Profª. Dra. Renata Mendes de Araújo
Universidade Presbiteriana Mackenzie

Juiz de Fora, 22/08/2024.

Documento assinado eletronicamente por **Jose Maria Nazar David**, **Professor(a)**, em 06/09/2024, às 18:51, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Regina Maria Maciel Braga Villela**, **Professor(a)**, em 06/09/2024, às 19:43, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Victor Stroele de Andrade Menezes**, **Professor(a)**, em 10/09/2024, às 13:51, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Renata Mendes de Araujo**, **Usuário Externo**, em 16/09/2024, às 19:16, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no Portal do SEI-Ufjf (www2.ufjf.br/SEI) através do ícone Conferência de Documentos, informando o código verificador **1939339** e o código CRC **EB7BA186**.

I dedicate this work to those who have always believed in the transformative power of education.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to the people who played crucial roles in my academic journey. Firstly, my mother Simone, whose boundless love and unwavering care served as the foundation for my personal development. My father, Noel, contributed significantly to my intellectual growth through his exceptional intellect and creative abilities. My brother, Filipe, has been a guiding force and source of motivation, exemplifying a commendable lifestyle and unwavering loyalty. I extend deep gratitude to my extensive circle of friends and family, whose unwavering support has proven instrumental in propelling me toward my academic goals.

Furthermore, I wish to convey my deepest appreciation to my supervisors, José Maria and Victor Ströele, for their invaluable support. I am also indebted to the professors of the Computer Science Department who generously shared their knowledge and time, playing a pivotal role in the successful completion of this work. Additionally, I extend special thanks to the dedicated professionals in the Postgraduate department, Sarah and Paulo, whose unwavering assistance has been instrumental in article publications, enrollment processes, and more. Your collective contributions have played a crucial role in shaping my academic journey, and for that, I am truly grateful.

I would like to dedicate this paragraph to express my profound gratitude for the privilege of undertaking this study. Pursuing a Master's degree in a country marked by numerous disparities, yet actively engaging in national and international events and contributing to discussions on cutting-edge topics such as artificial intelligence and large language models, represents a rare and invaluable opportunity for which I am deeply thankful.

The journey of learning is often arduous. It demands that we step out of our comfort zones, plunge into the unknown, and risk our time and sanity. In this process, we often discover that our initial assumptions were misguided. However, it is through these moments of failure that new ideas emerge, and solutions are refined. This is the transformative power of science – the ability to embrace setbacks, persist, and discover innovative approaches to problems. For this, too, I am sincerely grateful.

I wish to convey my appreciation to the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001 - for the financial assistance received during my research. I am thankful for the opportunities and resources made available by CAPES, which contributed significantly to the development of my knowledge and skills in the field of computer science. I express sincere gratitude to CAPES for their indispensable support.

"I believe that at the end of the century the use of words and general educated opinion will have altered so much that one will be able to speak of machines thinking without expecting to be contradicted."
Alan Turing

RESUMO

O Desenvolvimento Global de Software (GSD) tem experimentado um aumento significativo nos últimos anos, impulsionado pela crescente importância do software em diversos setores. A demanda por especialistas com habilidades técnicas específicas e experiência em domínios industriais tornou-se crucial para que as organizações mantenham uma vantagem competitiva. No entanto, o ritmo acelerado de mudanças característico da quarta revolução industrial apresenta desafios significativos para atender a essa demanda. Esta revolução evolui exponencialmente, afetando todos os países e indústrias simultaneamente, e exige que a indústria de software se adapte constantemente para acompanhar os rápidos avanços. Para abordar a lacuna de talentos e os desafios impostos pela rápida evolução da tecnologia, este estudo apresenta o DevFinder, uma arquitetura de software projetada para apoiar a busca por especialistas em desenvolvimento de software cujo conhecimento esteja alinhado com domínios industriais e tecnologias específicas. A arquitetura DevFinder utiliza técnicas como redes complexas e "Large Language Models" (LLMs) para processar informações de repositórios de código do mundo real, gerando uma lista classificada de desenvolvedores que atendem às necessidades específicas das partes interessadas. Ao extrair, interpretar e processar dados de repositórios de software, o DevFinder visa preencher a lacuna entre os requisitos técnicos das empresas e a escassez de talentos especializados. A implementação da arquitetura DevFinder foi realizada por meio de três ciclos de Pesquisa em Design Science, culminando na versão final atual. Os requisitos arquiteturais foram identificados através de um estudo de mapeamento sistemático, e as iterações de implementação foram avaliadas em três estudos separados. Os resultados da avaliação confirmam que a arquitetura desenvolvida é capaz de apoiar recrutadores na busca por especialistas em software com a combinação desejada de conhecimento em domínios industriais e expertise técnica. As principais contribuições desta pesquisa incluem especificar os requisitos para uma arquitetura eficaz, desenvolver um modelo de processo de negócios que comunique os processos de extração, filtragem e classificação às partes interessadas e implementar processos para extrair e filtrar repositórios utilizando técnicas de LLM. Além disso, foi definida uma métrica para classificar os colaboradores com base em aspectos de colaboração, e a arquitetura proposta foi implementada com dados reais de repositórios e desenvolvedores de software. Em geral, o DevFinder demonstra o potencial de ajudar as organizações a localizar desenvolvedores de software adequados, abordando assim a lacuna de talentos e garantindo que as empresas de desenvolvimento de software possam acompanhar os rápidos avanços da quarta revolução industrial.

Palavras-chave: especialistas em desenvolvimento de software; recomendação; redes complexas; large language models.

ABSTRACT

Global Software Development (GSD) has experienced a significant surge in recent years, driven by the growing importance of software across various industries. The demand for experts with specific technical skills and industry domain experience has become crucial for organizations to maintain a competitive edge. However, the rapid pace of change characteristic of the fourth industrial revolution presents significant challenges in meeting this demand. This revolution evolves exponentially, affecting all countries and industries simultaneously, and necessitates the constant adaptation of the software industry to keep up with swift advancements. To address the talent gap and the challenges posed by the rapid evolution of technology, this study introduces DevFinder, a software architecture designed to support the search for software development specialists whose knowledge aligns with specific industry domains and technologies. DevFinder leverages advanced techniques such as complex networks and Large Language Models (LLMs) to process information from real-world code repositories, generating a ranked list of developers who meet stakeholders' specific needs. By extracting, interpreting, and processing data from software repositories, DevFinder aims to bridge the gap between the technical requirements of companies and the scarcity of specialized talent. The implementation of DevFinder architecture was carried out through three cycles of Design Science Research, culminating in the current final version. The architectural requirements were identified through a systematic mapping study, and the implementation iterations were assessed in three separate studies. The evaluation results confirm that the developed architecture is capable of supporting job recruiters in finding software specialists with the desired combination of industry domain knowledge and technical expertise. The main contributions of this research include specifying the requirements for an effective architecture, developing a business process model that communicates the extraction, filtering, and ranking processes to stakeholders, and implementing processes to extract and filter repositories using LLM techniques. Additionally, a metric was defined to rank committers based on collaboration aspects, and the proposed architecture was implemented with real-world data from repositories and software developers. Overall, DevFinder demonstrates the potential to aid organizations in locating suitable software developers, thereby addressing the talent gap and ensuring that software development companies can keep pace with the fourth industrial revolution's rapid advancements.

Keywords: software development specialists; recommendation; complex networks; large language model;

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

AI	Artificial Intelligence
BS	Backward Snowballing
CF	Contribution Factor
CSS	Cascading Style Sheets
DSR	Design Science Research
DSP	Decision Support Process
DSS	Decision Support System
DVCS	Distributed Version Control Systems
FS	Forward Snowballing
GPT	Generative Pre-trained Transformer
GQM	Goal, Question, and Metric
HTML	Hypertext Markup Language
ICL	Instructed Commonsense Learning
JSON	JavaScript Object Notation
LLM	Large Language Model
NN	True Negative
NP	False Negative
OAuth	Open Authorization
OWL	Web Ontology Language
PHP	Hypertext Preprocessor
PICOC	Population, Intervention, Comparison, Outcome and Context
PP	True Positive
RQ	Research Question
SMS	Systematic Mapping Study
UI	User Interface
VCS	Version Control Systems

SUMMARY

1 Introduction

This chapter introduces the inspiration behind the study, outlines the primary challenges, and discusses the research methodologies applied, ultimately culminating in a solution proposal.

1.1 Contextualization

Global Software Development (GSD) has experienced a significant surge in recent years, driven by the growing importance of software across various industries (60, 61). As software continues to play a crucial role in driving business operations, organizations increasingly seek individuals with specialized technical knowledge and industry experience to meet their development needs. The demand for experts with specific technical skills—such as proficiency in particular programming languages or adherence to software development standards—combined with industry domain experience is of great interest to companies and institutions involved in software development (145).

This growing demand for specialized talent is further compounded by the rapid pace of change characteristic of the fourth industrial revolution. Unlike previous industrial revolutions that progressed linearly, this revolution evolves exponentially, affecting all countries and industries simultaneously (126, 127). Consequently, the software industry faces the challenge of keeping up with these swift advancements. As a result, software development companies often hire professionals at early stages in their careers to bridge the gap, as finding highly experienced experts has become increasingly difficult (145).

Therefore, addressing this dilemma is crucial for bridging the gap between a company's technical requirements and the scarcity of specialized talent. Developing effective strategies to identify professionals with precise technical skills, whether in specific programming languages, software development standards, or industry-specific knowledge, is paramount to organizations involved in software development (162). This ability to match the right talent to the right roles is essential for maintaining a competitive advantage and fostering innovation in the rapidly evolving software industry.

1.2 Motivation

In response to this dilemma, solutions have emerged addressing the challenge of meeting the growing demand for specialized talent in the software industry amidst the rapid and exponential changes of the fourth industrial revolution. Rostami (2023), for instance, proposes a deep learning-based method to identify agile software teams by focusing on T-shaped experts who have both specialized and broad knowledge, which is critical for adapting to fast-paced technological advancements. This approach improves

the selection process by considering the content of candidates' posts on StackOverflow and also outperforms traditional methods by a significant margin (48). Similarly, Vergne (2014) uses social network analysis, both syntactic and semantic, to identify experts in Global Software Development. Analyzing collaboration patterns and expertise with machine learning and ontologies helps pinpoint key members in a constantly changing industry (49).

Other studies also present innovative solutions to the challenge of rapidly changing demands for expertise. For example, Tuarob (2021) introduces RECAST, an intelligent recommendation system that considers both technical skills and teamwork compatibility to suggest optimal team configurations. This system uses machine learning to generate a scoring function from heterogeneous features, addressing the complexity of forming effective teams in large-scale software projects (46). Additionally, Hauff (2015) offers a practical solution by matching GitHub developer profiles to job advertisements, automating the process of finding suitable candidates based on their activities and contributions. These approaches collectively demonstrate a shift towards leveraging advanced algorithms and data-driven methods to bridge the talent gap and ensure that software development companies can keep pace with the fourth industrial revolution's rapid advancements (51).

Also, as a key aspect of the area, the emergence of artificial intelligence (AI), specifically Large Language Models (LLMs), has reshaped the landscape of research across various domains, with implications in Natural Language Processing (NLP) (30). LLMs, characterized by their massive training parameters, have demonstrated capabilities in tasks that involve language understanding, generation, and recommendation. For instance, Chat-Rec (5), which leverages ChatGPT for user interactions, has demonstrated the potential to enhance recommendation accuracy and explainability. Similarly, Zhang (2023) has harnessed T5, an LLM-based recommendation system, to enable users to express their preferences in natural language, significantly improving recommendation performance over traditional user-item interaction-based systems. These developments underscore LLMs' transformative impact and potential to augment recommendation systems (160).

Nevertheless, despite the works presented, there are still some challenges to be further comprehended regarding the recommendation of software developers for specific open positions. Therefore, this study introduces distinctive contributions in contrast to existing related work. As a prevalent contribution, this research offers an architecture, called DevFinder, that aims to tackle the challenge of finding specialists whose knowledge aligns with experience in specific industry domains but also specific technologies. Thus, the research problem addressed in this work is to support the search for software development specialists who align knowledge in specific industry domains to desired technologies. DevFinder architecture was designed to gather information regarding software developers in real-world code repositories, process this information using techniques such as complex networks and large language models, and generate a ranked list of software developers

who attend to the specific needs of stakeholders. The goal was to extract, interpret and process data from the repositories of software developers, in the context of specific industry domains and technologies inputted by the stakeholder. Thus, the research question posed within the scope of this study is ***"How can DevFinder architecture support the search for software experts whose knowledge aligns specific industry domains with specific technologies?"***

1.3 Objectives

This work aims to support stakeholders in decision-making processes through an approach that uses complex networks and Large Language Models to rank real-world software developers. The DevFinder architecture aims to support job recruiters in finding software specialists whose knowledge aligns with experience working in specific industry domains with experience working with specific technologies.

The implementation of DevFinder was carried out through three cycles of Design Science Research, culminating in the current final version. Thus, after conducting a Systematic Mapping study (138), the architectural requirements were identified and the implementation iterations were assessed in two separate studies (136, 137). Therefore, the main contribution of this research lies in the DevFinder software architecture that leverages the capabilities of complex networks and Large Language Models to recommend software specialists whose knowledge aligns specific software technologies to experience in desired industry domains. The implementation and assessment of the architecture were performed, presenting results that demonstrate that the developed architecture is, in fact, capable of supporting companies and job recruiters in the task of finding such software specialists.

To achieve these objectives, the following specific objectives were considered:

- Specify the requirements for an architecture capable of supporting stakeholders on the task of finding software specialists;
- Specify a Business Process Model that encompasses the requirements of the architecture, capable of communicating the extraction, filtering, and ranking processes to stakeholders;
- Specify and implement a process that can extract real repositories and software developers from a version control system;
- Specify and implement a process that can filter the retrieved repositories and software developers using Large Language Model techniques.
- Define a metric to rank committers from the retrieved repositories, taking into account collaboration aspects;

- Specify and implement a process that can rank the retrieved software developers using the defined metric;
- Implement the proposed architecture with real-world data from repositories and software developers.

1.4 Outline

This work is divided into six chapters. Chapter 2 explores the theoretical foundation for this work, providing a framework for understanding the subsequent chapters. Chapter 3 delves into the systematic mapping conducted, offering insights into the research process underpinning the DevFinder architecture's development. Chapter 4 details the various development DSR cycles of DevFinder, shedding light on its creation's iterative and evolving nature. Moving forward, Chapter 5 evaluates and discusses the proposed architecture, presenting a comprehensive analysis of its strengths and potential areas for improvement. Finally, Chapter 6 serves as the concluding chapter, summarizing key findings, reflecting on the journey undertaken, and outlining potential avenues for future research and development.

2 Theoretical Foundation

2.1 Industry Domains and Technical Knowledge

In this study, industry domains refer to distinct sectors or fields within the broader economic landscape, characterized by common patterns of operation, shared technologies, and similar business models. Drawing insights from various perspectives in strategic management and information technology literature (153, 154, 155), industry domains encompass not only the structural and competitive dynamics of industries but also the transformative impact of digitalization and enterprise systems. These domains serve as the contextual backdrop within which organizations operate, navigate challenges, and leverage opportunities, emphasizing the importance of understanding industry-specific factors in driving strategic alignment, technological investments such as enterprise resource planning (ERP) systems, and the interplay between digital transformation and employee competency.

Similarly, in our study, technical knowledge refers to the comprehension and expertise within specific technical domains. It is characterized by acquiring specialized knowledge and skills that enable individuals to proficiently navigate and contribute to their respective fields (158). Drawing insights from scholarly works (156, 158, 157, 159), technical knowledge is identified as possessing high levels of expertise in particular technical domains while often requiring less formal educational attainment, as articulated by Rothwell's classification of skilled technical worker (158). Furthermore, Tarafdar (2007) exploration of information systems competencies highlights the importance of knowledge management and collaboration within technical contexts, emphasizing the role of technical knowledge in facilitating process innovation and organizational advancement (159). Thus, technical knowledge emerges as a multifaceted construct, encompassing specialized expertise, practical skills, and the ability to innovate within technical domains.

Industry domains and technical knowledge are key components of this study, as the primary goal is to identify software experts whose expertise aligns with specific industry domains and technologies. Therefore, the definitions provided are carefully crafted to distinguish the concepts relevant to this research from other interpretations the reader might encounter.

2.2 Large Language Models

In recent years, the field of artificial intelligence has witnessed a paradigm shift with the advent of Large Language Models (LLMs). These models, based on deep learning architectures, have demonstrated unprecedented capabilities in natural language understanding and generation, significantly impacting various domains such as natural

language processing, information retrieval, and human-computer interaction.

Large Language Models have their roots in the broader field of artificial neural networks. The development of deep learning techniques, particularly the rise of Transformer architectures, has been instrumental in the evolution of LLMs. The seminal work of Ashish (2023) introduced the Transformer model, which laid the foundation for subsequent advancements in large-scale language understanding. The self-attention mechanism introduced in the Transformer architecture is central to the success of LLMs. This mechanism allows the model to weigh different parts of the input sequence differently, capturing long-range dependencies and contextual information effectively (28).

LLMs are often pre-trained on vast corpora of text data using unsupervised learning objectives, such as language modeling or masked language modeling. This pre-training phase is followed by fine-tuning on specific tasks of interest, enabling the models to adapt to domain-specific requirements (29).

LLMs have demonstrated good performance in tasks such as sentiment analysis, named entity recognition and question answering. Models like BERT have set new benchmarks by capturing intricate contextual relationships within language (29). OpenAI's GPT (Generative Pre-trained Transformer) series, has showcased the ability to generate coherent and contextually relevant text. These models have found applications in content creation, language translation, and even code generation (30).

Nonetheless, the rise of LLMs has raised ethical concerns related to bias, misinformation, and the responsible use of AI. Researchers emphasize the importance of addressing these challenges to ensure the fair and ethical deployment of LLMs in various applications (31).

LLMs play a relevant role in this study as they form the foundation for two key modules: filtering and insight generation within the proposed architecture. These modules were integrated into the DevFinder architecture as a solution to address specific issues identified in a previous iteration of the system.

2.3 Ontologies

To facilitate the exchange and reutilization of knowledge across diverse systems, it becomes imperative to establish a shared vocabulary for representing this knowledge. Gruber (1995) introduced the term ontology, borrowing it from philosophy, and defined it within the computational context as a formal and explicit specification of a shared conceptualization. This conceptualization provides a simplified and abstract perspective of the world, crafted for specific purposes. Generally, an ontology delineates a domain vocabulary, comprising definitions of classes, relationships, and functions. Its utility lies in fostering a common understanding of information structure among individuals or software

agents, facilitating the reuse of domain knowledge, elucidating assumptions about the domain, segregating domain knowledge from operational knowledge, and analyzing domain knowledge (139).

The Ontology Web Language (OWL) was specifically designed to enhance the interpretation of Web content through ontologies, providing a more comprehensive vocabulary with formal semantics compared to other languages like XML, RDF, and RDF Schema (RDF-S) (142). OWL's advantage lies in its suitability for situations where document information needs processing by applications, as opposed to cases where content is intended solely for human consumption. It serves as a language for constructing ontologies that offer high-level descriptions of Web content, employing class hierarchies to describe domain concepts and establishing relationships between classes using properties (142).

OWL and Semantic Web Rule Language (SWRL) stand out as the primary languages of the Semantic Web. OWL not only represents data as instances of OWL classes (referred to as individuals) but also provides mechanisms for reasoning and manipulating the data. Additionally, OWL features an axioms language for precisely defining the interpretation of concepts in an ontology (143). SWRL complements OWL by enabling the creation of rules expressed in terms of OWL concepts, facilitating reasoning about OWL individuals. Noteworthy is SWRL's ability to support user-defined predicates, known as built-ins, allowing the formulation of rules with diverse functionalities, including currency conversion, temporal manipulations, and taxonomy searches (144). While the arguments for these built-ins generally must be OWL DL property values, some built-in libraries may also support class or property built-in arguments, though reserved for use in OWL Full ontologies (143).

As a way to represent and conceptualize data, the concepts and techniques behind ontologies were employed to the first version of DevFinder architecture and, for this iteration, the concepts represent a crucial role.

2.4 Final Remarks of the Chapter

This Theoretical Foundation chapter has provided an overview of fundamental keywords to this study: industry domains and technical knowledge, large language models (LLMs), and ontologies. The emergence of Large Language Models (LLMs), such as BERT and the GPT series, signifies a transformative era in natural language understanding. Ontologies, formalizing shared conceptualizations, facilitate knowledge exchange.

The synergies between these foundations set the stage for the subsequent chapters, where practical applications and intersections are explored. Navigating the complexities of connectivity, language understanding, decision facilitation, and knowledge representation, this research aims to contribute to an understanding of contemporary complex systems.

3 Systematic Mapping Study

This chapter presents the Systematic Mapping Study (SMS) conducted and published in Willey Journal Of Software: Evolution And Process (138). The study aims to understand how recommendation approaches can effectively align technical knowledge with industry domains of software developers. By exploring this intersection, we seek to uncover insights that can inform the design, development, and optimization of systems tailored to specific industry needs. Following the guidelines for the Snowballing procedure (79), and the hybrid systematic literature mapping guidelines (74, 75), we employed a combination of database search and snowballing techniques. Through this process, we analyzed a total of 1251 papers, resulting in the classification of these papers into 21 distinct studies.

Therefore, to guide the study of this chapter, we focus on the following Research Questions (RQ): *How is the alignment of industry domains and technical expertise used in systems that recommend software developers?*

The main intent of answering this question is to investigate and understand how technical expertise and industry domains interact to shape the functionality, effectiveness, and performance of systems that recommend software developers. By addressing this question, the study seeks to uncover insights that can lighten the strategies, practices, and mechanisms through which industry-specific knowledge and technical proficiency are integrated to design, develop, and optimize such approaches.

This SMS presents contributions to software processes as it addresses overlooked aspects in systems that recommend software developers by identifying expert developers with both technical proficiency and diverse industry experience, the research enhances software development processes. For software practitioners, the study offers valuable insights into industry practices, technologies, and data sources for systems that recommend software developers. Thus, the main contributions of this work are: i) Identification and classification of 21 studies proposing software developer recommendation approaches over the optics of the alignment of technical skills to specific industry domains; ii) Our analysis yielded results regarding the prevailing technologies and data sources utilized in the examined solutions; iii) The aggregation of valuable insights into the current trends in the field, shedding light on future directions for research and development; iv) Bring spotlights to specific characteristics that a new RS of software experts that align technical skills to an industry domain knowledge should take into account.

By studying and analyzing data from the 21 selected studies, our research unveils current trends in this research field and offers a practical guide for future researchers interested in creating systems that recommend software developers that connect technical expertise with specific industry domains. This guide provides valuable insights, suggestions,

and proven methods for designing and developing effective RS systems that bridge the technical skills-industry knowledge gap.

A key point to consider is that, although the final architecture proposed the use of complex networks and Large Language Models (LLMs), these approaches were not initially considered during the systematic mapping process. The primary goal of the systematic mapping was to analyze existing work in the field to identify methods for recommending software developers in our target context. As a result, the findings from this SMS ultimately inspired the authors to incorporate complex networks and LLMs into the proposed solution.

3.1 SMS Related Work

In the software engineering field, several studies (88, 89, 132) have explored different aspects related to recommendation approaches, metrics, team formation, labor market analytics, code repository analysis, and requirement engineering challenges. This section compares our study on aligning technical knowledge to an industry domain with six relevant studies.

The first study, "Systematic mapping of recommendation systems for requirements engineering" (81), undertakes a systematic mapping to explore recommendation systems tailored for requirements engineering processes. Through this exploration, the study provides an overview of such systems, detailing their characteristics and the current state of validation. By analyzing the resulting maps, the study concludes and identifies limitations in existing research, consequently pinpointing areas for future investigation. Additionally, the study utilizes the insights gained from the mapping to lay the groundwork for future work, specifically focusing on the development of a recommendation system aimed at aiding product managers in making informed decisions regarding the allocation of requirements across subsequent releases, while factoring in constraints such as time, effort, quality, and resources.

The second study, "Metrics to quantify software developer experience: a systematic mapping" by Brasil-silva (2022), investigates the relationship between developer experience and software maintainability, particularly focusing on the metrics used to quantify developer experience in 34 different works. It emphasizes the crucial role experienced developers play in defining architectural solutions that enhance a system's maintainability, thus facilitating timely and cost-effective development of new features and defect fixes. Through a systematic mapping, the study catalogs various metrics employed to measure developer experience, categorizing them based on the type of experience they represent and their purpose of use. The findings reveal that a majority of the selected studies utilize metrics centered around counting developer activities within codebases (82).

The study by Costa (2020) on "Team Formation in Software Engineering" examines

the current state of research on software team formation, identifying and analyzing 51 primary studies out of a total of 2516. Through this analysis, the study aimed to classify the research based on the methods used, overall quality, characteristics of formed teams, and proposed solutions. It found that the majority of studies utilize search and optimization techniques, with technical attributes being the most common factors considered in team formation. The study also introduced a taxonomy for organizing the knowledge in software team formation, highlighting the prevalent use of search-based approaches that combine search and optimization techniques with technical attributes. However, there is a growing trend towards incorporating non-technical attributes as supplementary information. Identified research gaps include the subjective nature of software team formation and limitations in the scalability of proposed solutions (83).

The fourth study, "Extracting Knowledge from On-Line Sources for Software Engineering Labor Market: A Mapping Study" by Papoutsoglu (2019), addresses the evolving needs of the software engineering labor market through the application of data analytics sourced from digital platforms. By conducting a Systematic Mapping Study, researchers categorized and evaluated 86 primary studies to identify digital sources suitable for labor market analytics in software engineering. These sources encompass a variety of skill types, methods of data analysis, and goals. The study's objectives include facilitating continuous learning and training in new technologies, improving job matching between employers and candidates, and detecting expertise. By leveraging digital sources, the study seeks to provide timely and accurate insights to stakeholders, including both employers and employees, in the software engineering labor market (84).

Another study, "Characterizing the hyperspecialists in the context of crowdsourcing software development" (87), explores the behavior of hyperspecialists, a specific contributor profile, on crowdsourcing platforms, focusing on software development challenges within the Topcoder platform. Through quantitative analysis of 664 developers' participation over 18 months, the research identified traits associated with hyperspecialization, such as consistent engagement in specific challenge types and technologies. Notably, a high dropout rate of 66% was observed among participants during the study period. Hyperspecialists, who constituted a significant portion of developers, demonstrated distinct behavior and characteristics compared to non-specialists, including a lower winning rate. This study sheds light on the dynamics of crowdsourcing platforms, emphasizing the importance of understanding participant behavior for the success of such models.

3.1.1 Comparative Analysis

The comparative analysis among the presented studies showcases diverse investigations within the realm of software engineering and related fields. While each study employs systematic mapping techniques to explore different facets of the discipline, they

converge on the significance of data-driven insights for decision-making processes. The first study by Mohebzada (2012) delves into recommendation systems tailored for requirements engineering, aiming to enhance decision-making for product managers (81). In contrast, Brasil-silva (2023) focuses on quantifying software developer experience, underlining its impact on software maintainability (82). Both studies emphasize the role of informed decision-making in optimizing software development processes. Similarly, Costa (2020) examines software team formation, highlighting the necessity of efficient team structuring for project success. The integration of technical and non-technical attributes in team formation resonates with the multifaceted approach advocated by the first two studies (83). Further expanding the scope, Papoutsoglu (2019) addresses the evolving software engineering labor market, emphasizing the importance of data analytics for talent management and job matching (84) . Lastly, Neira (2018) explores the behavior of hyperspecialists in crowdsourcing software development, revealing insights into participant dynamics crucial for platform efficacy (87). Collectively, these studies underscore the interdisciplinary nature of software engineering research and the pivotal role of data-driven insights in shaping its evolution.

In comparison to the existing studies outlined in the comparative analysis, our research presents a focus on the alignment of technical knowledge with industry domains within recommendation approaches and, to the best of our knowledge we could not identify any other studies focusing on such aspects. While the referenced studies explore diverse aspects of software engineering, such as developer experience, team formation, labor market analytics, and crowdsourcing dynamics, our study specifically targets the integration of technical proficiency and industry-specific knowledge in systems that recommend software developers. By conducting a Systematic Mapping Study and analyzing 21 distinct studies, this study offers insights into industry practices, technologies, and data sources for recommender approaches providing a practical guide for future researchers interested in creating systems that recommend software developers that bridge the technical skills-industry knowledge gap. Therefore, this research distinguishes itself for its specific focus on aligning technical expertise with industry domains within recommendation approaches, offering different perspectives and contributions to the field.

3.2 Systematic Mapping Study

As an approach for identifying and consolidating evidence, a systematic literature mapping is employed (72, 68) in this study. The primary objective of a systematic mapping is to examine, assess, and interpret relevant studies in the literature concerning the research questions. This process enables gathering evidence to identify research gaps and opportunities, being an important evidence-based approach to software engineering (70).

For this systematic mapping, the years 2012 to 2023 were selected to provide a comprehensive overview of the contemporary landscape of global software engineering (GSD). This time frame was chosen to ensure relevance, capture recent trends, and avoid potential obsolescence, as suggested by previous studies (76, 91, 92). While acknowledging the historical foundations of technical knowledge in software development spanning decades, we focused on this dynamic period characterized by rapid advancements in technology, methodologies, and global collaboration practices. By prioritizing recent data and insights, our study aims to offer valuable insights into the current state of GSD, informing practitioners, researchers, and policymakers alike.

To mitigate research bias (73), a protocol was established. This section provides an overview of the steps to define the objectives and formulate the protocol. Reproducibility is a crucial aspect of systematic mappings, and the protocol plays a significant role in ensuring its attainment.

The organization of this mapping study follows key activities (71), which include planning, conducting, and reporting the study. In order to address the difficulties associated with database management and the comprehensive identification of synonyms during database searches, a hybrid approach was followed (74). We selected Scopus[1] as the digital library for this study, as it is widely recognized as the largest abstract and citation database of peer-reviewed literature (74, 75).

3.3 SMS Planning

In the planning phase, we established the study objectives and developed a protocol. This protocol outlines the methodology employed in the systematic review and mapping, ensuring that potential researcher biases are minimized. Additionally, it is crucial for the systematic review and mapping to be reproducible, and the protocol plays a significant role in meeting this requirement. This section provides a summary of the protocol (70).

3.3.1 Scope definition

To define the scope of our study, we have considered the PICOC method as proposed by Petticrew and Roberts (76). This method aligns with our research questions and helps structure our investigation.

- Population: Studies on systems that recommend software developers
- Intention: Studies related to software developers recommendation
- Comparison: Not applicable

[1] https://scopus.com

- Outcome: Solutions that recommended software developers aligning technical knowledge with industry domains
- Context: Global Software Development

Thus, the objective of the systematic mapping is to address the following research question (RQ): *How is the alignment of industry domains and technical expertise used in systems that recommend software developers?*

By investigating this RQ, we aim to reveal the methods, approaches, and mechanisms used to combine industry-specific expertise and technical skills in the creation, enhancement, and optimization of systems that recommend software developers. This undertaking has the potential to offer insights to professionals, scholars, and stakeholders engaged in the advancement and fine-tuning of systems that recommend software developers customized for distinct industry sectors. Specific questions to attend the main RQ were designed in order to detail the intrinsic aspects of the subject and are presented in the Discussion section.

3.3.2 Inclusion and exclusion criteria

The process employed to include and exclude papers was structured around nine inclusion criteria (IC) and ten exclusion criteria (EC). The considered IC were: IC1: Industry domains are taken into consideration in the solution of the study; IC2: Technical knowledge is taken into consideration in the solution of the study; IC3: The approach tested or applied in the industry; IC4: The paper approaches software OR system OR application; IC5: The paper discusses recommendation approaches of experts; IC6: The paper is a primary study; IC7: The paper is reported in a peer-reviewed conference journal; IC8: The paper is written in English; and IC9: The paper proposes a pragmatic recommendation approaches.

The following criteria were set for exclusion: EC1: The paper was published 10 years ago or more, EC2: The proposed solution analyses but does not order collaborators in order to generate a recommendation list and EC3: The proposed solution strictly adheres to the academic application context.

And the following criteria were set for exclusion: EC1: The paper does not approach software OR system OR application; EC2: The paper does not discuss the recommendation approaches of experts; EC3: The paper does not propose a pragmatic commendation system; EC4: The paper is not a primary study, EC5: The paper is not reported in a peer-reviewed conference journal, EC6: The paper is not written in English, EC7: The paper was published 10 years ago or more, EC8: The proposed solution analyses but does not order collaborators in order to generate a recommendation list, EC9: The proposed

solution strictly adheres to the academic application context, and EC10: The proposed solution strictly adheres to the open-source application context.

3.3.3 Query String

We first identified the key terms from the defined research questions and PICOC, along with their alternative spellings and synonyms. To construct the search string, synonyms, and alternate spellings were combined using Boolean OR, and then each set of terms was combined using Boolean AND to form a single string. The final search string used was:

("global software development" OR "collaborative software development" OR "collaborative software engineering" OR "dispersed locations" OR "dispersed teams" OR "distributed development" OR "distributed teams" OR "distributed work" OR "geographically distributed software" OR "global software engineering" OR "global software teams" OR "multi-site development" OR "offshore" OR "offshoring") AND ("expert" OR "architect" OR "coder" OR "collaborator" OR "developer" OR "engineer" OR "programmer") AND ("recommendation system" OR "advisory system" OR "recommender system" OR "suggestion system")

The constructed query string was analyzed and tested by two specialists in the area, and a set of potential primary studies was also defined to validate the search string accuracy in the selected databases and to determine if the search retrieved relevant studies (77). In that sense, (47) and (40) were known potential interest studies used to assess whether the search string successfully identified relevant studies.

To mitigate the risk of overlooking pertinent studies, we implemented a control group of papers during the search process. This control group was designed to ensure that a specific set of papers would consistently appear in the search engine results (39, 40).

3.4 Systematic mapping conduction

The systematic mapping of the literature was conducted following the outlined steps, also shown in Figure 1.

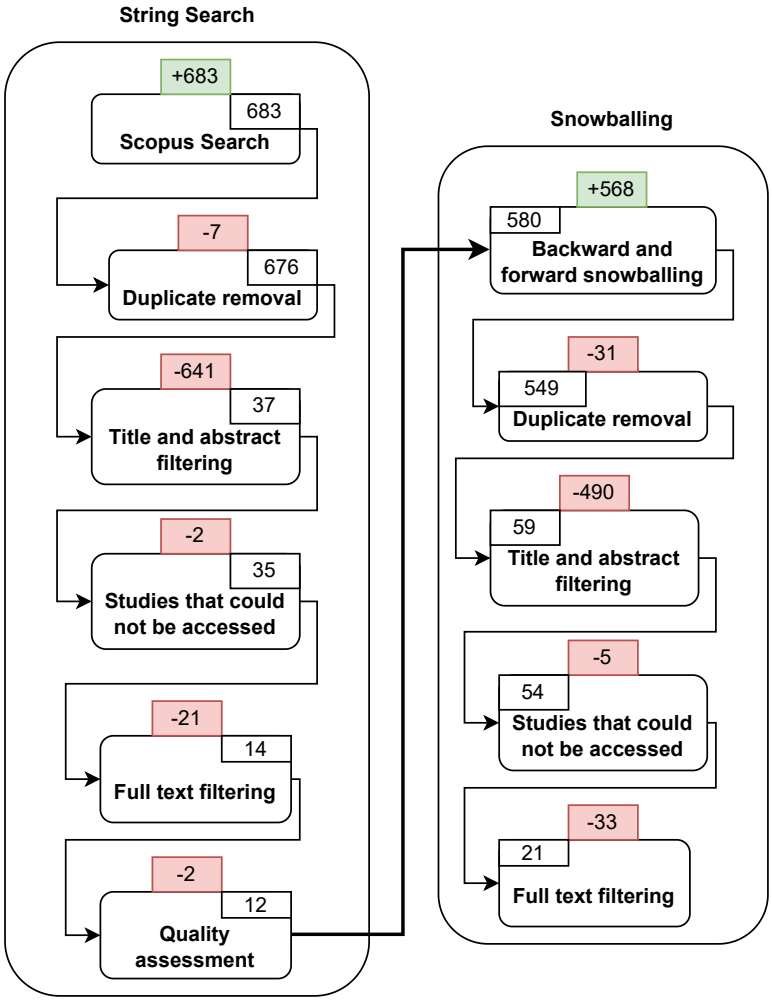
3.4.1 String Search on Scopus

To evaluate the initial corpus of systematic mapping studies, a search was performed in the Scopus database using the mentioned search string, resulting in 683 studies in this first stage.

The obtained studies from the search were then input into the parsif.al[2] system, which facilitated the subsequent steps of duplicate removal, filtering, and quality assess-

[2] https://parsif.al

– Figure 1 - Systematic Mapping Overview



ment. Parsifal is a system designed to aid researchers in conducting systematic literature reviews specifically tailored to the field of Software Engineering following the guidelines of Kitchenham (71).

After removing 7 duplicates, the remaining 676 studies underwent a thorough evaluation, with particular attention given to the title and abstract. This allowed for filtering 641 non-relevant studies, focusing the analysis on those most aligned with the research objectives. During the assessment, two papers could not be fully accessed due to limited availability. Nonetheless, the mapping process continued, utilizing the available information from these papers to the fullest extent possible.

Subsequently, to ensure the quality and reliability of the included studies, a quality assessment was conducted. As recommended by Kitchenham (2007), it is essential for researchers to create quality checklists that aid in evaluating individual studies. These

checklists serve as valuable tools during the paper selection process, providing support and guidance. In alignment with these guidelines, a customized quality assessment checklist was developed to ensure a thorough and rigorous evaluation of the selected papers (71).

Finally, an examination of the full texts of the selected studies was undertaken. This in-depth reading further refined the selection by eliminating studies that did not meet the predetermined relevance criteria, where 21 studies were identified as non-relevant and 14 studies remained.

In the full-text reading, the papers underwent an evaluation process using the developed quality assessment checklist in Table 1, where QA stands for Quality Assessment. Each study was thoroughly read, and its quality was assessed based on the 11 questions, listed in the last paragraph of this section. Each question had a scoring system where a "Yes"response scored 1 point, a "Partial"response scored 0.5 points, and a "No"response scored 0 points. This scoring system allowed each paper to achieve a score ranging from 0 to 11 points.

Table 1 – Quality Assessment Questions

ID	**Question**
QA1	Does the study present one or more approaches to recommend collaborators?
QA2	Is technical knowledge taken into consideration in the solution of the study?
QA3	Are industry domains taken into consideration in the solution of the study?
QA4	Is empirical research conducted on the study?
QA5	Is data properly described?
QA6	Are data statistical methods properly defined and justified?
QA7	Is the relation between data, interpretation, and conclusion coherent?
QA8	Do the authors clearly specify the research goals?
QA9	Are all research questions resolved?
QA10	Are issues related to validity and/or reliability discussed?
QA11	Are the findings debated, rather than only displayed?

To establish a cutoff point for inclusion, the first quartile ($11/4 = 2.75$) was utilized. Any paper scoring below 2.75 was excluded from the final list to ensure the exclusion of low-quality works. Alongside the evaluation process, an important task of data extraction was also performed, gathering relevant information from the selected papers.

The following checklist was considered for this step: i) Does the study present one or more approaches to recommend collaborators?; ii) Is technical knowledge taken into consideration in the solution of the study?; iii) Are industry domains taken into consideration in the solution of the study?; iv) Is empirical research conducted on the study?; v) Is data properly described?; vi) Are data statistical methods properly defined and justified?; vii) Is the relation between data, interpretation, and conclusion coherent?; viii) Do the authors clearly specify the research goals?; ix) Are all research questions

resolved?; x) Are issues related to validity and/or reliability discussed?; xi) Are the findings debated, rather than only displayed?

3.4.2 Snowballing

Snowballing is a technique that involves leveraging the reference list of a paper or the citations to that paper in order to identify additional relevant papers (79).

This study follows hybrid systematic mapping guidelines (74), combining a database search and snowballing techniques. The process includes applying inclusion and exclusion criteria to the retrieved studies, and then using the selected studies as an initial set for both Backward Snowballing (BS) and Forward Snowballing (FS) approaches.

In this study, both BS and FS techniques were employed, leading to the identification of 568 new studies. Out of these, 31 duplicate studies were identified and subsequently removed. Subsequently, title and abstract filtering was conducted, resulting in a reduction of 490 studies. However, during the full-text filtering stage, it was found that 5 studies were inaccessible. Eventually, after reading the full texts of the remaining studies, a total of 33 studies were filtered, yielding a final selection of 21 studies for this systematic mapping. All the studies returned from the snowballing passed the quality assessment.

An external link[3] to a comprehensive table is available, featuring a complete list of studies obtained from both the Scopus search and the snowballing process. This table contains essential information such as study titles, authors, sources, and the final status of each study that has undergone the described process, offering resources for cross-referencing and verifying study-related details.

3.5 Systematic Mapping Report

After conducting the search using the specified string in the database, a total of 12 studies were identified which are presented in Table 2. These studies, are denoted by an ID starting with the letter R (which stands for “returned”), followed by a corresponding number, that serves as the basis for the subsequent discussion.

[3] https://tinyurl.com/msmpfece

Table 2 – Selected studies returned from the search string

ID	Title	Published at	Org.	Type	Ref.
R1	Developer recommendation for Topcoder through a meta-learning based policy model	Empirical Software Engineering	Springer	Journal	(39)
R2	Harnessing global expertise: A comparative study of expertise profiling methods for online communities	Information Systems Frontiers	Springer	Journal	(40)
R3	Replacing the Irreplaceable: Fast Algorithms for TeamMember Recommendation	24th International Conference on World Wide Web	ACM	Conference	(41)
R4	The Zen of Multidisciplinary Team Recommendation	Journal of the Association for Information Science and Technology	Wiley	Journal	(42)
R5	Identifying Reputable Contributors in Collaborative Software Development Platforms	International Conference on Research Challenges in Information Science	IEEE	Conference	(43)
R6	Collaboration Analysis in Global Software Development	23rd International Conference on Computer Supported Cooperative Work in Design	IEEE	Conference	(44)
R7	Recommending External Developers to Software Projects based on Historical Analysis of Previous Contributions	XXXIII Brazilian Symposium on Software Engineering	ACM	Conference	(45)
R8	Automatic team recommendation for collaborative software development	Empirical Software Engineering	Springer	Journal	(46)
R9	CoopFinder: Finding Collaborators Based on Co–Changed Files	IEEE Symposium on Visual Languages and Human-Centric Computing	IEEE	Conference	(47)
R10	A deep learning-based expert finding method to retrieve agile software teams from CQAs	Information Processing & Management	Elsevier	Journal	(48)
R11	Expert Finding Using Markov Networks in Open Source Communities	Advanced Information Systems Engineering: 26th International Conference	Springer	Conference	(49)
R12	A broad approach to expert detection using syntactic and semantic social networks analysis in the context of Global Software Development	Journal of Computational Science	Elsevier	Journal	(50)

Table 3 provides an overview of the 9 studies discovered through the snowballing process. Each study is identified by an ID starting with the letter S (representing

"snowballing"), followed by a unique number. This table serves as a valuable reference for the ensuing discussion in the following section.

Table 3 – Selected studies from snowballing process

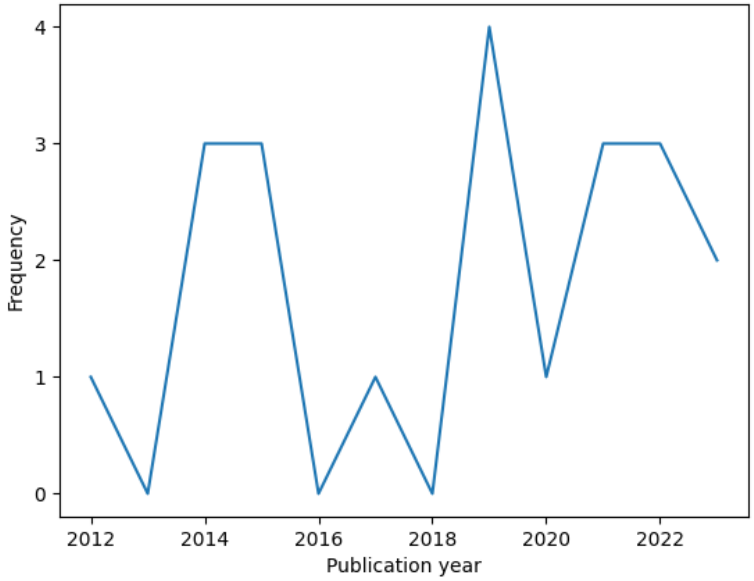
ID	Title	Published at	Org.	Type	Ref.
S1	Matching GitHub developer profiles to job advertisements	12th Working Conference on Mining Software Repositories	IEEE	Conference	(51)
S2	Expert Recommendation in OSS Projects Based on Knowledge Embedding	2017 International Workshop on Complex Systems and Networks	IEEE	Conference	(52)
S3	Mining Shape of Expertise: A Novel Approach Based on Convolutional Neural Network	Information Processing & Management	Elsevier	Journal	(53)
S4	DevRec: Multi-Relationship Embedded Software Developer Recommendation	IEEE Transactions on Software Engineering	IEEE	Journal	(54)
S5	Identifying Experts in Software Libraries and Frameworks among GitHub Users	16th International Conference on Mining Software Repositories	IEEE	Conference	(55)
S6	Intern retrieval from community question answering websites: A new variation of expert finding problem	Expert Systems with Applications	Elsevier	Journal	(56)
S7	Assessing Developer Expertise from the Statistical Distribution of Programming Syntax Patterns	25th International Conference on Evaluation and Assessment in Software Engineering	ACM	Conference	(57)
S8	Mining Experienced Developers in Open-Source Projects	17th International Conference on Evaluation of Novel Approaches to Software Engineering	SCITEPRESS	Conference	(58)
S9	CodeCV: Mining Expertise of GitHub Users from Coding Activities	2nd International Working Conference on Source Code Analysis and Manipulation	IEEE	Conference	(59)

3.5.1 Research trends over the years

To map and understand the research area, we aim to identify key aspects by observing trends in publication years. By analyzing these factors, we can gain insights into the research landscape and its evolution over time.

A noticeable concentration of papers is observed in 2019, 2021, and 2022, with four papers published in each of these years, as shown in Figure 2. This concentration implies that these particular years saw a high research activity and publication on the subject matter.

– Figure 2 - Publication over the years



The analysis of the studies also suggests the emergence of new trends during those years, attracting significant attention from researchers, also summarized in Table 4.

Table 4 – Correlation between trends and the year of publication of the studies

	Deep Learning and AI	**Collaboration and Social Networks**	**Evolution and Improvement**
2012-2014		R2, R4, R8	
2015-2017		R7, S1, S5, S2	R3, R11
2018-2020		R6, S3, S6	
2021-2023	S7, S3	R10, R11, R9, S8, S9, S4, S6	R1, S7, S2, S5

The Emergence of Deep Learning and AI: Several papers published in recent years (2020-2023) have mentioned deep learning, meta-learning, and convolutional neural networks (CNNs). This issue indicates an increasing focus on leveraging advanced machine learning and artificial intelligence techniques in the context of expert finding, team recommendation, and collaborative software development. The incorporation of these technologies suggests a shift towards more sophisticated and data-driven approaches to enhance the effectiveness of recommendation approaches.

Collaboration and Social Networks: A subset of papers explores the analysis of collaboration and social networks in the context of software development. These papers

highlight the importance of understanding and leveraging the relationships, interactions, and social structures among developers to improve expert findings and team recommendations. Syntactic and semantic analysis, as well as historical analysis of previous contributions, are mentioned as techniques to uncover valuable insights from collaboration networks.

Evolution and Improvement: The publication years of the papers suggest an evolution and ongoing improvement of research in the field of expert findings and team recommendations. Earlier papers published in 2012 and 2014 lay the groundwork for subsequent studies, while more recent papers build upon and extend previous approaches. This trend suggests a continuous effort to refine and enhance the accuracy, efficiency, and applicability of recommendation approaches in real-world software development scenarios.

3.6 Discussion

In order to address the Research Question (RQ) of this systematic mapping, which focuses on the alignment of industry domains and technical knowledge in recommendation approaches, three specific questions were formulated, as follows:

- How do authors describe the approach to recommend collaborators?
- What are the data sources for the recommendation approach?
- Could the described approach be used to recommend collaborators aligning desired technical skills to industry domains?

These questions served as guidelines for extracting relevant data during the full-text reading process. The subsequent discussion debates findings over the extracted answers to these three questions.

3.6.1 The described approaches

One common approach observed in multiple papers is the use of machine learning models for recommending collaborators. These models leverage various data sources such as social networks, coding repositories, and QA websites like StackOverflow. For example, Paper R10 proposes a deep learning-based method that utilizes candidate posts on StackOverflow to estimate their relevant knowledge and select team members. Paper R12 (Table 2) presents a broad approach using syntactic and semantic analysis in social networks for global software development, incorporating machine learning algorithms to explore the network and ontology to enrich the data semantically.

Another trend is the consideration of different factors and criteria in the recommendation process. Papers like R8 and R1 (Table 2) focus on recommending collaborators for specific tasks or challenges, filtering out developers who are unlikely to participate,

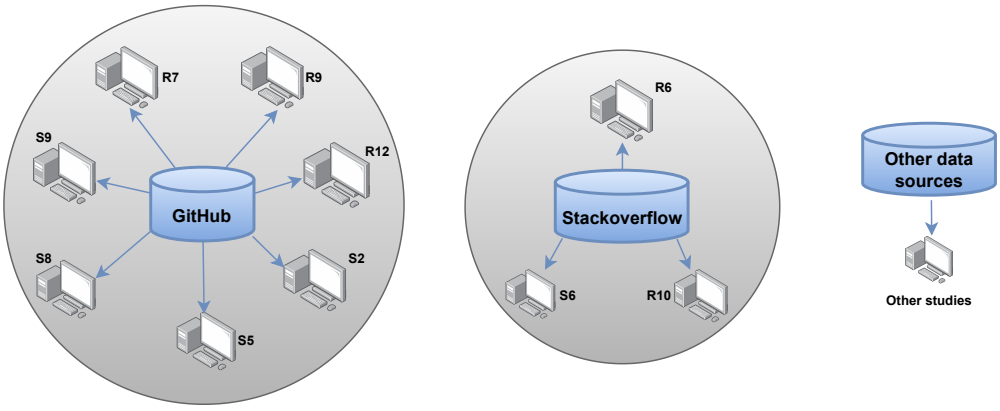
and recommending those with the highest possibility of success. This issue highlights the importance of tailoring recommendations to specific contexts and objectives. Paper R9 (Table 2) introduces COOPFINDER, which suggests collaborators based on co-changed files, emphasizing the relevance of shared code modifications as an indicator of collaboration potential.

Furthermore, several papers discuss expertise identification as a crucial aspect of collaborator recommendation. Approaches such as those presented in Paper R2, R11, S9, and S6 (Table 2 and Table 3) aim to identify and rank experts based on their knowledge, skills, and contributions in specific domains or communities. These methods often rely on analyzing coding activities, commit histories, and other artifacts to determine expertise levels and recommend suitable collaborators accordingly.

3.6.2 Data sources

Several patterns and trends can be identified regarding the data sources used for recommendation approaches in the context of aligning the technical skills of software developers with their industry domain experience. Figure 3 shows the representation of the data sources used on the projects. GitHub was the data source of 6 studies, StackOverflow was the data source of 3 papers the other studies used different fonts.

– Figure 3 - Data sources frequency representation



Firstly, it is evident that GitHub repositories play a significant role as a data source in several papers. Papers such as R7, R9, R12, S2, S5, S8, and S9 (Table 2 and Table 3) explicitly mention using GitHub repositories to gather information for the recommendation approach. This indicates that analyzing developers' activities, commits, and projects on GitHub can provide valuable insights into their expertise and skills.

StackOverflow is another prominent data source mentioned in the papers. Papers R16, R10, and S6 (Table 2 and Table 3) make use of StackOverflow forum data to identify experts or retrieve information about developers' expertise. StackOverflow, a popular

platform for knowledge sharing and QA among developers, is a valuable resource for understanding their technical skills and domain knowledge.

Additionally, some papers focus on specific online communities or platforms. For instance, Paper R2 (Table 2) collects data from Sun forums (now Oracle forums) and Apple Discussions, while Paper R5 concentrates on the Launchpad collaborative software development platform. These papers highlight the importance of considering data sources specific to particular communities or platforms when making recommendations.

Moreover, a few papers utilize specialized datasets or corpora. Paper R1 (Table 2) uses Topcoder data between 2009 and 2018 for developer recommendation, while Paper R3 (Table 2) mentions utilizing the DBLP dataset, MovieLens dataset, and NBA/ABA statistics. These papers indicate that domain-specific datasets can be valuable in determining the expertise and skills of software developers.

Furthermore, some papers involve mining job advertisements and developer profiles. Paper S1 (Table 3) mentions crawling job advertisements from the UK job portal Indeed, while extracting developer profiles from the GHTorrent dataset. This approach suggests that analyzing job requirements and developers' profiles can provide insights into their technical skills and align them with industry domains.

Regarding similarities, GitHub repositories are frequently mentioned as a common data source across multiple papers, indicating their significance in identifying developers' skills. StackOverflow is another widely used data source, highlighting its importance as a platform for acquiring expertise-related information. Additionally, the utilization of specialized datasets, online communities, and job-related data also appears in several papers, demonstrating their relevance in making accurate recommendations.

To align the technical skills of software developers with their industry domain experience, the most relevant mentioned sources include GitHub repositories and StackOverflow. Analyzing developers' activities, commits, and projects on GitHub can provide insights into their technical proficiency, while StackOverflow data can reveal their expertise in specific programming languages and domains. Additionally, considering specialized datasets, online communities, and job-related data can further enhance the accuracy of the recommendation approach by incorporating domain-specific knowledge and requirements.

In summary, the analysis of the provided scientific papers indicates that GitHub repositories, StackOverflow, specialized datasets, online communities, and job-related data are valuable sources for developing a recommendation approach that aligns the technical skills of software developers with their industry domain experience.

3.6.3 Technical skills and industry domain alignment

To answer the question "Could the described approach be used to recommend collaborators aligning desired technical skills to industry domains?", the authors undertook

a meticulous analysis of each paper. This involved dissecting the methodologies outlined in the papers, and the underlying datasets employed, and subsequently categorizing the response into three distinct possibilities: "Yes,No,"and "Partially.".

Following this procedure, it was possible to obtain a broader overview of the state of the art in terms of the proposed alignment. The majority of papers (15 out of 21) have been answered with "No,"indicating that the described approach in these papers is not suitable for recommending collaborators from different industry domains. This suggests that the proposed methods may have limitations or are not directly applicable to this particular use case. Only six were classified with "Partially", suggesting that some aspects of the described approach could be used to recommend collaborators across industry domains, but the methods might require modifications or additional considerations to be effective. No paper was classified with "Yes", indicating that none of the papers directly present an approach that is fully suitable for recommending collaborators from different industry domains. This might be due to the complexity and specificity of this task.

While none of the papers received a "Yes"answer, the fact that several papers received "Partially"answers indicates that there might be potential for future research and development in this area. Researchers could potentially build upon the existing approaches to create more effective methods for recommending collaborators from different industry domains.

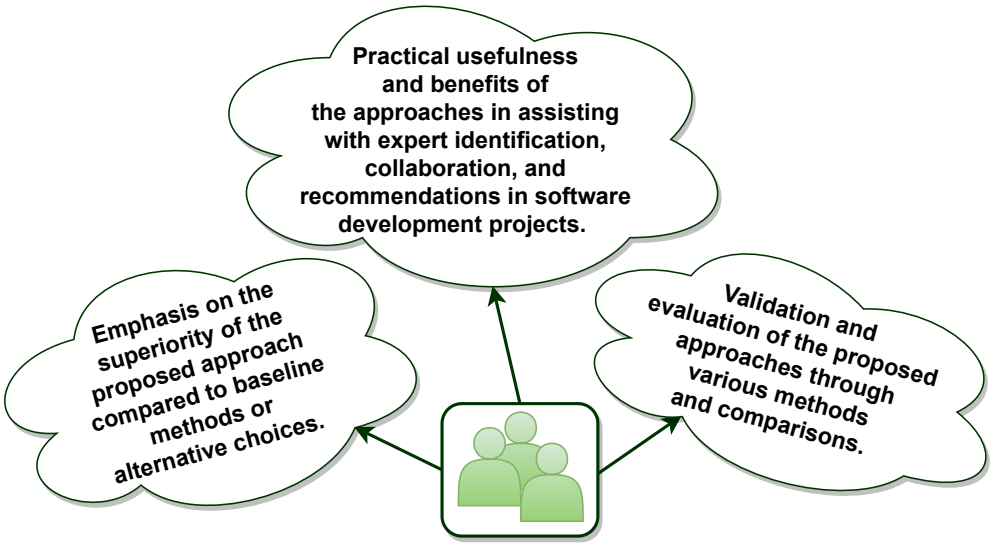
In conclusion, the dataset suggests that while none of the papers directly provide a solution for recommending collaborators from different industry domains, there are some papers that offer elements that could be adapted or combined to create effective methods. The prevalence of "No"answers highlights the challenges in finding a straightforward solution for this complex task. Researchers could focus on leveraging the partially applicable methods and exploring ways to enhance their effectiveness for cross-domain collaboration recommendations.

3.6.4 Authors conclusions

The conclusions of the analyzed papers primarily focus on evaluating the effectiveness, efficiency, and performance of the proposed methods in expert finding, recommendation, and expertise identification within software development contexts.

Among the similarities (Figure 4) in the authors' conclusions, one prominent similarity is the emphasis on the superiority of the proposed approach compared to baseline methods or choices. For instance, paper R10 (Table 2) highlights that their deep learning-based method outperforms the best baseline method by significant margins in terms of F-measure on datasets C and Java. Similarly, paper R3 (Table 2) concludes that their method, which combines skill matching and structural matching, significantly outperforms choices in terms of both average precision and recall.

– Figure 4 - Conclusions similarities



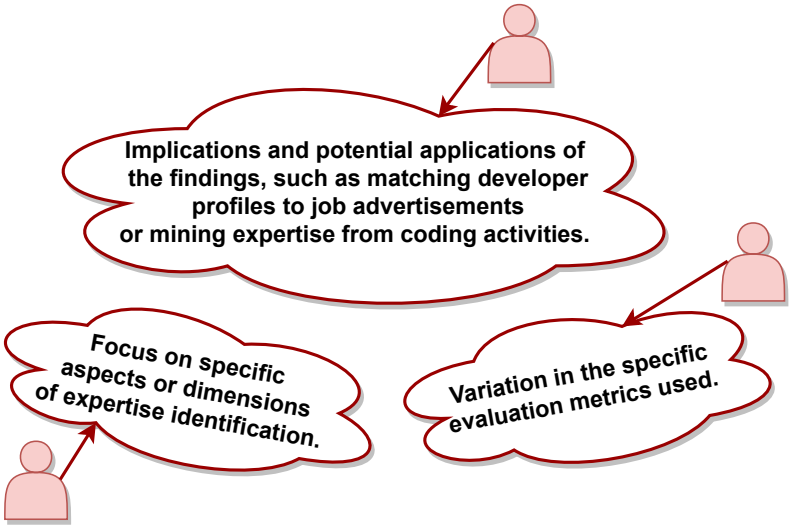
Another recurring theme is the validation and evaluation of the proposed approaches. Many papers, such as Paper R8 (Table 2) and Paper S8 (Table 3), emphasize the validation and efficiency of their approaches through stress testing, accuracy checks, and comparisons with existing methods. The authors often highlight the practical usefulness and benefits of their approaches in assisting with expert identification, collaboration, and recommendation in software development projects.

Regarding differences (Figure 5), some papers focus on specific aspects or dimensions of expertise identification. Paper R2 (Table 2), for example, compares expertise profiling methods and highlights the appropriateness of the Latent Dirichlet Allocation (LDA) model, particularly when combined with specific filtering strategies. Paper R4 distinguishes its approach by considering three dimensions: individuals, expertise areas, and social dimensions, which sets it apart from other studies.

Several papers also discuss the implications and potential applications of their findings. For instance, Paper S1 (Table 3) concludes that their approach of matching developer profiles to job advertisements shows promise and provides interesting insights into the overlap of concepts between adverts and README files. Paper S9 (Table 3) indicates that their approach of mining expertise from coding activities can detect higher-level expertise in commit histories and describe skills in programming languages, libraries, and higher-level concepts.

Among the most relevant conclusions from the provided papers, several stand

– Figure 5 - Conclusions differences



out. Paper R10's deep learning-based method achieving superior performance on different datasets is noteworthy. Paper R3's combination (Table 2) of skill matching and structural matching resulting in improved precision and recall is another significant finding. Additionally, Paper S5's proposition (Table 3) of using clustering techniques to identify experts in specific libraries and frameworks shows promise, as it outperforms standard machine learning classifiers.

The analyzed papers present various approaches to expert finding, recommendation, and expertise identification in software development. The conclusions highlight the superiority of the proposed methods, the validation and evaluation of the approaches, and the potential applications of the findings. The most relevant conclusions include the superior performance of deep learning-based methods, the effectiveness of combining skill matching and structural matching, and the use of clustering techniques for identifying experts in specific contexts.

The conclusions from the analyzed papers provide valuable insights for companies looking to find software developers with expertise in specific industry areas. These conclusions highlight the effectiveness and practical benefits of various expertise identification methods, showing they outperform basic approaches. The validation and evaluation of these methods give confidence in their reliability, and insights into specific expertise dimensions and potential applications help tailor strategies for talent acquisition and team building. Key findings, like the success of deep learning methods and combining matching techniques, offer practical tools for decision-making and competitive advantage in the software development field.

3.6.5 Answering the Research Question

This subsection answers the RQ (*"How is the alignment of industry domains and technical expertise used in systems that recommend software developers?"*) through an analysis of the sub-questions presented.

The alignment of industry domains and technical expertise within recommendation approaches is explored through an in-depth analysis of various research findings. Authors of the surveyed papers reveal two predominant trends in collaborator recommendation.

Firstly, machine learning models leveraging diverse data sources such as social networks, coding repositories, and platforms like StackOverflow are prevalent. These trends underscore the significance of advanced algorithms and varied data to enhance recommendation accuracy.

Secondly, contextualized recommendations are prominent, exemplified by papers like R8 and R1 (Table 2), which tailor suggestions to specific tasks, challenges, and success probabilities. The introduction of COOPFINDER in Paper R9 (Table 2) emphasizes shared code changes as valuable collaboration indicators. Expertise identification also plays a pivotal role, evident in Papers R2, R11, S9, and S6 (Table 2 and Table 3), where methods gauge domain-specific skills through coding activities and commit histories.

In essence, these trends reflect ongoing efforts to refine collaborator recommendations, considering intricate collaboration contexts, diverse data, and expertise-driven evaluations. Additionally, the pivotal role of GitHub repositories and StackOverflow as primary data sources underlines their importance in aligning technical skills with industry domains.

This combined with supplementary data forms a robust foundation for effective alignment. While challenges persist, the study suggests the potential for future research to enhance cross-domain collaboration recommendation approaches, encapsulating both complexities and possibilities in this domain.

Therefore, in short, an answer to the RQ of this systematic mapping could be: Recommendation alignment of industry domains and expertise leverages machine learning, contextualization, and data from sources like GitHub and StackOverflow for enhanced accuracy and relevance.

3.7 Insights

This section aims to facilitate meaningful discussions with researchers seeking to contribute innovative and relevant solutions in this research area. To guide our discussion, we address two key questions that provide valuable insights and help steer the direction of further investigations:

- What is the future work mentioned by the authors?
- What should a new software developer recommendation system that aims to align desired technologies to the industry domains take into consideration?

In addition to addressing the general needs of collaborative software development, it's imperative to underscore the significance of aligning the recommendation system with the unique requirements of established industries such as finance, automotive, industrial systems, and healthcare. These sectors demand specialized expertise and stringent regulatory compliance, making tailored recommendations essential for successful project outcomes. By incorporating domain-specific customization into the recommendation system, it can effectively cater to the distinct challenges and nuances of each industry, ensuring that recommended developers possess the requisite skills and knowledge pertinent to their respective domains. For instance, in finance, expertise in risk management and regulatory compliance may be paramount, while the automotive industry may prioritize developers with experience in embedded systems and automotive software standards. Similarly, healthcare projects necessitate developers well-versed in healthcare regulations and data privacy protocols. By explicitly addressing the needs of these industries, the proposed recommendation system not only enhances collaboration but also adds practical value by facilitating the creation of high-quality software solutions tailored to industry-specific requirements.

3.7.1 Mentioned future work

After reviewing the future work outlined by the authors, several relevant approaches and technologies emerge and gathering them into a concrete list could ease and shed some light on the path ahead to the research and industry communities. In Table 5, the main future work and the papers that mention them are summarized, nonetheless, we discuss each future work in detail.

Table 5 – Summary of the main future work and their papers

Future Work	**Papers**
Leveraging Network Analysis and Social Interactions	R10, R12
Incorporating Contextual Information and Domain-Specific Knowledge	R5, S7
Exploring Advanced Techniques and Technologies	S3, S9
Continuous Improvement and Expansion	R8, R4

Firstly, leveraging network analysis and social interactions among candidates is mentioned in several papers. The authors of Paper R10 (Table 2) propose using

the questions and answers exchanged among candidates on StackOverflow to create a network of candidates. This approach can provide valuable insights into the expertise and knowledge of individuals, allowing for a better estimation of candidates' suitability for agile teams. Similarly, Paper R12 (Table 2) discusses enhancing the ontology and incorporating syntactic and semantic social network analysis to recommend specialists and teams with complementary skills in Global Software Development (GSD) contexts. These approaches highlight the importance of leveraging collaborative platforms and social interactions to improve the accuracy of expert findings.

Secondly, there is a focus on incorporating contextual information and domain-specific knowledge. Paper R5 (Table 2) mentions considering project domain information to improve the reputation scores of contributors. Recognizing that reputation and expertise can vary across different domains emphasizes the need for context-dependent evaluation and systems that recommend software developers. Additionally, Paper S7 (Table 3) mentions the need to define patterns that specify the domain expertise of programmers. By capturing and utilizing domain-specific knowledge, future systems can provide more accurate and tailored recommendations based on the specific requirements and challenges of different software projects.

Thirdly, there is a push to explore advanced techniques and technologies to enhance recommendation approaches. Deep learning approaches, such as convolution neural networks (CNNs), are mentioned in Paper S3 (Table 3) for mining the shape of expertise. The authors suggest that CNNs can extract meaningful patterns and features from expert profiles, enabling more accurate expert grouping. Furthermore, Paper S9 (Table 3) proposes improving the quality of indicator words by applying different selection mechanisms and conducting user studies to evaluate the effectiveness of the system. These advancements in machine learning and natural language processing techniques can significantly improve the recommendation accuracy and overall performance of expert-finding systems.

Moreover, the papers emphasize the need for continuous improvement and expansion of the systems that recommend software developers. Several papers, including Paper R8 and Paper R4 (Table 2), discuss the incorporation of new metrics and the integration of additional tools for more seamless collaboration. The authors highlight the importance of identifying and incorporating further new metrics to quantify team characteristics, as well as integrating third-party tools to harness detailed digital footprints. This ongoing evolution of recommendation approaches ensures that they remain relevant and effective in the dynamic landscape of collaborative software development.

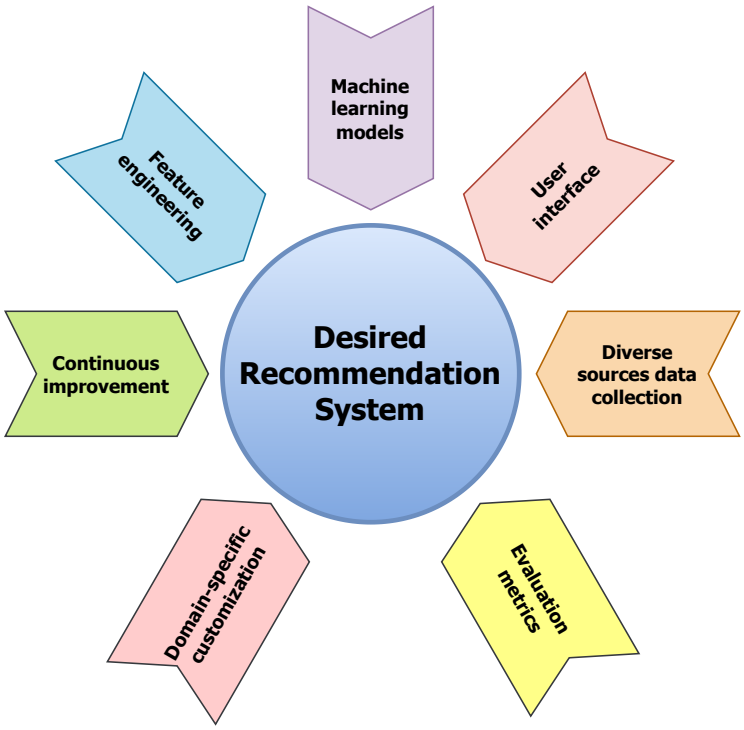
In summary, the future work mentioned in the analyzed papers indicates that the field of expert finding, team recommendation, and collaborative software development is moving towards leveraging social interactions, incorporating contextual information, exploring advanced techniques like deep learning, and continuously improving and expanding

the systems that recommend software developers. By embracing these approaches and technologies, future systems can provide more accurate and tailored recommendations, enabling effective collaboration and enhancing the productivity and success of software development projects.

3.7.2 A new systems that recommend software developers should consider

As you embark on the design of a groundbreaking recommendation system geared toward aligning software developers with precise technical expertise relevant to specific industry domains, it's imperative to ensure that all requirements are identified and met. To attain this, delving into the state-of-the-art within this domain is a prudent initial step. By conducting this systematic requirements analysis, we have effectively compiled a comprehensive list of requirements that warrant consideration. Therefore, a recommendation system with the overarching goal of facilitating this matchmaking process must give utmost importance to the following seven key aspects [4]: 1. Data Collection and Integration, 2. Feature Engineering, 3. Machine Learning Models, 4. Evaluation and Validation, 5. Domain-Specific Customization, 6. User Interface and Interaction, 7. Continuous Improvement. (Figure 6).

– Figure 6 - The desired composition of the recommendation system



[4] The terms that comprise each priority are related to the studies so that readers can easily refer to them in the texts.

Priority 1: Data Collection and Integration Begin by gathering relevant data from diverse sources, including social-coding repositories, collaborative platforms, coding activities, and developer profiles. Thoroughly integrate and preprocess the collected data to construct a comprehensive dataset that encompasses the necessary information for generating meaningful recommendations.

Data collection and integration are foundational for the recommendation system. Priority should be given to gathering data from various sources, including developer profiles, project repositories, industry-specific knowledge bases, and user interactions. Efficient data integration allows an understanding of developers' skills and expertise.

Priority 2: Machine Learning Models Machine learning models play a central role in analyzing and matching developers' skills with industry-specific domains. Prioritizing the development of accurate models is crucial for making relevant recommendations. These models should consider both user behavior and content-based recommendation approaches.

Employ state-of-the-art machine learning techniques, such as neural networks, graph convolution networks, or latent Dirichlet allocation, to construct models capable of learning from the collected data. These models should be adept at handling the intricate complexities inherent in the software development context and yield accurate recommendations based on the extracted features.

Priority 3: Feature Engineering Effective feature engineering is essential for extracting relevant information from the data. Prioritize the creation of meaningful features that capture developers' skills, project requirements, and industry-specific knowledge. These features enhance the accuracy of recommendations.

Extract meaningful features from the collected data, taking into account factors such as expertise, coding patterns, collaboration history, and social network analysis. These features should capture both the technical skills possessed by the developers and their collaborative characteristics, thereby enabling a holistic representation of their capabilities.

Priority 4: User Interface and Interaction A user-friendly interface is important for user engagement. The system should provide an intuitive and responsive user interface that allows developers and industry stakeholders to input requirements, explore recommendations, and provide feedback. Prioritize usability, visual design, and responsiveness.

Devote attention to the development of a user-friendly interface that seamlessly facilitates user interaction with the recommendation system. Provide intuitive visualizations and explanations to enhance transparency and assist users in comprehending the rationale behind the recommendations presented to them. A well-designed user interface enhances user engagement and promotes the adoption of the system.

Priority 5: Continuous Improvement Continuous improvement is crucial for the system's long-term success. Prioritize the implementation of mechanisms for collecting user feedback, monitoring system performance, and making iterative enhancements. Regular updates and refinements keep the recommendation system relevant and valuable to users.

Implement mechanisms to gather user feedback and establish an iterative process for improving the recommendation system over time. Embrace an agile development approach that allows regular updates and enhancements based on evolving user needs and collaboration dynamics. By actively seeking and incorporating user feedback, the system can be refined and further optimized to meet the evolving requirements of software development teams.

Priority 6: Evaluation and Validation Evaluation and validation are necessary to assess the system's effectiveness and reliability. Prioritize the establishment of robust evaluation metrics and validation methods to measure the quality of recommendations. This includes precision, recall, user satisfaction, and algorithm performance.

Establish appropriate evaluation metrics to rigorously assess the performance of the recommendation system. Utilize robust validation techniques, including cross-validation or A/B testing, to validate the effectiveness and robustness of the employed machine learning models. These techniques ensure that the recommendations generated meet the desired quality standards.

Priority 7: Domain-Specific Customization While important, domain-specific customization can be addressed once the foundational components are in place. Customization allows the system to adapt to specific industry needs and requirements. It should be considered after the core functionality is established.

Tailor the recommendation system to the specific context and domain of software development. Take into consideration the unique challenges, collaboration patterns, and expertise requirements prevalent in the target domain, in order to provide relevant and effective recommendations aligned with the specific industry contexts.

By adhering to these recommendations and considering the current trends found in the existing literature, a new project could create a recommendation system that effectively suggests suitable collaborators in software development. This system would help foster smooth collaboration and improve team performance. Emphasizing critical aspects such as data quality, feature richness, appropriate machine learning model selection, rigorous evaluation, domain-specific customization, user experience, and continuous improvement contribute to the development of a robust and valuable recommendation system.

3.7.3 Insight conclusions

This priority list emphasizes the importance of building a strong foundation with data collection, machine learning models, and feature engineering. These components are critical for the accuracy of recommendations. User interface and interaction are also essential for user adoption. Non-functional requirements like continuous improvement and evaluation ensure the system's long-term success and quality.

While the trend of AI and deep learning may seem ubiquitous in today's technological landscape, it is a trend that cannot be overlooked or ignored. These advanced techniques offer transformative capabilities that have the potential to revolutionize software developer recommendation approaches. Ignoring this trend risks falling behind competitors and missing out on opportunities to significantly improve recommendation algorithms' accuracy, efficiency, and effectiveness. Embracing AI and deep learning allows for a more nuanced analysis of developer expertise and project requirements. It opens doors to innovative approaches that can better cater to the diverse needs of collaborative software development across various industries. Therefore, while acknowledging the prevalence of this trend, researchers and practitioners alike must embrace it wholeheartedly and explore its full potential in advancing recommendation system technologies.

Domain-specific customization is placed lower in priority because it typically involves tailoring the system to specific industries, which can be addressed as the system matures.

As a way of directly answering the questions proposed at the beginning of this section, we can conclude that an answer for the first Insight question (*What is the future work mentioned by the authors?*), could be: Authors propose future work in expert finding and team recommendations, emphasizing data source expansion, context integration, advanced techniques like CNNs, and ongoing system improvement with new metrics and tools for seamless collaboration, ensuring effectiveness in evolving collaborative software development.

Therefore, a resumed answer to the second insight question (*What should a new software developer recommendation system that aims to align desired technologies to the industry domains take into consideration?*), could be: A novel software developer recommendation system, aiming to align technologies with industry domains, must consider data collection and integration, feature engineering encompassing expertise and collaboration aspects, state-of-the-art machine learning models, rigorous evaluation, domain-specific customization, user-friendly interface design, and continuous improvement through user feedback. This holistic approach ensures meaningful recommendations, tailored to the software development context, fostering seamless collaboration, and enhancing team performance through an agile and evolving system.

In conclusion, the insights gained from exploring future work in expert finding and

team recommendations, along with recognizing the significance of AI and deep learning trends, contribute to advancing software developer recommendation approaches. By prioritizing aspects like social interactions, contextual information, and user-friendly design, researchers and practitioners can develop more effective systems tailored to collaborative software development needs. These insights provide a roadmap for addressing research questions and objectives, fostering the creation of systems that recommend software developers and that facilitate smooth collaboration and improve team performance in software development projects.

3.8 Threats to validity

This systematic literature mapping was conducted to comprehensively identify, categorize, and analyze systems that recommend software developers for software developers in the global software development domain. However, like any research method, there are certain potential threats to its validity and limitations that need to be acknowledged.

One potential limitation is associated with the possibility of errors in the protocol definition and the search string employed, which may have inadvertently excluded relevant keywords and studies, thereby potentially omitting valuable insights. To mitigate this limitation, the mapping planning presented in Section 3 underwent rigorous review and scrutiny by other experienced researchers to enhance its comprehensiveness and minimize potential oversights.

Beyond that, the inherent limitations imposed by the nature of Systematic Mapping Studies (SMS) also represent a threat to the validity of the study. Specifically, the involvement of multiple companies or entities with distinct hard boundaries presents a challenge. As an SMS, we do not exercise control over the selection of companies participating in the studies, nor do we influence the delineation of their boundaries. Consequently, the applicability of data source mining techniques discussed in this paper may be constrained in such contexts. While efforts were made to mitigate this limitation through comprehensive analysis, the variability introduced by diverse company structures remains a pertinent consideration.

Another potential limitation pertains to selection bias. The inclusion and exclusion criteria used to select studies could introduce bias if they are not clearly defined or consistently applied. To address this potential bias, a meticulous peer review of the accepted and rejected studies was conducted, ensuring that the criteria were rigorously applied and adhered to.

Furthermore, there is the inherent risk of data extraction bias. The process of extracting data from the included studies can be subjective and susceptible to bias if it is not standardized or if different researchers interpret the data differently. To mitigate this bias, a structured form was developed, encompassing predefined data fields that need to be

consistently extracted from each included study. This approach helps minimize subjectivity and ensures a more objective data extraction process across different researchers involved in the study.

By acknowledging and addressing these potential limitations, we strive to enhance the validity and reliability of our systematic literature mapping, providing a more robust foundation for the insights and conclusions drawn from this research.

4 DevFinder architecture

Based on the results presented in the systematic mapping described in the last chapter, gaps and opportunities were identified. Firstly, it is noted the absence of studies that directly propose architectures to recommend software developers specialists whose experience of working on specific industry domains aligns with knowledge of specific technologies. Secondly, the answer to the RQ of the systematic mapping highlights the usage of machine learning, contextualization, and data from sources such as GitHub for enhanced accuracy. This piece of information underscores the relevance of these approaches in the construction of the desired architecture to recommend software developer specialists who align knowledge in specific industry domains to desired technologies. Thirdly, the authors' conclusions and future work point out leveraging network analysis, incorporating contextual information, and exploring advanced techniques to better support decision-makers on the task of finding software developers. Lastly, the insights gathered serve as seeds for the elicitation of requirements for the features and characteristics that an architecture of this nature would have to perform.

Therefore, the requirements, advice, and ideas gathered on the systematic mapping were used as the bedrock for the development of the DevFinder architecture. To develop this architecture, the Design Science Research (DSR) methodology was used, and three full iterations of the DSR cycle were performed to guarantee the quality and trustworthiness of the solution.

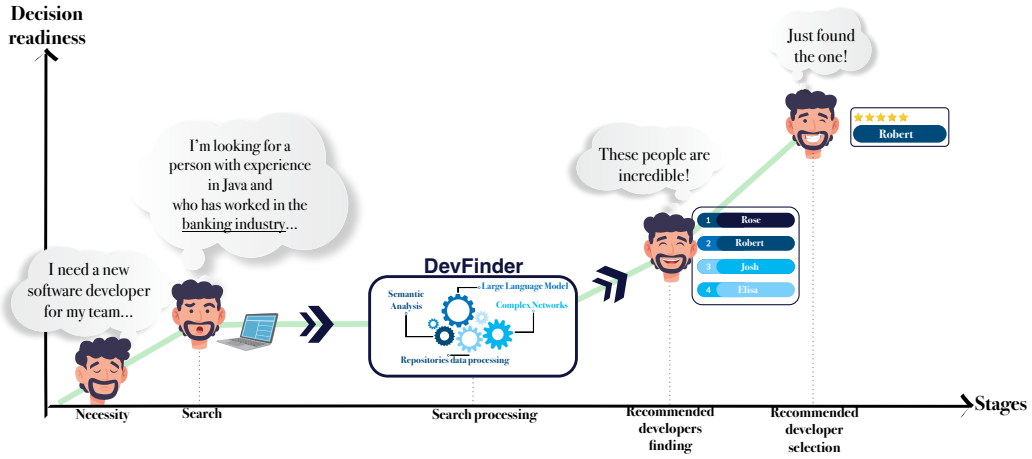
As an illustration for the employment scenario of DevFinder, consider that John serves as the hiring manager for a software company that recently entered into a contract with a bank to enhance a legacy application. Due to the current capacity of the development team and the specificity of the project from the client, John is actively seeking a developer with hands-on experience in the banking industry and proficiency in Java, which serves as the primary language for the application (Figure 7).

To find matching candidates, John turns to DevFinder. Inputting the specific industry and technology criteria, he initiates the search, prompting DevFinder to employ techniques such as data mining, complex network analysis, large language models, and semantic analysis. This approach allows DevFinder to identify and rank software developers from a version control system based on contribution factors, also providing insights to assist John in evaluating each candidate.

The outcome is a ranked list of software developers proficient in Java who have worked on banking-related projects. By clicking on each developer, John can read key information and insights generated by large language models, summarizing the developer's skills based on the criteria he input.

Equipped with this information, John's decision readiness is higher, and he now feels

– Figure 7 - John's journey on DevFinder



better prepared to make an informed decision, armed with a comprehensive understanding of each developer's qualifications and how they align with the project's specific requirements, serving as an illustrative example of a use-case scenario in DevFinder. The processes for searching, ranking, and generating insights are further discussed in the next sections.

4.1 Methodology

Design Science Research (DSR) stands as a methodological approach driven by the continual enhancement of a solution through the introduction of novel artifacts and the corresponding construction processes (114). The application domain, comprising individuals, organizations, and technological systems working towards a common goal, serves as the backdrop for DSR research. Typically, the research journey begins by identifying and portraying opportunities and challenges within a real-world application environment.

The initiation of the relevance cycle marks the commencement of the research, with the application context not only providing the research requirements—such as the identified opportunity or problem—as inputs but also establishing acceptance criteria for the final assessment of research outcomes. The research output is then reintegrated into the environment for thorough examination and evaluation within the application domain. The examination of the artifact in the domain can be carried out through various technology transfer methods, such as applied research (115, 116).

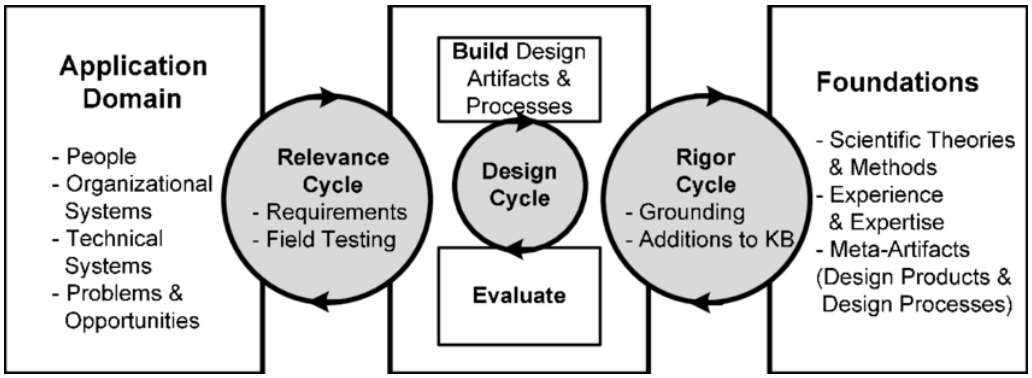
The outcomes of the domain test determine whether additional iterations of the relevance cycle are necessary for the ongoing Design Science project. The newly deve-

loped artifact may require additional functionalities or inherent qualities, like improved performance or usability, to enhance its practical utility. The resulting artifact may unveil the need for new requirements or expose misconceptions or incompleteness in the existing requirements. Subsequently, another iteration of the relevance cycle commences, incorporating feedback from the research environment and redefining research requirements based on experiential insights.

Throughout the relevance cycle, there is a continual refinement of the requirements to align the artifact with its intended goal. Additionally, a comprehensive analysis of the application context, involving the people and organizational systems, is essential to ensure the congruence of requirements with the identified problem. The design cycle encompasses the construction of the artifact, involving its implementation and development processes, culminating in an evaluation. This process can yield a tangible product, an improved process, or contribute to the body of scientific knowledge.

The rigor cycle is where the theoretical underpinning comes into play, guiding the construction of the artifact using methods, theories, or processes available in the literature. In this cycle, the validity of initial theoretical assumptions is scrutinized through the evaluation of the artifact. A holistic view of Design Science Research involves three interconnected cycles: relevance, design, and rigor, as depicted in Figure 8. Each cycle plays a crucial role in the systematic and iterative development of artifacts and the advancement of knowledge within the Design Science paradigm.

– Figure 8 - DSR Flow



Source: (117)

This research follows DSR methodology by employing cycles of development and undergoing DSR thoroughness on each cycle iteration.

4.2 Requirements

For the system specification, functional and non-functional requirements were prioritized for its development taking into account the key factors found in the systematic

mapping presented in the last chapter. Therefore, the final implementation of DevFinder architecture targets the support of job recruiters stakeholders on the search for software specialists through the functional and non-functional requirements that follow.

4.2.1 Functional Requirements

- FR 001. The architecture must support the data processing from domain-specific combinations of industries and technologies specified by the user.
- FR 002. The architecture must support the data extraction from real-world repositories on version control systems.
- FR 003. The architecture must support the repositories filtering that are not related to the specified industry domain.
- FR 004. The architecture must support the software developers ranking based on collaboration metrics.
- FR 005. The architecture must support the provision of insights about each ranked software developer generated through the available data of software developers.
- FR 006. The architecture must support the provision of a final recommendation list so that user interfaces can be integrated into it.

4.2.2 Non-functional Requirements

- NFR 001 (Dependability). The architecture should facilitate interaction with external data sources.
- NFR 002 (Extensibility). The architecture design should adhere to extensibility principles to accommodate the potential expansion.
- NFR 003 (Scalability). The architecture must respect scalability principles. This involves the ability to integrate new data sources and perform intelligent processing dynamically.

4.3 Related Work

Previous studies have been published regarding previous DSR iteration cycles (136, 137). On the first DSR iteration cycle (136), the study aimed to streamline the process of identifying experts with specific technical skills and industry knowledge in software development by integrating data from various platforms such as LinkedIn, GitHub, and Topcoder. Utilizing semantic and syntactic techniques along with an ontology, the approach

matched data and inferred non-obvious information to compile a list of recommended experts.

On the second DSR iteration cycle (137), the study aimed to address the challenge of finding software development experts who possess both specific technical skills and experience in particular industry domains, a crucial requirement in software development environments. To tackle this issue, we proposed a recommendation approach that identifies and categorizes eligible experts by extracting data from repository databases and constructing a complex network that considers three key aspects: the impact of a developer on industry-related repositories, the technologies relevant to the search terms, and the timeline of the developer's contributions to projects.

Other approaches have been developed to aid stakeholders in decision-making processes. Focusing on the current DSR iteration of DevFinder architecture, which incorporates LLM modules, this section presents research related to this paper, highlighting the employment of LLM on systems that recommend software developers. In the end, we make a comparative analysis among them, bringing up the contribution of this paper.

As the first related work, Hua (2023) explore the advancements by Foundation Models, particularly Large Language Models (LLMs), in recommender systems. This tutorial not only introduces the concept of Foundation Models like LLMs for recommendation but also delves into the evolution of recommender systems from shallow to deep models and ultimately to large models. Additionally, it discusses how LLMs enable generative recommendation as opposed to the traditional discriminative recommendation and provides insights into building LLM-based recommender systems from multiple perspectives, encompassing data preparation, model design, pre-training, fine-tuning, multi-modality, multi-task learning, fairness, and transparency (151).

Zhang (2023) presents a novel approach to developing recommendation models by leveraging Large Language Models for instruction-based recommendation systems. Rather than solely relying on historical behavior data, the researchers propose using natural language instructions to convey user preferences and needs to LLMs, allowing them to generate recommendations accordingly. They design a general instruction format and create a substantial amount of user-personalized instruction data. Through experiments on various recommendation tasks using real-world datasets, the proposed approach demonstrates superior performance compared to competitive baselines, including GPT-3.5. This study showcases the potential of integrating natural language instructions into recommender systems, enabling more user-friendly interactions and providing more accurate recommendations (160).

Fan (2023) addresses the limitations faced by Deep Neural Networks (DNNs) in recommender systems, such as understanding the interests of users, capturing textual side information, generalizing to different recommendation scenarios, and reasoning on

predictions. It highlights the potential of Large Language Models (LLMs) like ChatGPT and GPT4, aiming to provide a systematic overview of LLM-empowered recommender systems, covering aspects like Pre-training, Fine-tuning, and Prompting. The paper introduces methods utilizing LLMs as feature encoders for learning user and item representations, reviews recent techniques of LLMs for enhancing recommender systems through pre-training, fine-tuning, and prompting, and discusses future research directions in this emerging field (8).

A study conducted by Di Palma (2023) explores the integration of Large Language Models (LLMs) into Recommender Systems (RSs), leading to the development of a novel approach termed Retrieval-augmented Recommender Systems. Traditionally, RSs excel in offering personalized recommendations within well-defined domains but struggle with adaptability to novel data. By combining these technologies, the study aims to enhance RSs' ability to provide relevant recommendations even in scenarios with sparse data. This fusion of retrieval-based and generation-based models presents a promising solution for delivering contextual and personalized suggestions across various domains, ranging from e-commerce to content streaming platforms (152).

Liu (2023) investigates the application of ChatGPT in the domain of recommendation systems. The researchers explore ChatGPT's potential as a general-purpose recommendation model by leveraging its extensive linguistic and world knowledge acquired from large-scale corpora. They design a set of prompts and evaluate ChatGPT's performance across five recommendation scenarios without fine-tuning the model during the evaluation process, relying solely on the prompts for task conversion. Additionally, they explore the use of few-shot prompting to incorporate user interaction information, enhancing ChatGPT's understanding of user needs and interests. Experimental results on the Amazon Beauty dataset demonstrate promising performance across various tasks, with human evaluations confirming the model's capability to generate clear and reasonable recommendations. The study underscores the potential of language models like ChatGPT to enhance recommendation system performance, urging further exploration in this field (9).

Fan (2020) introduces a novel framework called GraphRec+ for social recommendations, leveraging Graph Neural Networks (GNNs) to effectively model graph data inherent in social networks and user-item relationships. By addressing challenges such as the simultaneous involvement of users and items in various graphs, integration of user opinions in user-item interactions, and the heterogeneous nature of social relations, GraphRec+ offers a coherent solution for learning user and item representations. The framework introduces a principled approach to jointly capture interactions and opinions in the user-item graph, along with an attention mechanism to differentiate heterogeneous strengths of social relations. Through comprehensive experiments on real-world datasets, the efficacy of GraphRec+ is demonstrated, highlighting its potential to advance social recommendation

Table 6 – Related work summary

Study	Authors	Focus	Approach/Technique	Key Findings
DevFinder (1st iteration)	de Campos et al.	Identifying experts with specific skills and knowledge in software development	Integration of data from LinkedIn, GitHub, Topcoder; use of semantic and syntactic techniques, ontology	Successful in matching data and inferring non-obvious information to recommend experts
DevFinder (2nd iteration)	de Campos et al.	Finding software development experts with specific skills and industry experience	Recommendation system extracting data from repositories; complex network analysis	Effective in identifying experts considering repository impact, relevant technologies, and contribution timelines
LLM in Recommender Systems	Hua et al.	Advancements in recommender systems using Large Language Models (LLMs)	Overview of LLM applications in recommendation, covering data prep, model design, pre-training, fine-tuning, and fairness	LLMs enable generative recommendations and provide insights into building sophisticated recommender systems
Instruction-based Rec. Models	Zhang et al.	Developing recommender systems using natural language instructions	Leveraging LLMs for user preferences via instructions; creation of user-personalized instruction data	Demonstrated superior performance in recommendation tasks compared to traditional baselines
LLM Empowered Rec. Systems	Fan et al.	Enhancing recommender systems with Large Language Models	Systematic overview of LLMs in recommendations, including pre-training, fine-tuning, and prompting	LLMs address limitations in DNN-based recommenders, improve feature encoding and scenario adaptability
Retrieval-Augmented RS	Di Palma	Combining LLMs with recommender systems for better adaptability	Fusion of retrieval-based and generation-based models	Improved recommendations in scenarios with sparse data, enhancing contextual and personalized suggestions across domains
ChatGPT for Rec. Systems	Liu et al.	Applying ChatGPT to recommendation tasks without model fine-tuning	Use of prompts and few-shot prompting for task conversion and user interaction integration	ChatGPT showed promising performance in various recommendation tasks, with human evaluations supporting the results
GraphRec+ for Social Rec.	Fan et al.	Social recommendations using Graph Neural Networks (GNNs)	Modeling user-item relationships and social networks, using attention mechanisms for heterogeneous relations	Demonstrated effectiveness in capturing interactions and opinions in social recommendation tasks

systems (21).

4.3.1 Comparative Analysis

As also depicted in Table 6, considering the employment of LLM on systems that recommend software developers, we can summarize the related work as: Hua (2023) and Zhang (2023) focus on LLMs, emphasizing their role in the generative recommendation and the incorporation of natural language instructions for personalized recommendations, respectively. Fan (2023) extends this discussion by addressing the limitations of Deep Neural Networks (DNNs) and emphasizing the potential of LLMs like ChatGPT and GPT4. Conversely, Di Palma (2023) proposes a fusion approach termed Retrieval-augmented Recommender Systems, combining LLMs with traditional RSs to enhance adaptability and relevance in sparse data scenarios. Liu (2023) further explores the utility of ChatGPT for recommendation tasks, demonstrating its effectiveness in generating contextually relevant suggestions without fine-tuning. On the other hand, Fan (2020) delves into social recommendations, presenting GraphRec+ as a comprehensive framework leveraging GNNs to model complex social networks and user-item interactions.

These studies collectively highlight the diverse approaches and methodologies employed to leverage advanced AI models for improving recommendation approaches and addressing various challenges including adaptability, personalization, and integration with

social structures.

Nevertheless, despite the number of works employing Large Language Models on systems that recommend software developers, there are still some challenges to be further comprehended regarding the recommendation of software developers for specific open positions. Hence, this study introduces distinctive contributions in contrast to existing related work. As a prevalent contribution, this research offers a system that assists in decision-making tailored for industry domains, addressing the needs of stakeholders. Furthermore, we investigate the integration of Large Language Models within systems that help in decision-making, emphasizing their potential in this context. The concept of Accuracy Zones introduced in our work provides a practical method for optimizing recommendation accuracy based on the number of characters used in Large Language Model prompts. Finally, our research culminates in developing functional software capable of recommending software developers under specific conditions, offering a unique and practical solution tailored to industry domains. This highlights the applicability and versatility of LLMs in systems that recommend software developers for real stakeholders.

4.4 DSR Cycles

The DSR approach functions through iterative phases, wherein both artifacts and procedures undergo development, reassessment, enhancement, and progression. For this reason, each cycle may introduce new requirements, challenges, and opportunities for refinement. Therefore, throughout the development of DevFinder, three main iterations were performed, incorporating additional requirements and refining the solution at the end of each iteration.

4.4.1 First Cycle

In the first cycle, we propose an approach that extracts data from multiple sources, as indicated as a relevant approach by the results of the systematic mapping conducted. Hence, we propose extractions on LinkedIn[1], TopCoder[2] and GitHub[3] databases. To achieve that, 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. As theoretical knowledge learned in this cycle, it stands out ontology and advanced databases theories comprehension, and as the main technical knowledge produced, it stands out: i) the modeling for extraction and integration of three different relevant bases, and ii) an ontology that considers collaboration aspects of the specialists found, resulting on a published paper (136). Regarding the

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

architectural functional requirements, this iteration of the DSR aims to work mainly on the FR 001, supporting data processing from domain-specific combinations of industries and technologies specified by the user, and FR 002 supporting data extraction from real-world repositories on version control systems. Other requirements were further implemented and assessed on the next DSR iterations.

As a way to gather data, Figure 9 presents an overview of the first cycle approach phases. The extraction process begins with API connections with the Local Conceptual Schemas (LCSs) (118). The extraction service then saves the raw data to a data lake (119). 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 is 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 and are 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.

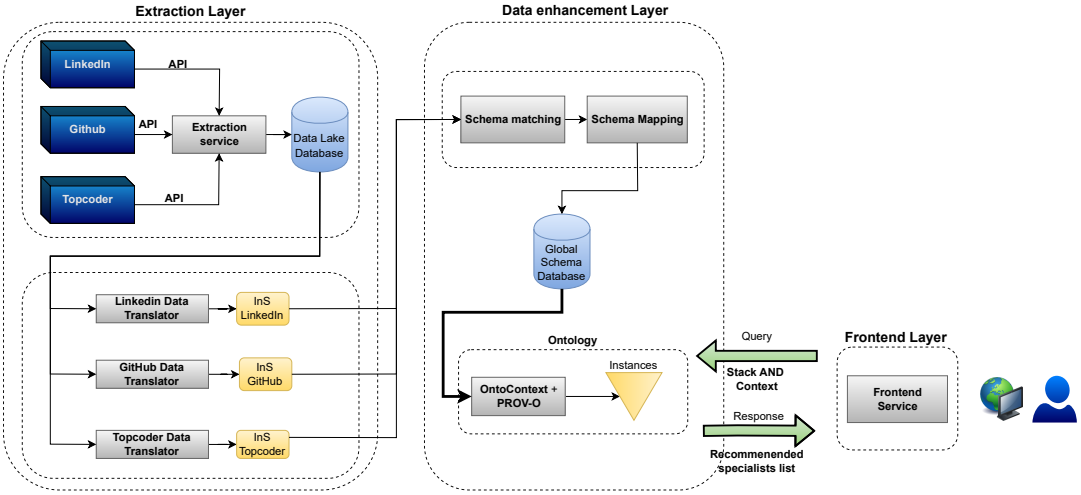
At the end of the extraction layer, the data from each database in its canonical representation flows into the enhancement layer. In this layer, the data is aggregated, and non-obvious inferences are made from the ontology.

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 is determined. In the Schema Mapping, the way in which each element of the LCSs is mapped to the Global Conceptual Schema (GCS) (118) is determined.

The database model is defined in advance with the GCS, so the results of queries prompted by the system user are restricted to the set of objects defined in the global model. This is a Global As View (GAV) model (118).

The ontology step is the last in the data enhancement layer. Non-obvious inferences are made from the data contained in the canonized base, which is an important intelligence

– Figure 9 - Overview of the first cycle proposed approach



step in the process. We propose a novel ontology[4] considering the GSC schema and the recommendations on provenance standards (120).

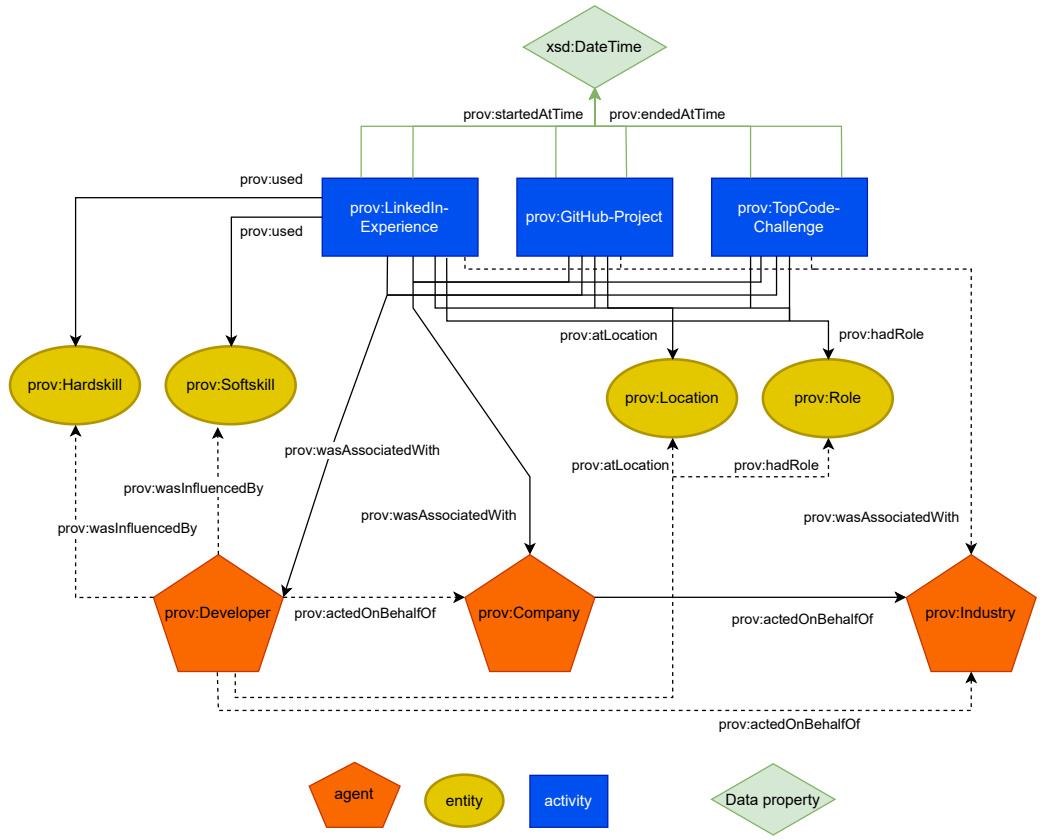
In order to represent the personal attributes of developers, hereby called soft skills, the list of such skills compiled by Matturro (2019) and described in their systematic mapping is used in this work. When querying the databases, our approach searches for such soft skills and assigns them to the related 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 (109).

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 10.

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 possesses. For that, we used Semantic Web

[4] 4 - https://www.github.com/vitorqcq/devFinderOntology
[4] 4 - https://www.w3.org/TR/prov-o/

– Figure 10 - Proposed ontology representation



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 11, 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)

- **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)

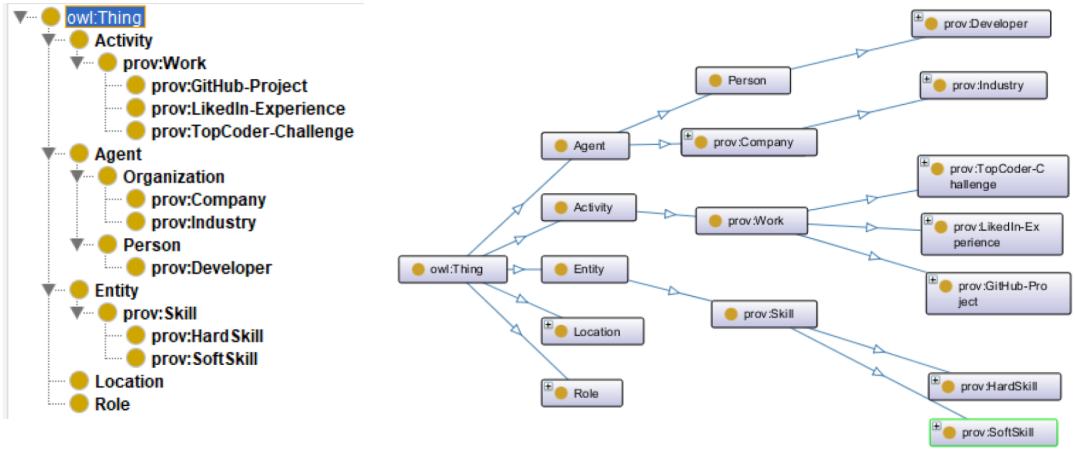
- **Rule 5:** $Role(?r) \frown hadRole(?w, ?r) \frown Work(?w) \frown wasAssociatedWith(?w, ?d) \frown Developer(?d) \rightarrow$hadRole(?d, ?r)

- **Rule 6:** $Work(?w) \frown wasAssociatedWith(?w, ?c) \frown Company(?c) \frown actedOnBehalfOf(?c, ?i) \frown Industry(?i) \rightarrow$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 11 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 11 - Representation of the proposed ontology in the Protègè system



4.4.1.1 Lessons Learned on the first cycle

The first development cycle of DevFinder brought to light some important lessons and decisions that had to be taken for the next iteration of the development of DevFinder.

The first lesson lies in the usage of multiple data sources. Even though the amount and variety of data of software developers from various sources may be beneficial, the consequences are overcomplicated to be dealt with. Handling diverse, non-standardized,

and disconnected data poses challenges that require extra-sophisticated approaches. For instance, the integrated platforms do not source data to match different users as being the same person, and thus, they are represented by different unique IDs or emails by each platform, which turns the work of matching users from distinct platforms a challenging task. Therefore, the first decision for the next cycle of DevFinder's implementation was to use a single data source, which is discussed and justified in the next section.

The second lesson learned for the next DSR iteration regards the usage of ontologies. While ontologies can be powerful tools for representing and organizing data in certain contexts, one key factor that led us to opt not to employ them for the next iteration. As listed in the requirements section, DevFinder is intended to evolve with dynamic data requirements and adapt swiftly to changing business needs. As also discussed by Welty (2021) and Motik (2007), while ontologies offer a structured way to represent knowledge, their inherent complexity and rigidity could potentially impede our ability to make agile adjustments and ensure extensibility principles (NFR 002) and requisites of scalability (NFR 003) (146, 147).

4.4.2 Second Cycle

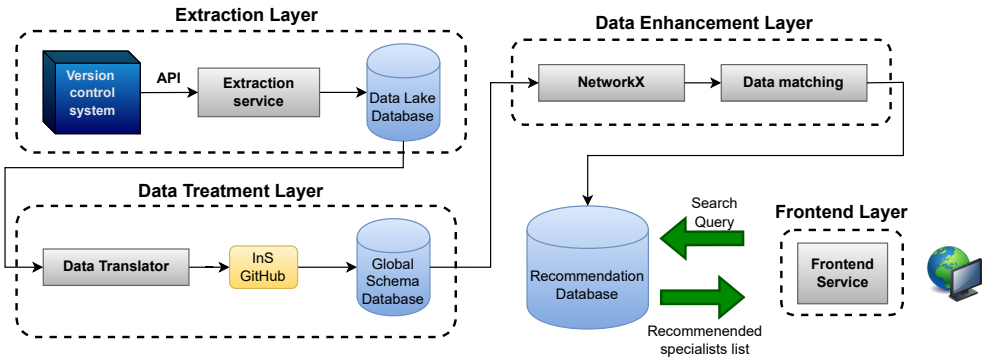
On the second DSR cycle, the focus of our work is the model of a complex network that aims to rank software developers through collaboration aspects. Thus, regarding the architectural functional requirements, this iteration also implements FR 001 and FR 002 as the first cycle, but it delves into the FR 004, supporting the software developers ranking based on collaboration metrics, and FR 006, supporting the provision of a final recommendation list so that user interfaces can be integrated into it. Other functionalities are discussed and implemented on the third DSR cycle.

Therefore, as a result of the second cycle development of theoretical, technological, and scientific knowledge was produced. As the main theoretical knowledge, it stands for the complex network for modeling graphs, resulting in the technical and scientific knowledge expressed in a published study (137).

As shown in Figure 12, the extraction process on the second cycle starts at the Extraction Layer, where requests are made to the version control system API, returning information from repositories and users by using Node.js for the requests and file manipulation. The main target at this point is to save the essential information from the API to make the network modeling possible. At the end of this layer, a JSON file is saved in the Data Lake database. A non-relational database expressing the classes and fields of the diagram of Figure 13.

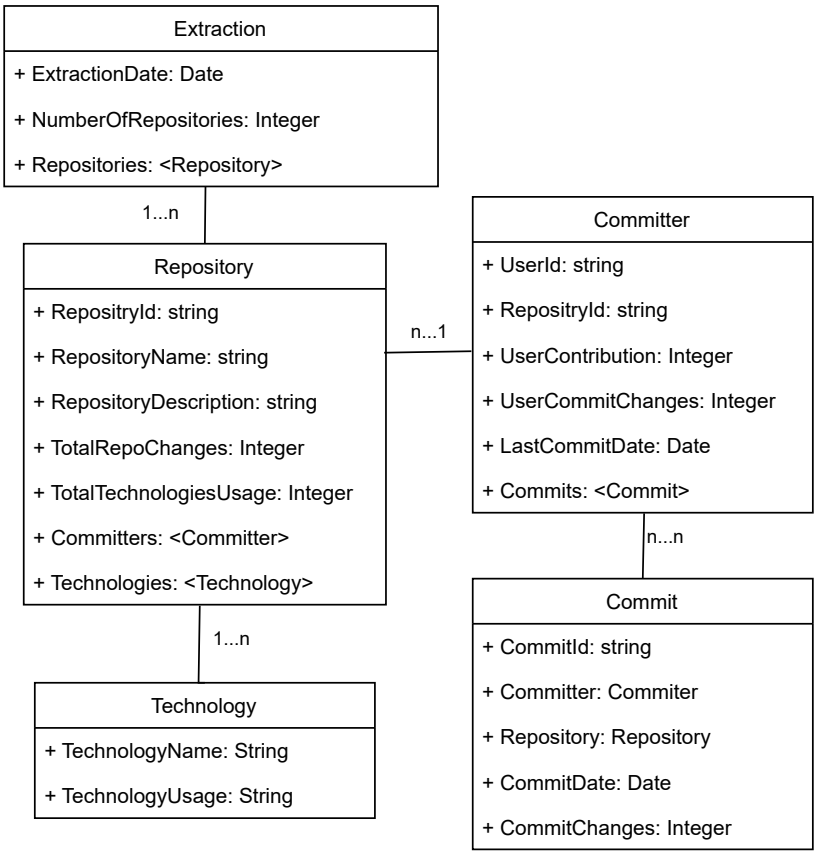
To better understand the further proposed network modeling of DevFinder, note that the field *TotalRepoChanges* in the *Repository* class stores the total number of changes done to that particular repository, whereas the field *UserCommitChanges* in the class

– Figure 12 - DevFinder overview on the second DSR cycle



Committer stores the number of changes done by a version control system user summing up all the number of changes in commits done by the user to this particular repository. In a later discussion, these values are important for calculating the impact of a committer on a repository.

– Figure 13 - Classes diagram of the Data Lake database



The *TotalTechnologiesUsage* in the *Repository* class and the *TechnologyUsage* in

the *Technology* are also worth attention. On the one hand, *TotalTechnologiesUsage* is intended to store the total number of bytes used to store all technologies of the repository on the version control system. On the other hand, *TechnologyUsage* stores the number of bytes used by those technologies targeted by the search process. These values are further used to calculate the impact of the target technologies on the repositories found.

The Data Treatment Layer converts data to the global schema format. This layer is also responsible for translating data if another source than the chosen version control system is used to ensure the project is extensible and modular.

The final intent of this layer is to provide the next one with a readable JSON file containing the necessary fields for generating the graph network. This layer also calculates the weights for the graph's edges. The final JSON file that serves as input for the next layer contains three fields: the committer ID, the repository ID, and the edge weight.

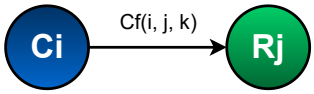
Further data processing takes place in the Data Enhancement Layer when information is better analyzed by going through NetworkX[1]. Being a Python package, NetworkX is responsible for creating, manipulating, and studying the proposed complex network of DevFinder. Finally, data is saved in the last database that a front-end service could use.

4.4.2.1 Network modeling

The DevFinder network can be represented as a bipartite graph $G = (V, E)$. In the proposed network, there are two kinds of vertices. A "Committer" vertex is a user from the version control system that contributes to the repository. A "Repository" vertex represents an artifact containing the source code committed by different Committers. The committers are defined as a set of nodes $C = c_0, c_1, ..., c_n$ and the repositories as a set of nodes $R = r_0, r_1, ..., r_n$ connected by a set of edges $E = e_0, e_1, ..., e_n$, where $C \cup R = V$.

To represent the weight of the edges of the network, we define the Contribution Factor (CF). CFs represent the contributions of a committer to a repository considering three aspects: i) the changes made by the committer through commits, ii) the technologies considered in the search terms, and iii) the dates when the committer has contributed to the repository. Hence, for every committer i committing to a repository j using the technology k, the weight, or Contribution Factor, can be written as $Cf(i, j, k)$, as shown of Figure 4.

– Figure 16 - DevFinder's Network representation



[1] 2 -https://networkx.org/

To calculate the proper CF for every edge of the network, the following expression can be assumed:

$$Cf(i,j,k) = Ruc(i,j) \times \left(\frac{Ucc(i,j)}{Trc}\right) \times \left(\frac{Pu(j,k)}{Tbu(j)}\right),$$

where $Ruc(i,j)$ is the User Contribution to the repository. This is a value given by the version control system API that aims to represent a user's contribution to a repository. It takes into account when a user commits to a project's branches, opens an issue, or proposes a Pull Request on the version control system.

$Ucc(i,j)$ represents the total user commit changes and Trc represents the total repository changes. With these two values, it is possible to measure the impact of developer commits on a repository.

Note that $Ucc(i,j)$ considers the date of each commit and results in a final value following the expression below:

$$Ucc(i,j) = \sum_{c=1}^{n} \frac{C(i,j,c)}{t - Ct0(i,j)},$$

where $C(i,j,c)$ represents the number of changes of commit c done by a committer i in the repository j. The extraction date is represented by t, $Ct0(i,j)$ is the commit date, and n is the total number of commits done by a committer i to a repository j.

The right-most parenthesis of $Cf(i,j,k)$ is intended to represent how much the target technology impacts the repository. $Pu(j,k)$ is the usage of technology k in repository j, whereas $Tbu(j,k)$ is the total usage of all technologies in a repository j.

4.4.2.2 Implementation choices on the second cycle

Once the outlined process considers the integration with a version control system, many systems could be used to play this role, such as GitHub, Git, Subversion, and others. The decision to integrate GitHub as the version control system into the application stems from its capacity to consolidate pertinent information within the realm of software development. As described by the systematic mapping conducted and pointed out by other authors in the area, GitHub serves as a robust platform capable of aggregating specific details crucial to our context, including the programming languages employed, executed projects, associated companies or institutions, and the duration of the experience (7). Leveraging GitHub enhances our system's ability to obtain a comprehensive profile of potential software developers, providing valuable insights into their skills, projects, and professional backgrounds.

Therefore, the GitHub API[1] is used to search the desired industry domain and technology through terms contained in the description fields of the repositories through the

[1] https://docs.github.com/pt/rest/search/search?apiVersion=2022-11-28

API's *q* parameter and also desired technologies used in the projects through the *language* parameter. In this project, the term *q* is used to enter the desired industry domain, and the term *language* is used to enter the desired technologies that an end user might be looking for.

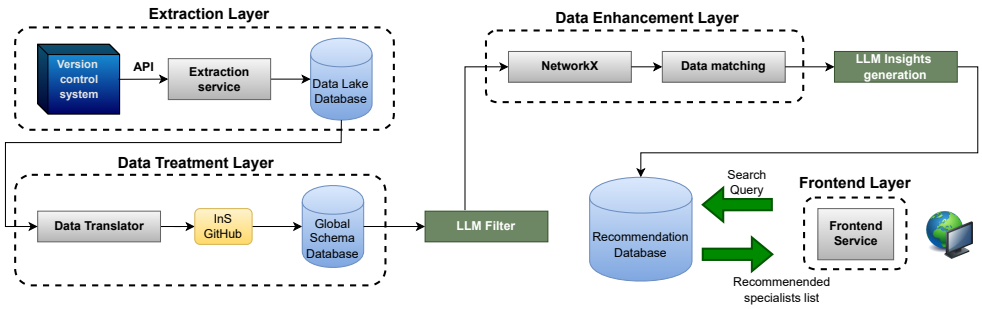
4.4.2.3 Lessons Learned on the second cycle

After publishing the study related to the second cycle (137), gathering feedback from other authors in the area, and investigating the results over DevFinder's recommendation list, two main improvements were identified as opportunities:

- Repositories filter: Some of the repositories returned from the search contained the industry terms, but were not strictly related to the industry itself. For instance, when the industry was related to oil or petroleum, repositories related to the game Minecraft[2] were returned.
- Data enhancement: The contribution factor could be supported by some more contextual and qualitative data regarding the recommended committers. For example, a committer has a Contribution Factor of 4.8 and is well positioned on the recommendation list, but other information returned from the API would still need to be comprehended by analyzing a large amount of data scattered across the committer's repositories.

Figure 15 depicts the extension of the diagram of Figure 13 representing our understanding regarding the two improvement areas where LLM could be used at the end of the second cycle where the green LLM Filter and LLM Insight generation boxes represent the steps to be included on the architecture.

– Figure 15 - Improvements representation on the second iteration of DevFinder



Therefore, to perform these improvements, a new iteration of DevFinder could leverage the capabilities of Machine Learning, as also suggested by the SMS conducted

[2] Minecraft is a sandbox game developed by Mojang Studios.

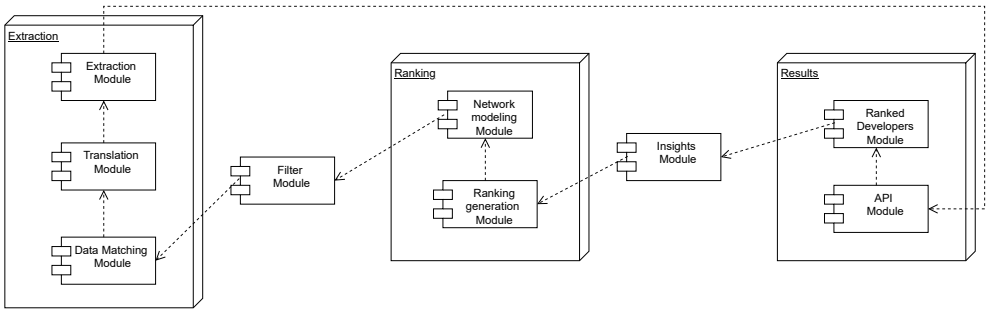
(138), to improve the two perceived weaknesses: the filter and the data enhancement. A thorough analysis of the second DSR cycle is explored in the Evaluation chapter.

4.4.3 Third Cycle

To incorporate the improvements derived from the lessons learned during the second DSR interaction, the new version of DevFinder architecture integrates additional modules. Therefore, as functional requirements implemented in this iteration we highlight FR 003, supporting the Repositories filtering that are not related to the specified industry domain, and FR 005, supporting the provision of insights about each ranked software developer generated through the available data of software developers. Thus, in this iteration all the functional requirement were considered, however, a higher attention to the new two implemented, FR 003 and FR 005, was given.

As theoretical knowledge of this iteration, it stands out the LLM and Machine Learning theories, and as technical and scientific knowledge, it stands for the study in the acceptance process of the Information and Software Technology (IST) journal. Hence, the new architecture, including the enhancements of the new iteration, is depicted in the components diagram shown in Figure 16.

– Figure 16 - Components Diagram of DevFinder



A complete description of functionalities is elaborated in the following sections. Below is a brief description of each module:

- **Extraction** (Addresses RF 001, RF002, NRF 001 and NRF 003):
  - **Extraction Module**: Collects data from various repositories and developer activities, serving as the initial step in data acquisition.
  - **Translation Module**: Converts extracted data into a standardized format, ensuring consistency for further processing.
  - **Data Matching Module**: Aligns the standardized data with predefined search criteria to filter out irrelevant information.

- **Filter Module** (Addresses RF 003): Uses large language models to refine data by excluding repositories and profiles that are not directly relevant to the specified industry or technology.
- **Ranking** (Addresses RF 004 and NRF 002):
  - **Network Modeling Module**: Builds a network model to analyze relationships and contributions among developers, highlighting key contributors.
  - **Ranking Generation Module**: Uses the network model to rank developers based on their contributions, technology expertise, and activity levels.
- **Insights Module** (Addresses RF 005): Generates detailed profiles and insights about developers, aligning their skills with job requirements and providing context for recruitment decisions.
- **Results** (Addresses RF 006):
  - **Ranked Developers Module**: Stores the ranked list of developers, making it easily accessible for retrieval and analysis.
  - **API Module**: Provides external access to architecture implementation through APIs, turning it available to integration with other applications and systems.

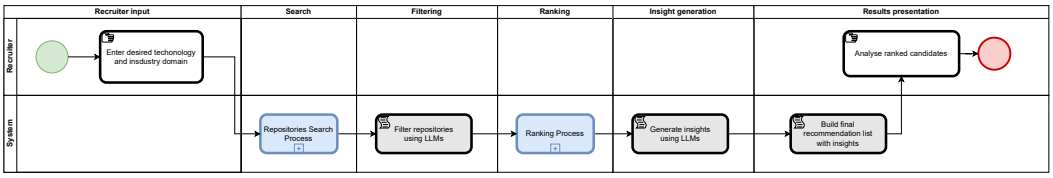
In order to further discuss the implemented system on a more detailed level, we have opted to describe the solution in a Business Process Model (BPM) notation. The adoption of BPM representation was deliberate, driven by its ability to provide a standardized portrayal of business processes. This choice aims to enhance clarity and facilitate effective communication among diverse stakeholders, including business analysts, developers, and end-users (22), all of whom may find value in utilizing the designed solution. Notably, BPM notation serves as a shared language, fostering cross-functional collaboration (23), an aspect that is particularly advantageous for ensuring the reproducibility of the study, aligning with the best practices in research and system design.

Thus, in Figure 17, the activities contained in the "Recruiter input"and "Search"columns are representatives of the "Extraction"block from the components diagram of Figure 16. In the same way, the activity in the "Filtering"column represents the "Filter Module", the activity in the "Ranking"column represents the "Ranking"block, the activity in the "Insight generation"column represents the "Insights Module"and the activities contained in the "Results presentation"column are the representation of the "Results"block.

Therefore, the third DSR iteration primarily focuses on the incorporation of new LLM modules into the DevFinder architecture. Therefore, it's crucial for the reader to pay close attention to two key tasks: "Filter repositories using LLMs" and "Generate insights using LLMs". As depicted in Figure 17, the process involves two key actors:

i) Recruiter and ii) System. Human interaction activities from the Recruiter actor are accommodated in two distinct lanes: the Recruiter Input lane and the Result Presentation lane. The remaining lanes are dedicated to the steps undertaken by the System actor, encompassing the search, ranking, and filtering processes for the results obtained from repository searches. DevFinder's primary process encompasses the specified actors and lanes, along with sub-processes like the Search and Ranking processes. The activities within the process involve the utilization of large language models. The culmination of the entire sequence is marked by a concluding script activity responsible for generating the final JSON file containing the recommended software developers.

– Figure 17 - DevFinder BPM representation



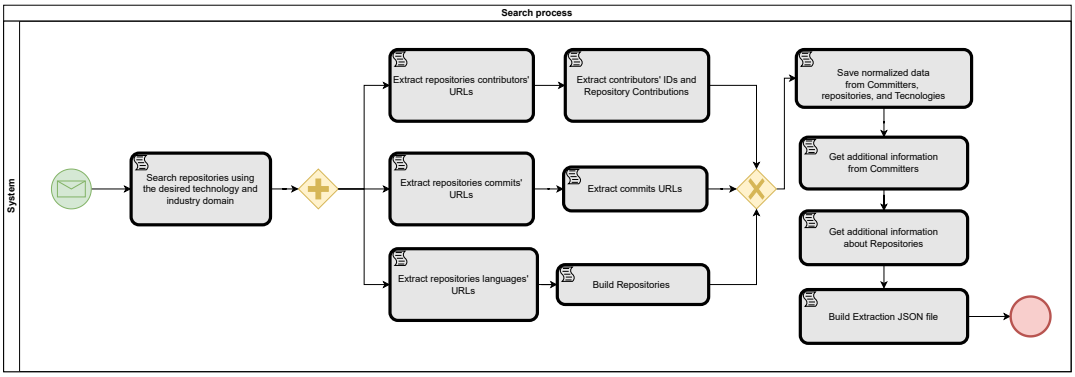
4.4.3.1 Inputting activity

The inputting activity's main objective is to prompt the system with the desired technology and industry domain to be used in the search process. It would be the equivalent of filling the 'Industry' and 'Technology' fields, and hitting the button 'Search' to start the process.

4.4.3.2 Search sub-process

This sub-process aims to systematically explore and retrieve information about potential repositories and software developers from a version control system, as shown in Figure 19.

– Figure 19 - Search sub-process BPM representation



The initiation of a sub-process involves a crucial step of searching repositories. This means inputting specific criteria for desired technology and industry to find relevant repositories. Then, three activities happen simultaneously to gather key information.

Firstly, contributors' URLs are collected, providing data about people actively contributing to the repositories. At the same time, commits' URLs are gathered, offering a comprehensive understanding of the development history in these repositories. Additionally, languages' URLs are extracted to show the coding languages used.

Once these activities are done, the system organizes and saves data from the version control system. This ensures that the collected information is structured for further analysis. Using the foundational data, the sub-process retrieves additional information regarding the technologies of the repositories and additional information about the committers.

The sub-process concludes by creating an Extraction JSON file. This file encapsulates all the extracted data, offering an overview of the software developers identified during the search. The class diagram in Figure 13 (from the second DSR cycle) demonstrates the data structure contained in the JSON, being composed primarily by the Extraction Class, which contains general information such as the extraction date, the number of repositories retrieved, and an array of repositories. Each object of the class Repository is stored with the repository ID, name, description, a number to represent the number of bites contained in the performed changes, and data structures to represent the Committers and the Technologies of each Repository.

4.4.3.3 LLM filter activity

Some of the repositories retrieved contained industry-related terms but were not directly aligned with the specified industry. Hence, the utilization of LLMs addresses a particular challenge encountered during the retrieval of repositories.

To mitigate such instances of misalignment, our filter harnesses LLMs' comprehensive language understanding capabilities. By evaluating the nuanced content of repository descriptions, the LLM-based filter distinguishes repositories genuinely pertinent to the specified industry from those that only contain industry-related terms tangentially.

As a way to assess the effectiveness of the prompt used to filter repositories in this activity, two approaches were tested.

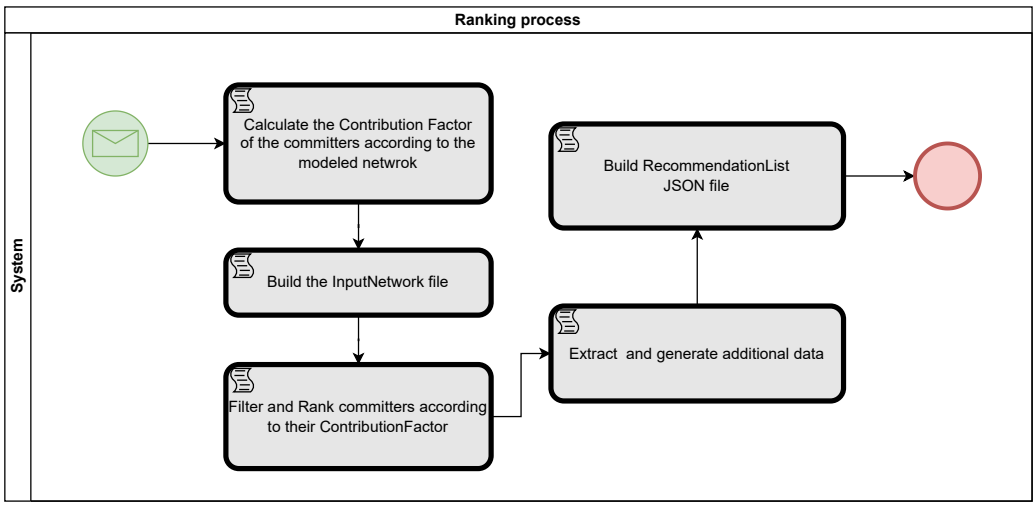
The main idea of Prompt 1 is to give one single instruction to the LLM containing all the repositories and their descriptions. The LLM would then give a single answer containing only the repositories related to the specified industry domain. On the other hand, prompt V2 aims to request the LLM to answer a singular boolean value, representing a granular representation of the correlation between one repository to the industry domain being analyzed. In the Evaluation section, we discuss the comparative results of each approach.

4.4.3.4 Ranking sub-process

In the previous DSR cycle, we introduced the concept of the Contribution Factor (137). This factor serves as a fundamental element for the construction of a graph network, enabling the classification of the repository's committers according to their contributions. Our study delves into the comprehensive analysis of three key factors: i) the influence of a developer within the target context, ii) the technological aspects covered by the search terms, and iii) the temporal dimension, specifically the date on which a developer made contributions to the project.

The ranking sub-process (Figure 19) takes advantage of the Contribution Factor modeling and results in a recommendation list JSON as output.

– Figure 19 - Ranking sub-process BPM representation



The initial step involves quantifying the Contribution Factor for each committer within the repository, guided by the intricacies of the modeled network, as already discussed. Leveraging graph-based algorithms and network analysis techniques, this calculation takes into account factors such as code commits, issue resolutions, and collaborative interactions. The Contribution Factor serves as a quantitative metric, offering a nuanced evaluation of a committer's impact on the collaborative software development ecosystem (137).

After calculating Contribution Factors, an important step involves filtering and ranking committers based on their respective metrics. This activity aids in identifying and highlighting contributors who significantly influence the software development process. The ranking mechanism provides a clear hierarchy of contributors, offering insights into their relative impact and contributions. Additional data is extracted and generated to enrich the understanding of committers' contributions and further refine the ranking process.

The final step involves consolidating the ranked committers and their associated information into a RecommendationList JSON file. This file serves as a resource, providing a structured representation of committers, their Contribution Factors, and additional insights. The JSON format facilitates seamless integration with other tools and systems, enabling downstream processes such as team formation, mentorship allocation, and project assignment based on the nuanced analysis of contributor dynamics.

4.4.3.5 Insight generation activity

On the one hand, if the Contribution Factor is used for ranking developers on the final recommendation list, on the other hand, a representation of how much the data of each developer is aligned with the requirements of the hiring company would provide a better glimpse of the person being screened. This contextual data is essential for accurately representing each professional's profile, particularly considering the unique requirements of the specific job opening. For instance, if a hiring company is seeking a developer with extensive experience in Python and a background in data science, the system not only ranks developers based on their overall contributions but also highlights those whose work and skills specifically match these criteria. This contextual data could be essential for accurately representing each professional's profile, ensuring that the unique requirements of the specific job opening are met.

Therefore, the core objective of the insight field is to offer a comprehensive overview of developers and an initial analysis conducted by an LLM to assess the alignment between the job specifications and the technical skills possessed by the developers.

A prompt is crafted for the LLM to achieve this, incorporating pertinent developer information sourced from their repositories. This includes details such as name, email, address, bio, repository descriptions, and the primary languages used in the repositories. The prompt also encompasses information about the technology and industry domain specified by the Recruiter as constraints for the search. By providing this comprehensive data set, the LLM can generate insights that facilitate a more informed evaluation of the match between the developers' capabilities and the specific requirements of the job at hand.

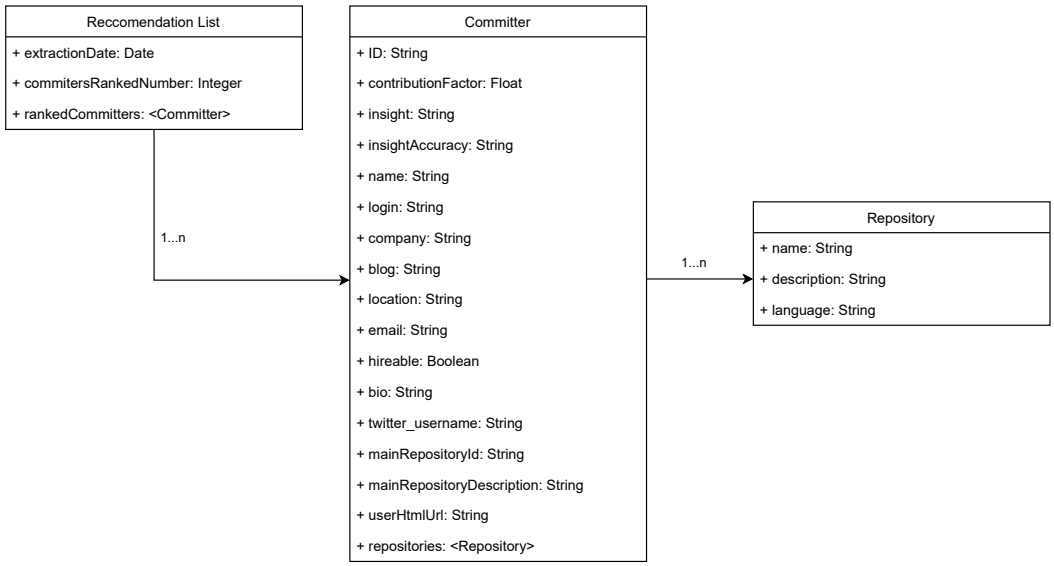
The construction of the insight generation prompt was guided by strategies outlined by Fan (2023). These strategies contextualize the LLM, providing clear instructions on the nature of the request and encouraging it to produce responses that positively impact job recruiters involved in the personnel selection process (8). Further discussed in the Evaluation section, this approach aims to optimize the relevance and quality of the generated insights, aligning them with the specific needs and expectations of recruiters in the context of personnel selection.

4.4.3.6 Final JSON activity

The primary function of the last script activity is straightforward: it retrieves data from preceding steps and assembles a final JSON file, which serves as the solution's response. Figure 20 demonstrates, through a class diagram, the resultant structure of the DevFinder architecture. This structure aims to provide all the essential information to a requesting service that may use the API Module of DevFinder.

On the Recommendation List class representation, there is information regarding the request itself, such as the extraction date, the number of returned committers, and the list of committers. The list of the committers is composed of items of the Committer class representation, containing not only essential information of the committer, such as name and email, but mainly the contribution factor, the insight, and the insight accuracy for the committer. Beyond that, each Committer is attributed with a list of repositories represented by the class Repository, which is formed by the fields: name, description, and language.

– Figure 20 - Classes Diagram representation of the final JSON



This JSON, once generated, provides integration into other applications through an API. Consequently, the solution can respond to requests and deliver a curated list of recommended software developers as a service through the API Module (Figure 16). Therefore, multiple applications could leverage the solution proposed by the implementation of DevFinder architecture.

4.4.4 Non-functional requirements assessment

Regarding the described implementation and the outline non-functional requirements (NFR), the NFR can be considered to be met. The dependability (NFR 001) of the DevFinder system is enhanced by the design and implementation of the API Module within the Data Storage Block. This module is responsible for providing data access to other systems that consume data from DevFinder. To adhere to extensibility principles (NFR 002), the system is architected with modular components that can be extended or replaced. The clear separation of concerns within the Extraction Block, Filter Module, Ranking Block, and Insights Module allows for the seamless addition of new features or enhancements. Scalability (NFR 003) is a core consideration in the DevFinder system's design. The architecture supports the integration of new data sources and the scaling of data processing capabilities. The use of advanced large language models (LLMs) in the Filter and Insights Modules guarantees that the analysis and filtering processes remain relevant and accurate.

4.4.4.1 Implementation choices on the third cycle

To operationalize and assess the outlined process, decisions must be made regarding selecting systems for integrating the version control system and Large Language Models (LLMs).

Once the outlined process considers the integration with a version control system, many systems could be used to play this role, such as GitHub, Git, Subversion, and others. The decision to integrate GitHub[3] as the version control system into the application stems from its capacity to consolidate pertinent information within the realm of software development. GitHub serves as a robust platform capable of aggregating specific details crucial to our context, including the programming languages employed, executed projects, associated companies or institutions, and the duration of experience (7). Leveraging GitHub enhances our system's ability to obtain a comprehensive profile of potential software developers, providing valuable insights into their skills, projects, and professional backgrounds.

For instance, consider a scenario where our system needs to evaluate the expertise of a software developer named Jane Doe. By integrating GitHub, our application can automatically pull Jane's profile, which includes her repositories, contribution history, and specific details about the programming languages she has used. For instance, Jane's GitHub profile reveals she has actively contributed to several projects involving Python and JavaScript over the last three years, working on diverse applications from web development to machine learning. This information helps our system to compile a detailed portfolio, highlighting her proficiency in these languages and the nature of her projects.

[3] https://github.com

Various models can be employed to fulfill this function when incorporating LLMs integration into the outlined process. Examples include GPT (Generative Pre-trained Transformer), BERT (Bidirectional Encoder Representations from Transformers), T5 (Text-to-Text Transfer Transformer), and others. The choice of employing GPT[4] as the Large Language Model for the application is grounded in its proficiency in language-related tasks. GPT is primarily designed for language generation, excelling in tasks that involve mapping embedding vectors back to the text space and generating contextually relevant responses (8). This characteristic aligns with our system's objectives, where effective communication and contextual understanding play a pivotal role. By integrating GPT, our application can generate insights and responses based on the collected data, fostering a more dynamic and responsive user experience (8).

4.5 Final Remarks of the Chapter

This chapter explores the DevFinder architecture, detailing its key characteristics and methodologies for recommending software developers. We refined the system through cycles of the Design Science Research framework, focusing on improving the repository filter and the data.

In the first DSR cycle, DevFinder was proposed to be integrated into three data sets and to use ontologies, which were revised to be removed from the solution. The second cycle reveals insights on how to optimize the relevance of results and the importance of contextual data for the Contribution Factor, evaluating a developer's impact on collaborative software development. In the third cycle, we integrated Large Language Models (LLMs), improving repository filtering and analysis and in the representation of the Business Process Model (BPM), increasing clarity in DevFinder architecture. The iterative journey exemplifies Design Science Research, continually refining the solution's functionalities.

A thorough analysis of the third DSR cycle is explored in the Evaluation chapter.

[4] https://openai.com

5 Evaluation

In this chapter, we present an evaluation and assessment regarding DevFinder architecture steps in generating the recommendation list. Firstly, we delve into the complex network discussion and secondly, we deepen the assessment regarding the LLM integration.

5.1 Complex network

This section explores the evaluation of the Complex Network proposal implemented in the second DSR cycle and discussed in Chapter 4. To further understand the results of DevFinder in terms of collaboration, the interaction between the proposed Contribution Factor with other classic metrics, such as closeness and degree centrality, was investigated. Considering the proposed bipartite graph, as presented on the second cycle of development of DevFinder, a committer node with a high closeness value would represent a developer who committed to multiple repositories, possibly indicating a strong collaboration behavior. As so, a committer node with a high degree of centrality value would also indicate a strong collaboration behavior once this node would have a high number of connections.

Aligned with the proposed Contribution Factor, the most desired software developer would be ranked at the top of the three lists: Contribution Factor, closeness, and degree centrality. Hence, the final result of this approach is the intersection between the best-ranked committers regarding these three metrics. We define the committers with the highest Contribution Factors as $TopCf$, the committers with the highest closeness centrality as $TopCl$, and the committers with the highest degree centrality as $TopDg$. Therefore, the final recommended specialists are represented by the intersection $TopCf \cap TopCl \cap TopDg$.

To assess the proposed solution's effectiveness, we performed instances aligning different industry domains and technologies. The chosen contexts were: i) Education and PHP, ii) Oil and JavaScript, and iii) Finance and Python. These combinations of industry domains and technologies were chosen due to the higher numbers of repositories returned after a study of multiple API responses to different combinations.

Applying the intersection to the three chosen contexts, we obtained the final number of committers. As shown in Table 7, for the Education and PHP search, the initial number of committers was 233, and the final 16. For the second search, Oil and JavaScript, the initial number of committers was 222 and the final number was 33. Finally, the Finance and Python search presented an initial number of committers of 239 and a final number of 31.

Figures 21, 22, and 23 show box plot graphics of the mentioned metrics for each of the three combinations of industry domain and technology. The metric values were normalized so that a better understanding of their importance could be discussed.

Table 7 – Number of committers

	Education and PHP	**Oil and JavaScript**	**Finance and Python**
Initial number of committers	233	222	239
Final number of committers	16	33	31

For the first search, shown in Figure 21, the highest contribution factor value was 0.7949, the lowest was 0.0112, and the standard deviation of 0.2110. These numbers show a great leap between the final experts' CFs, which would result in higher attention to the best-ranked experts on the list by a final user of the RS looking for a new contributor for his or her team.

The first expert on this search was also best allocated in terms of closeness and degree centrality. The difference in closeness between the first and second places in our ranking is 0.1142, and the difference in degree centrality is 0.3781. When the last expert on the list has a degree centrality of 0.1944, it turns out that only the difference between the first and second-best-ranked experts is higher than the last-raked expert's degree centrality. This shows that the first expert not only contributed to one repository but to multiple repositories, being well connected to the network, becoming the most sought-after specialist if you are an RS end user looking for a new collaborator for your team in the PHP and Education context.

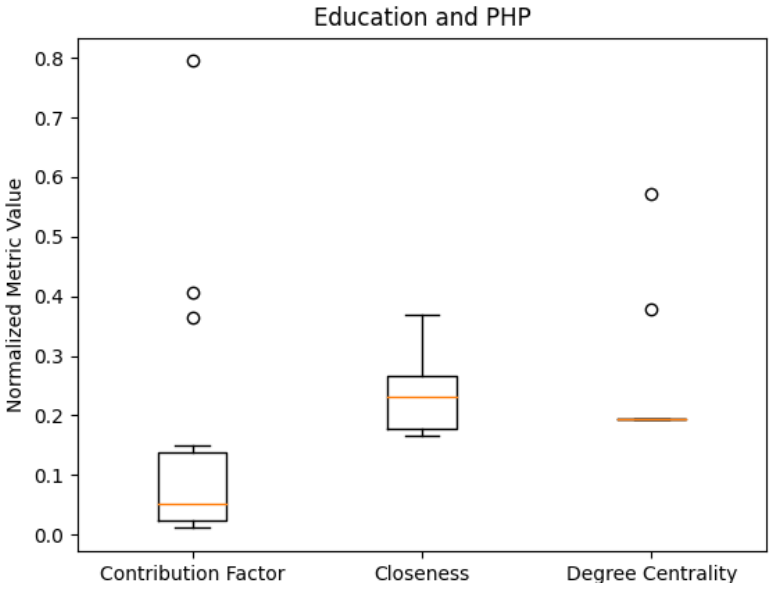
In the second search, shown in Figure 22, the highest contribution factor value was 0.6448, the lowest was 0.0158, and the standard deviation of 0.1395. Despite having a considerable standard deviation, the CFs values are more homogeneous than those in the first search.

The highest Closeness was 0.3728 and the highest degree centrality was 0.4376. Nonetheless, the ranked expert with these highest metrics is not the same as the best ranked in the Contribution Factor. In fact, the position of the best-ranked expert considering closeness and degree centrality is 26 from 33.

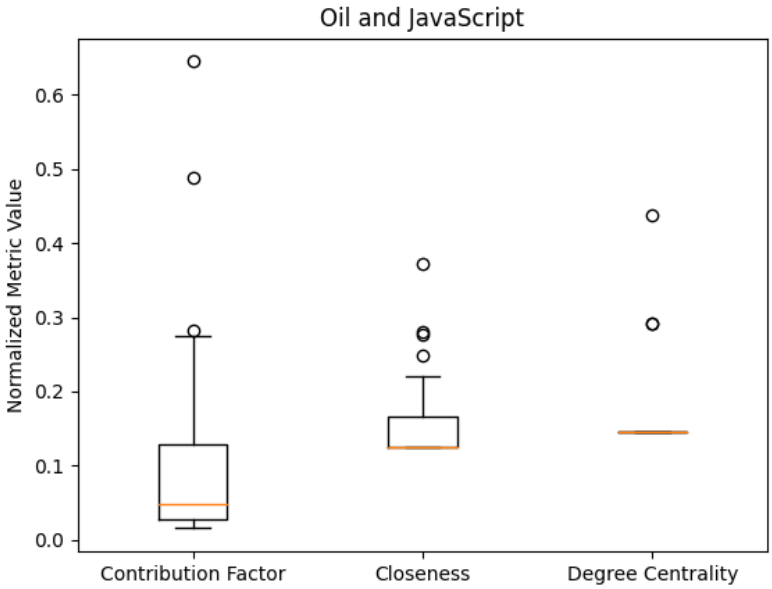
In this case, if you are a recruiter looking for a new team member, a choice would have to be made. The first-positioned experts from this combination of technology and industry domain are those with the highest contributions to the repositories they have committed, but they are not the same as those who committed to the highest number of repositories. The ones with the highest Closeness and Degree Centrality metrics would likely be the ones with the highest chances of having collaborative behavior.

In the third search, shown in Figure 23, the highest Contribution Factor value

– Figure 21 - Results of the first search



– Figure 22 - Results of the second search

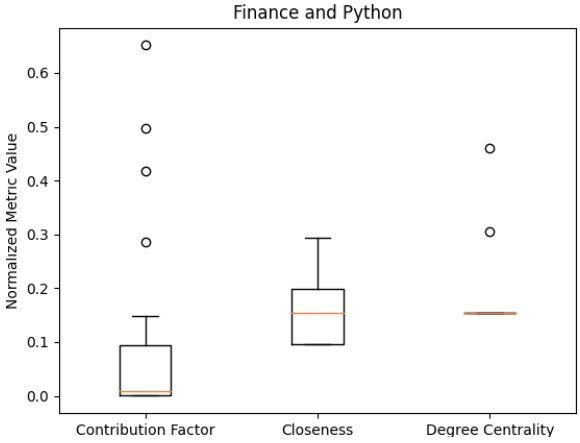


was 0.6513, the lowest was 0.0004, and the standard deviation of 0.1615. This was the most homogeneous result of the three, presenting more committers with high scores at the top of the list. In terms of the Contribution Factor, a recruiter looking for a new team member would have more options, once there is a higher number of experts with a relatively elevated CF.

From a Closeness and Degree Centrality point of view, something similar to the

second search happens: the ranked expert with the highest metrics is not the same as the ones with the highest CFs. Hence, a decision would have to be taken to prior different collaboration aspects when recruiting this new team member.

– Figure 23 - Results of the third search



The discussed results are a positive indication that our research question is attended. DevFinder solution is capable of providing suitable specialists that meet the criteria of technical specificity allied to the specificity of acting in specific domains of the industry. The presented data would also ease recruiters' decision-making in terms of collaboration, indicating different aspects of the ranked experts.

5.1.1 Analysis

Further analysis can be made over the final returned committers. Figure 24 shows the Contribution Factor Frequency. The x-axis represents the Contribution Factor, whilst the y-axis represents the number of committers having the same value of Contribution Factors.

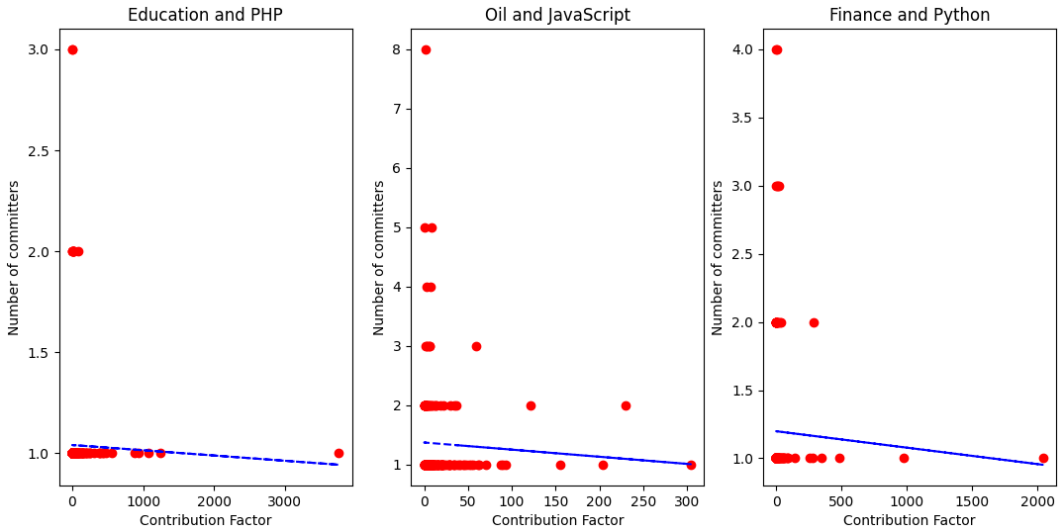
It is possible to see a trend in the three different contexts when the same sort of behavior happens to all searches. There is a high number of CFs with values near zero. These committers could be understood as those who contributed little to repositories, being the most of the final committers list. On the other hand, there are few contributors with high CF values, being these ones the most desired contributors to a project in each context.

There are differences between the three contexts, though. The education and PHP along with Finance and Python graphics show fewer intermediary committers between the lowest and highest CF values, whilst the Oil and JavaScript demonstrate a more heterogeneous distribution on the x-axis. This would mean that if specialists from the top

of the Oil and JavaScript list eventually did not join a recruiter's project, there would be more intermediary options to fulfill the role.

It should be highlighted that these graphics take into account only the Contribution Factor values of all committers found. The discussed intersection between CF, closeness, and degree centrality filtered some of the committers from the list, even those with high CF values. Due to this fact, some of the highest committers shown on the graphic are not present in the previous Figures and analyses once they did not possess high enough Closeness and Degree centrality values.

– Figure 24 - Contribution Factor Frequency

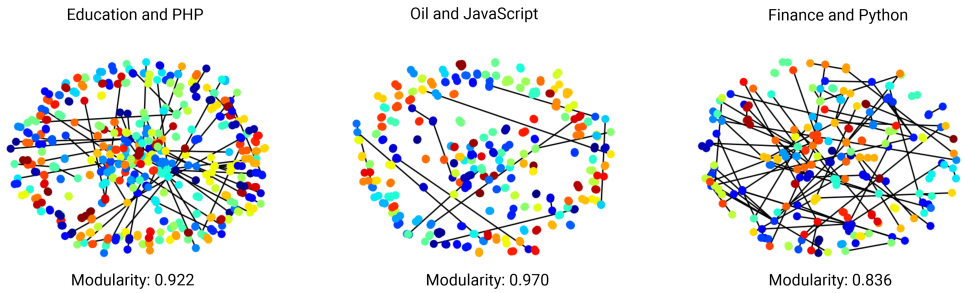


Once we now have data from three different contexts, it is possible to take a broader look at the committers as a community. In an effort to investigate deeper collaboration aspects of the three different contexts as communities, the Modularity Clustering for each of them was calculated by first calculating partitions of the graph using the Louvain heuristics and then plotting the graphics on Figure 25 using NetworkX packages.

A network with high modularity would have a dense connection between its nodes but also sparse connections between different clusters. In our graph, each color would mean a different community or cluster and a connection between them would indicate a possibility of collaboration.

An interesting aspect of the plotted graphic is that the Education and PHP along with Finance and Python show a denser connection overview, whilst the Oil and JavaScript network presents a less dense connection overview. This could indicate that these two denser would have a higher chance to present committers that collaborate between them.

– Figure 25 - Modularity Clutering



5.2 Large Language Model

This section explores the evaluation of the Large Language Model proposal implemented in the third DSR cycle and discussed in Chapter 4. In the world of recommendation tasks, Large Language Models have been often employed. Liu (2023) demonstrated a method employing ChatGPT as a system where task-specific descriptions were introduced to enable few-shot Instructed Commonsense Learning (ICL) based on input-output examples tailored for distinct recommendation tasks (9). In-context demonstrations have also proven valuable for enhancing recommendation performance. Building on this, a text description strategy, as outlined by Zhiyuli (2023) involves contextualizing the LLM's function with prompts like "You are a book rating expert" to augment in-context demonstrations (10).

5.2.1 Method

In order to evaluate the new proposed iteration on DevFinder, we propose the following Goal, Question, and Metric (GQM) parameters (161). GQM was chosen due to its structured framework for aligning measurement activities with project goals. GQM enables clear goal-setting, precise question formulation, and relevant metric definition, ensuring a comprehensive evaluation of LLM performance in recommendation tasks.

5.2.1.1 Goal

To assess the efficacy of the integration of Large Language Models integration to DevFinder on the filtering task and on the insight generation task.

5.2.1.2 Questions

For the filtering task:

- Q1: How accurately can the LLM filter repositories be relevant to specific industry domains?

- Q2: What is the precision and recall of the LLM in distinguishing relevant repositories from irrelevant ones?
- Q3: How does the LLM's performance vary between different prompt models and industry domain contexts?

For the insight generation task:

- Q4: To what extent does the LLM generate insights that are strictly related to the input data?
- Q5: How often does the LLM provide insightful analyses regarding the contributor's suitability for a position?
- Q6: How do the lengths of input prompts affect the accuracy of insights generated by the LLM?

5.2.1.3 Metrics

For the filtering task:

- M1: Precision, recall, F1-score and accuracy (%) for each version of the filter prompt (V1 and V2) across different industry domain contexts.
- M2: Comparative analysis of confusion matrices to assess the performance of different prompt models in each industry domain context.

For the insight generation task:

- M4: Percentage of insights strictly related to the input data.
- M5: Percentage of insightful analyses regarding contributor suitability for a position.
- M6: Correlation between prompt length and the occurrence of incorrect position matching in insights generated by the LLM.

Our filter evaluation incorporates 855 repositories from four case studies involving different industry domains and technologies: i) Insurance & Python (138 repositories), ii) Research & Python (230 repositories), iii) Banking & Java (249 repositories), and iv) Games & C (238 repositories). Depicted on Figure 19, the search for the repositories considers both technology and industry domains on the API request, returning repositories containing the desired technology as the primary programming language and the industry within the terms on the title and description of the repository. The selection of these specific combinations was strategic, driven by the need for a comprehensive understanding

derived from a larger number of repositories obtained through the GitHub API after testing other combinations. The chosen industry domains and technologies serve as real-world contexts for evaluating the LLM filter and insight generation capabilities. The results of this assessment contribute to our understanding of LLMs in recommendation tasks and provide information for market stakeholders, organizations, and job recruiters. Our analysis aims to bridge the gap between theoretical advancements and practical applications by focusing on real-world scenarios.

In the evaluation of the filtering task, we manually classified 855 repositories into those related and not related to the specific industry domain under analysis. The methodology for this classification involved an examination of each repository's title and description by the authors of this study, followed by the assignment of a Boolean value indicating its relevance to the subject industry domain. After that, a comparison between the author's classification of each repository and LLM's classification for each repository was done to determine the accuracy rate of the filtering task using the Large Language Model.

As previously mentioned, two versions of a filtering prompt were designed with distinct approaches to address concerns about the explainability of LLMs, often viewed as "black boxes" (8). We sought to understand the behavior of the LLM in the filtering task. To achieve this understanding, we tested the two distinct prompt models in the application and discuss the results on the following paragraphs.

The first model, Filter Prompt V1, instructed the LLM to analyze and filter a single JSON file containing all repository descriptions. On the other hand, Filter Prompt V2 involved more granular evaluations, requesting the LLM to determine the relevance of a specific repository to the industry domain. Therefore, the two proposed versions of the filtering prompt are:

> **Filter Prompt V1:** "Filter the JSON below to include items with descriptions related to the ${industry} industry. Use the 'Description' field for filtering. Your answer should be in JSON format, listing the remainder repositories.
> Here is the input JSON: ${repositoriesJSON}."

> **Filter Prompt V2:** "Return 0 or 1. The character should be 1 if the description below is correlated to the ${industry} industry and 0 if it is not. Your answer should have only one character. Description: ${repositoriesDescriptions}."

To answer questions Q1, Q2, and Q3, while also applying the metrics M1 and M2 of the proposed GQM, we provide a set of confusion matrices (24), delineating the

performance metrics for every filter version within each distinct case study. The confusion matrices facilitate the systematic examination of true positive (PP), true negative (NN), false positive (PN), and false negative (NP) classifications, offering a comprehensive overview of the model's strengths and weaknesses in distinguishing repositories meeting a specified condition from those that do not.

A true positive (PP) represents a case where the authors' classification tagged a repository as related to the desired industry and the LLM also tagged it as related to the desired industry. A true negative (NN) occurs when both the model and authors identify the repository as not related to the desired industry. A false positive (PN) arises when the authors classify a repository as related to the desired industry, but the model incorrectly classifies it as not related. A false negative (NP) occurs when the authors correctly classify a repository as related to the desired industry, but the model incorrectly classifies it as not related.

5.2.1.4 Insurance & Python

As shown in Table 8, in the case of Filter V2, there is an improvement in accurately identifying actually positive instances compared to Filter V1, achieving a higher percentage of predicted positives for actually positive cases (69.06%) compared to Filter V1's 19.57%. This demonstrates a substantial increase in the ability of Filter V2 to identify positive instances correctly.

Table 8 – Confusion matrix for the first case study

Insurance & Python					
Filter V1			**Filter V2**		
	Actually Positive	**Actually Negative**		**Actually Positive**	**Actually Negative**
Predicted Positive	19,57%	60,14%	**Predicted Positive**	69,06%	9,35%
Predicted Negative	5,07%	15,22%	**Predicted Negative**	12,23%	9,35%
Precision		**24,55%**	**Precision**		**88,07%**
Recall		**79,41%**	**Recall**		**84,96%**
F1		**37,50%**	**F1**		**86,49%**
Accuracy		**34,78%**	**Accuracy**		**78,42%**

Moreover, Filter V2 shows progress in reducing false positives, with only 9.35% predicted positives for actually negative cases, whereas Filter V1 had a higher false positive rate of 60.14%. This indicates an improvement in the precision of Filter V2.

However, challenges remain evident. The true negative rate for Filter V2 is 9.35%, which is lower than Filter V1's 15.22%. Additionally, the false negative rate for Filter V2

is 12.23%, higher than Filter V1's 5.07%. This suggests that there are areas where Filter V2's performance does not surpass that of Filter V1.

Overall, the PP + NN values highlight that Filter V2 achieves a combined accuracy of 78.42%, significantly higher than Filter V1's 34.78%. This represents a substantial improvement of 43.64% in overall accuracy for Filter V2 in this case study, meaning that Filter V2 could perform better than V1 on correctly identifying those repositories that are not related to the search terms but also leaving the ones that are correctly related to the search terms on the final repositories list.

Table 8 also presents the GQM metrics used to assess the effectiveness of two different filters, Filter V1 and Filter V2. Precision indicates the accuracy of positive predictions made by the filters, with Filter V2 achieving a significantly higher precision of 88.07% compared to Filter V1's 24.55%. Recall measures the ability of the filters to correctly identify positive cases from the total actual positives, where Filter V2 outperforms Filter V1 with a recall of 84.96% versus 79.41%. The F1 score combines precision and recall into a single metric, with Filter V2 achieving a higher score of 86.49% compared to Filter V1's 37.50%. Finally, accuracy measures the overall correctness of the predictions, with Filter V2 demonstrating a substantially higher accuracy of 78.42% compared to Filter V1's 34.78%. These metrics collectively illustrate the significant performance improvements achieved by transitioning from Filter V1 to Filter V2 in insurance data analysis using Python.

5.2.1.5 Research & Python

In the second case study (Table 9), the analysis reveals that Filter V2 demonstrates an improvement in accurately identifying actually positive instances, achieving 55.22%, compared to Filter V1's 8.70%, also displaying progress in reducing false positives from 46.96% on V1 to 0.43% on V2.

Table 9 – Confusion matrix for the second case study

Reseach & Python					
Filter V1			**Filter V2**		
	Actually Positive	**Actually Negative**		**Actually Positive**	**Actually Negative**
Predicted Positive	8,70%	46,96%	**Predicted Positive**	55,22%	0,43%
Predicted Negative	5,65%	38,70%	**Predicted Negative**	40,43%	3,91%
Precision		**15,63%**	**Precision**		**99,22%**
Recall		**60,61%**	**Recall**		**57,73%**
F1		**24,84%**	**F1**		**72,99%**
Accuracy		**47,39%**	**Accuracy**		**59,13%**

However, while progress is observed, challenges persist. True negative from V1 (38.70%) demonstrates a better performance than V2's 3.91% and false negative from V2 (40.43%) is higher than V1's 5.65%.

Overall, The PP + NN values highlight that Filter V2 achieves an accuracy of 59.13%, surpassing Filter V1's 47.39%, presenting an improvement of 11.74% in this case study.

In comparing Filter V1 to Filter V2 based on precision, recall, F1, and accuracy, it is observed that Filter V1 exhibits lower precision, recall, and F1 scores compared to Filter V2. Specifically, Filter V1 demonstrates precision of 15.63%, recall of 60.61%, and F1 score of 24.84%, whereas Filter V2 shows significantly higher precision (99.22%), recall (57.73%), and F1 score (72.99%). Moreover, Filter V2 also surpasses Filter V1 in terms of accuracy, with Filter V2 achieving an accuracy of 59.13% compared to Filter V1's 47.39%.

5.2.1.6 Banking & Java

The analysis of the Banking & Java case study's confusion matrices, depicted in Table 10, shows another improvement of V2 in accurately predicting positive instances, achieving a positive classification rate of 53.41% compared to the 18.47% achieved by Filter V1. Moreover, Filter V2 is effective in minimizing false positives, achieving a low rate of 4.82%, contributing to an overall accuracy of 83.53%. In contrast, Filter V1 exhibits a higher false positive rate at 39.76%.

Table 10 – Confusion matrix for the third case study

Banking & Java					
Filter V1			**Filter V2**		
	Actually Positive	**Actually Negative**		**Actually Positive**	**Actually Negative**
Predicted Positive	18,47%	39,76%	**Predicted Positive**	53,41%	4,82%
Predicted Negative	8,84%	32,93%	**Predicted Negative**	11,65%	30,12%
Precision		**31,72%**	**Precision**		**91,72%**
Recall		**67,65%**	**Recall**		**82,10%**
F1		**43,19%**	**F1**		**86,64%**
Accuracy		**51,41%**	**Accuracy**		**83,53%**

While Filter V2 demonstrates success in positive classifications and overall accuracy, challenges persist in reducing false negatives, as indicated by the 11.65% rate if compared to V1's 8.84%.

Overall, Filter V2 presented an improvement of 32.12% in this case study.

Comparing Filter V1 to Filter V2 based on precision, recall, F1, and accuracy, it is apparent that Filter V2 performs better across all metrics. Filter V1 shows lower precision, recall, and F1 score compared to Filter V2. Specifically, Filter V1 exhibits precision of 31.72%, recall of 67.65%, and F1 score of 43.19%, while Filter V2 demonstrates higher precision (91.72%), recall (82.10%), and F1 score (86.64%). Additionally, Filter V2 achieves a significantly higher accuracy of 83.53%, whereas Filter V1 obtains a lower accuracy of 51.41%.

5.2.1.7 Games & C

The analysis of the Games & C case study's confusion matrices (Table 11) reveals disparities in the performance of Filter V1 and Filter V2. Filter V2 demonstrates an enhancement in accurately predicting positive instances, achieving an 88.24% positive classification rate compared to the 48.32% achieved by Filter V1.

Table 11 – Confusion matrix for the fourth case study

Games & C					
Filter V1			**Filter V2**		
	Actually Positive	**Actually Negative**		**Actually Positive**	**Actually Negative**
Predicted Positive	48,32%	42,02%	**Predicted Positive**	88,24%	2,10%
Predicted Negative	3,78%	5,88%	**Predicted Negative**	7,56%	2,10%
Precision		**53,49%**	**Precision**		**97,67%**
Recall		**92,74%**	**Recall**		**92,11%**
F1		**67,85%**	**F1**		**94,81%**
Accuracy		**54,20%**	**Accuracy**		**90,34%**

Moreover, Filter V2 minimizes false positives, achieving a low rate of 2.10%, while maintaining a true negative rate at 2.10%. In contrast, Filter V1 exhibits a higher false positive rate at 42.02%.

The PP + NN values emphasize that Filter V2 achieves an overall accuracy of 90.34%, a significant improvement over Filter V1's 54.20%, an improvement of 36.10%. While Filter V2 excels in positive classifications and overall accuracy, challenges persist in reducing false negatives, as indicated by the 7.56% rate.

Comparing Filter V1 to Filter V2 based on precision, recall, F1, and accuracy, it is evident that Filter V2 outperforms Filter V1 across all metrics. Filter V1 shows lower precision, recall, and F1 score compared to Filter V2. Specifically, Filter V1 exhibits precision of 53.49%, recall of 92.74%, and F1 score of 67.85%, while Filter V2 demonstrates higher precision (97.67%), recall (92.11%), and F1 score (94.81%). Additionally, Filter

V2 achieves a significantly higher accuracy of 90.34%, whereas Filter V1 obtains a lower accuracy of 54.20%. These results indicate that Filter V2 provides more accurate predictions in the context of games and programming in the C language.

5.2.2 Trends over the Filters comparison

Across all four case studies, trends emerge in the GQM metrics, highlighting that, in each case, Filter V2 consistently exhibits improvements in accurately identifying positive instances compared to Filter V1, being particularly pronounced in positive classification rates, as shown in Table 12.

Table 12 – Mean Average of GQM Metrics

Metric	Mean Everage	
	Filter V1	Filter V2
Precision	31,35%	94,17%
Recall	75,10%	79,22%
F1	43,35%	85,23%
Accuracy	46,95%	77,85%

Starting with precision, Filter V1 exhibits a relatively low precision of 31.35%. This figure indicates that out of all instances predicted as positive by Filter V1, only approximately 31.35% are actually positive. Conversely, Filter V2 showcases a substantially higher precision of 94.17%, implying a significantly better ability to accurately identify positive instances. This stark contrast highlights the superiority of Filter V2 in terms of precision, suggesting a more reliable positive identification compared to Filter V1.

Moving on to recall, Filter V1 demonstrates a recall of 75.10%, indicating that it correctly identifies approximately 75.10% of all actual positive instances. While this figure is relatively high, Filter V2 surpasses it with a slightly higher recall of 79.22%, indicating a marginally better ability to capture positive instances. Despite the subtle difference, this suggests that Filter V2 may be more effective in comprehensively identifying positive instances compared to Filter V1.

The F1 score, which balances precision and recall, provides further insights into the overall performance of each filter. Filter V1 achieves an F1 score of 43.35%, reflecting a moderate overall performance considering both precision and recall. In contrast, Filter V2 attains a significantly higher F1 score of 85.23%, indicating a better balance between precision and recall. This suggests that Filter V2 maintains a more optimal trade-off between accurately identifying positive instances and minimizing misclassifications compared to Filter V1.

Finally, examining accuracy, which denotes the proportion of correctly classified instances out of the total instances, Filter V1 achieves an accuracy of 46.95%. While this figure indicates a relatively low accuracy rate for Filter V1, Filter V2 demonstrates a substantially higher accuracy of 77.85%. This significant disparity underscores the superior overall classification accuracy of Filter V2 compared to Filter V1, further reinforcing its efficacy in accurately classifying instances.

Notably, the atomic and task-oriented prompts employed in Version 2 showcase superior results compared to the broader, with multiple points for analysis prompts in Version 1. This aligns with findings from other researchers in the field (8), reinforcing the efficacy of a more focused and specific approach in filter tasks.

Even though filter the results of V2 suggests a better performance, challenges persist across all domains in mitigating false negatives. In each case, Filter V2 faces difficulties achieving optimal performance in correctly identifying instances that Filter V1 previously misclassified. This common challenge suggests that while language model-based enhancements contribute to improvements in positive classifications and overall accuracy, addressing false negatives remains a complex aspect that requires further attention and fine-tuning.

5.2.3 Insights generation

The following Parts 1 to 5 compose the input prompt sent to the LLM to generate insights about each developer:

> **Part 1:** "The following data presents information about a software developer and the repositories of this software developer on GitHub."

The main idea of Part 1 is to present the data and contextualize the LLM on what it is looking for.

> **Part 2:** "Write a paragraph containing insights about this software developer, considering that a job recruiter will read the paragraph."

In Part 2, it is given a direct request, positioning how the answer should be written and who it would be read by.

> **Part 3:** "The job recruiter is looking for a person who has worked with projects related to the ${industry} industry and who possesses experience in ${stack} programming language"

Part 3 establishes the constraints of insight generation, telling the industry and the desired technology the LLM should focus on.

> **Part 4:** "You must consider whether the software developer is or is not suitable for composing a team where experience in the ${stack} programming language is mandatory and experience with projects related to the ${industry} industry is preferable."

Here, additional context is provided by elucidating the specific considerations that the LLM should take into account during the analysis. This contextual information aims to enhance the positioning of the job recruiter who will be reading the response.

> **Part 5:** "If the software developer is not a match for these conditions, you can also mention that he or she is not an interesting candidate."

Finally, in Part 5, the LLM is clearly suggested to consider not recommending the subject committer in case their data does not align with the desired industry and technology.

In summary, the final prompt sent to the LLM comprises the amalgamation of Parts 1 to 5, coupled with the essential information of the committer. The following discussion delves into a detailed examination of the outcomes and analysis, exploring the interaction dynamics between the provided prompt description and the LLM within the context of the solution.

5.2.3.1 Insight generation assessment

In the assessment of LLM's insight generation, resulting from the second module that employs LLM that was integrated into DevFinder, our primary objective was to discern instances of "hallucination" (11, 12), where the language model produces outputs that may sound reasonable but lack factual accuracy or fail to trace back to the input data. This evaluation was conducted across the same four contexts considered in the filter evaluation: i) Insurance & Python, ii) Research & Python, iii) Banking & Java, and iv) Games & C.

The dataset, encompassing contributions from 697 committers, was submitted to the insight generation prompt detailed in section 3, with 59 contributors from the Insurance & Python context, 281 from the Research & Python context, 101 from the Banking & Java context, and 256 from the Games & C context.

To assess potential instances of "hallucination" in the Large Language Model's responses, each output from LLM underwent investigation. The methodology for this investigation involved comparing the input data transmitted to the LLM's API and the subsequent responses generated. In this investigative process, two key questions were addressed, with boolean values annotated to facilitate metric generation:

Table 14 – Insights assessment metrics results

Metric	Insurance & Python	Research & Python	Banking & Java	Game & C
Strictly related	93,22%	98,10%	97,57%	95,43%
Not strictly related	6,78%	1,90%	2,43%	4,57%
Correct position matching	95,22%	96,60%	96,27%	96,63%
Incorrect position matching	4,78%	3,40%	3,73%	3,37%

1. Is the insight strictly related to the information passed to the LLM?
2. Does the final analysis regarding the contributor's suitability for the position make sense?

These questions present an effort to attend metrics M4, M5 and M6 and aim to provide not only a quantitative assessment of the LLM's accuracy but also qualitative insights into the potential occurrence of "hallucinatory" outputs in the context of contributor assessments.

After manually annotating the Boolean values to each LLM response for all committers, it was possible to assemble a table to present the final results of the analysis. Hence, Table 14 demonstrates the results for the "hallucination" assessment. To better represent each answer of the analysis, four metrics were defined and presented in Table 13.

Table 13 – Insights assessment metrics definition

Metric	Meaning	Correlation
Strictly related	The insight is strictly related to the information passed to the LLM	Question *a* was tagged as True
Not strictly related	The insight is not strictly related to the information passed to the LLM	Question *a* was tagged as False
Correct position matching	The final analysis regarding the contributor's suitability for the position does make sense	Question *b* was tagged as True
Incorrect position matching	The final analysis regarding the contributor's suitability for the position does not make sense	Question *b* was tagged as False

The data in Table 14 provides insights into the LLM's performance during the "hallucination" assessment across different contexts. There is a noticeable trend with a high percentage of responses falling under the category of strictly related insights (93.22% to 98.10%). This suggests that LLM can connect its responses to the input data, consistently generating relevant and clear answers. Therefore, the percentage of responses marked as not strictly related is low (6.78% to 1.90%).

Looking at the correct position-matching metric, the percentages are consistently high across all contexts (95.22% to 96.63%). This implies that LLM's final analyses regarding contributor suitability generally make sense and align with the input information.

Similarly, the percentage of responses categorized as incorrect position matching is low (4.78% to 3.37%). This suggests that LLM successfully provides contextually appropriate and accurate analyses, minimizing instances where the final assessment doesn't fit the context.

5.2.3.2 Accuracy Zones

We conducted an investigation into the size of input texts to ascertain whether there exists any correlation with the performance of the LLM, particularly in cases of misaligned positions. Given the observed variance in text lengths across prompts, our analysis focused on exploring potential links between the accuracy of LLM responses and input text length.

The main objective was to analyze trends in the LLM input length and their correlation over the generated insights previously tagged with the metrics of "Correct position matching"and "Incorrect position matching". Hence, the defined Accuracy Zones are Trust, Caution, and Suspect, where their names reflect the warning that a job recruiter should have on trusting or suspecting the generated insight.

Therefore, to define each zone, we examined the study cases under investigation to establish the boundaries for each accuracy zone. The character count for each accuracy zone is determined by calculating the mean average across the set of Accuracy Zones for each study case (as shown in Table 15), following the outlined rule:

- **Trust Zone**: Spans from 43 characters to the lowest count where an incorrect position matching occurs. The average ranges from 0 to 872 characters.
- **Caution Zone**: Extends from the lowest count where an incorrect position matching occurred to the highest count where a correct position matching occurred. The average ranges from 873 to 4129 characters.
- **Suspect Zone**: Covers from the highest count where a correct position matching occurred to the highest count where an incorrect position matching occurred. The average ranges from 4130 to 7929 characters.

Given the potential inaccuracies in the insight generation task, the proposed Accuracy Zones could help job recruiters assess the precision of the generated insight. Therefore, the generated insights should be accompanied by a tag indicating the accuracy zone in which it falls. Therefore, the generated insights should be accompanied by a tag

Table 15 – Accuracy Zones Ranges

	Trust		Caution		Suspect	
	Min	Max	Min	Max	Min	Max
Insurance & Python	37	1375	1347	4276	4277	6042
Research & Python	42	895	896	3810	3811	6662
Banking & Java	35	583	584	5088	5089	12701
Games & C	57	635	636	3341	3342	6309
Everage	**43**	**872**	**873**	**4129**	**4130**	**7929**

of the respective accuracy zone where the insight is categorized, being represented on the final JSON (Figure 20) by the field “insightAccuracy”.

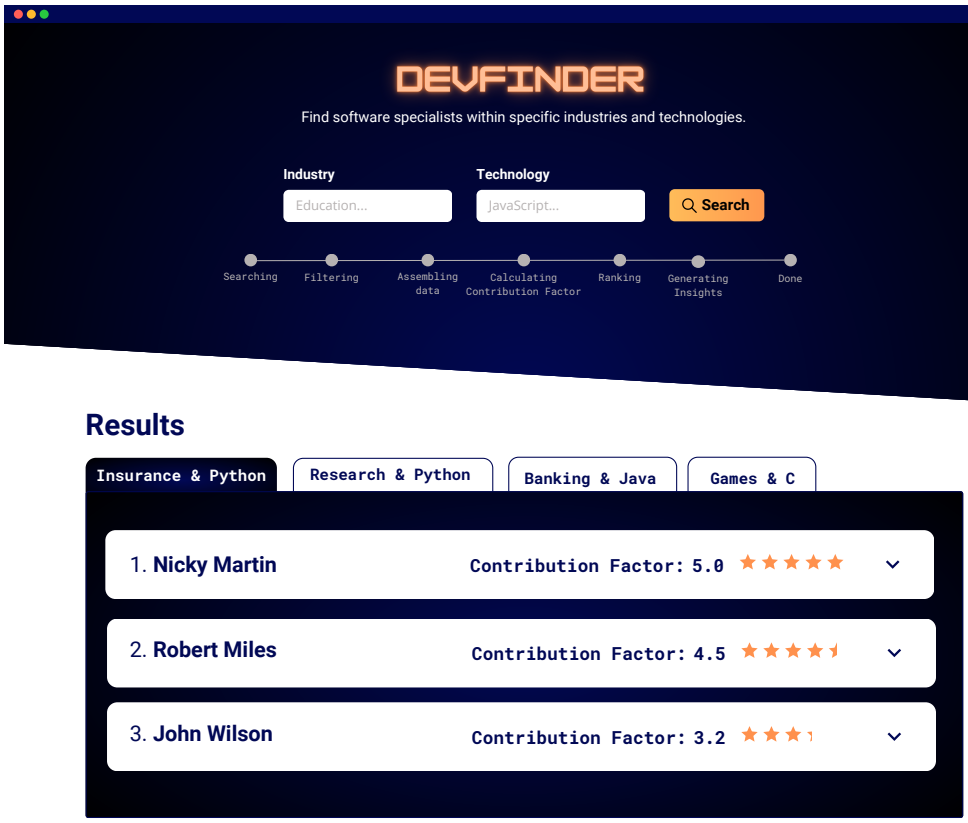
5.2.4 User Interface

Figures 26 and 27 depict a user interface that could leverage DevFinder’s architecture implementation by leveraging GitHub repositories and utilizing LLMs for filtering and insight generation. The figures reflect the system implementation detailed in Section 3, considering the mitigations discussed in Section 4 to address potential downsides. The elements presented on the UI encapsulate the concepts and concerns deliberated throughout this study.

The UI presented in Figure 26 offers a view of the search and result components. Within the search component, users would input their desired industry and technology, clicking the “Search” button to initiate the process outlined in Section 3. Following this, DevFinder executes the specified process, and a new tab appears on the Result component, allowing users to access the final recommendation list of software developers.

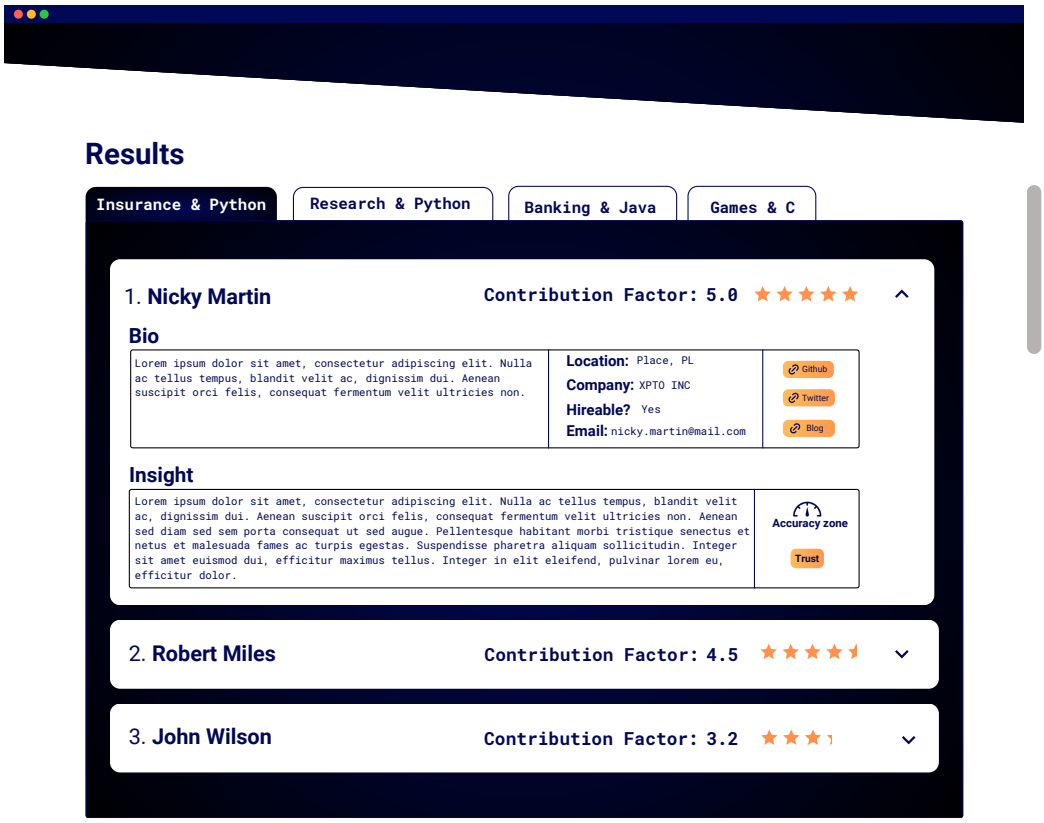
Users would encounter developers arranged by Contribution Factor (CF) in the tab displaying the resultant recommendation list. Each developer’s name is accompanied by the CF value, conveniently represented with star icons, facilitating user experience and visual interaction with the Contribution Factor, a visual representation of the data presented on the final JSON of Figure 20

– Figure 26 - Searching elements of DevFinder's user interface



When users click on a recommended developer, additional information is revealed. Users can view the developer's name, bio, the generated insight accompanied by the corresponding Accuracy Zone, the Contribution Factor, and various details sourced from the developer's GitHub account. This includes information such as location, company, hireability status, email, and buttons providing direct links to the developer's GitHub, Twitter, and Blog profiles. Figure 27 illustrates the UI that portrays the described elements.

– Figure 27 - Results elements of DevFinder's user interface



5.2.5 Generated Insights

To offer a tangible glimpse into the insights generated by the integrated LLM in the outlined system, we present two real examples generated by the LLM integration through data of real GitHub users, as outlined previously. The first example is associated with the Banking & Java study case, highlighting a positive alignment between the recruiter's criteria and the developer's skills.

Based on the data provided, this software developer seems to be a suitable candidate for a team where experience in the Java programming language is mandatory and experience with projects related to the banking industry is preferable. The developer has extensive experience with Java, as shown by their repositories such as “keystore-management”, “open-banking-gateway”, and “secure-token-service”. These projects indicate that the developer has worked with Java in the context of the banking industry and has experience implementing OAuth 2.0 Token Exchange. Additionally, their repository “spring-boot” demonstrates familiarity with the popular Java framework for building enterprise Java applications. Overall, this software developer aligns with the job recruiter’s requirements and would be a valuable asset to a team requiring Java expertise in the banking industry.

The generated insight offers key information for job recruiters to assess the software developer’s skills and suitability for a position. Firstly, it emphasizes the developer’s extensive experience with the Java programming language, evident from specific repositories like “keystore-management”, “open-banking-gateway”, and “secure-token-service”. These projects indicate a solid background in Java development.

Furthermore, the insight underscores the developer’s involvement in projects related to the banking industry, providing assurance to recruiters about the candidate’s domain knowledge. This aligns with the recruiter’s preference for experience in the banking sector.

The mention of specific projects, such as the implementation of OAuth 2.0 Token Exchange in the banking context, showcases the developer’s technical capabilities. Recruiters can infer the candidate’s proficiency in security protocols and industry-specific technologies. Additionally, the presence of the “spring-boot” repository signals the developer’s familiarity with Spring Boot, a widely used Java framework for building enterprise applications. This indicates the candidate’s ability to work with industry-standard frameworks.

On the other hand, the subsequent example illustrates a developer who is not well-suited for the open position, specifically associated with the Research & Python study case.

Based on the given data, this software developer may not be suitable for a team where experience in the Python programming language is mandatory and experience with projects related to the research industry is preferable. While the developer does have some experience with Python, as seen in repositories such as “atg” and “baselines”, their overall GitHub activity indicates a diverse set of programming languages being used. The repositories range from Arduino-based projects to PHP, JavaScript, and even HTML. Additionally, there is no clear evidence of projects specifically related to the research industry. Therefore, it is recommended that the job recruiter considers other candidates who have a stronger focus on Python and relevant experience in the research industry.

The conclusion suggests that the developer may not be the most suitable candidate for a team where expertise in the Python programming language is mandatory, and experience with projects related to the research industry is preferable.

While the developer does possess some experience with Python, demonstrated in repositories like “atg” and “baselines”, a comprehensive review of their overall GitHub activity reveals a diverse set of programming languages being utilized. The repositories encompass a spectrum from Arduino-based projects to PHP, JavaScript, and HTML. This diversity indicates the developer’ s versatility but may raise concerns regarding a focused proficiency in Python.

As a result, the job recruiter is recommended to explore other candidates who exhibit a more concentrated focus on Python and possess relevant experience in the research industry.

5.2.6 Threats to validity

Regarding the network and the Contribution Factor analysis that was discussed, some potential threats warrant consideration. External validity may be compromised as the application of DevFinder is limited to specific industry domains (Education, Oil, Finance) and technologies (PHP, JavaScript, Python), potentially hindering the generalizability of findings to broader software development contexts. Sampling bias is another concern, as the chosen domains and technologies may not be representative of the entire software development landscape. Algorithmic biases within DevFinder’s algorithms may influence committer selection and ranking, impacting the study’s overall validity. Additionally, assumptions related to metric normalization and the chosen graph representation may not universally hold true. It is essential to recognize these potential threats and conduct thorough sensitivity analyses, considering alternative explanations to fortify the robustness of the study’s conclusions.

Similarly, the integration of Large Language Models into systems that recommend

software developers, as explored in this study, is not without its challenges. Existing literature has highlighted concerns such as the potential for “hallucination” (12, 13), safety and robustness issues (15), non-discrimination and fairness issues (16, 17), explainability challenges (18), and privacy considerations (19, 20). Addressing these challenges is paramount to ensuring LLMs’ responsible and ethical deployment in decision-support contexts. Therefore, we recognize the following threats to the validity of the current version of our study.

Firstly, during the evaluation processes for both the filter and insight generation, the authors engaged in the subjective task of categorizing data and assigning boolean values to assess the effectiveness of the LLM. This evaluative process involved interpreting diverse information from different industry contexts, some of which fell outside the authors’ complete domain expertise. Consequently, the classifications made during this process may carry a degree of subjectivity, introducing potential variations in the presented accuracy. Recognizing and acknowledging this subjectivity is crucial for comprehensively understanding the study’s outcomes.

Secondly, there is a lack of direct evaluation with industry professionals. While repositories may contain valuable information, they may not fully capture the intricacies and nuances of real-world decision-making scenarios. Therefore, the absence of input from industry experts could limit the study’s findings’ applicability and generalizability to real-world contexts.

Furthermore, there is a potential limitation of this study is the use of a limited number of contexts to generate results. By considering four different contexts, the study may not have captured the full breadth and diversity of decision-making environments. Different contexts can introduce unique challenges and considerations that may influence the performance and effectiveness of systems that assist in decision-making.

As this study employs generative artificial intelligence (AI), there is inherent variability in the results obtained from different runs of the model. Generative AI systems, including neural networks, may produce different outputs each time they are run, even when provided with the same input data. Researchers should be mindful of this variability and consider its potential impact on the interpretation and generalizability of the results.

Lastly, the evaluation of results in this study relied on the analysis of data by individuals referred to as "committers."However, these individuals did not participate in the analysis of their data. This lack of involvement from the data analysts themselves introduces a potential threat to the validity of the study’s findings. Without direct input from the individuals responsible for generating and analyzing the data, there may be a risk of overlooking important insights or nuances that could affect the interpretation of the results.

5.3 Final Remarks of The Chapter

The outcomes of this work address our central research question: *How can DevFinder support the search for software experts whose knowledge aligns specific industry domains with specific technologies?*

In addressing this question, we integrated a Large Language Model into an existing system that recommends software developers, leveraging its advantages across two distinct scenarios: the filtering process and insight generation.

Our case studies, focusing on the downsides of LLM integration within the system, revealed that, on average, the filter performed well in 77.85% of cases. Additionally, the insight generation process proved effective when complemented by the use of the proposed Accuracy Zones.

Despite initial concerns about the accuracy, potential "hallucination", and reliability of the LLM within the integration, we found that these factors did not significantly impact the effectiveness of the recommendation tasks within the solution, once mitigated by the presence of the proposed Accuracy Zones.

The wireframes presented provide a visual representation of the DevFinder user interface, representing the search and result components. These wireframes encapsulate the concepts and mitigations discussed throughout the study, providing users with a seamless experience in searching and accessing the final list of software developer recommendations.

The insights generated, illustrated through real examples, demonstrate the practical application of the information generated by LLM for recruiters. The positive alignment between a developer's skills and a recruiter's criteria in the Banking & Java case, as well as the recommendation to explore other candidates in the Research & Python case, demonstrate the system's ability to assist decision-making processes effectively. Going forward, LLM integration into DevFinder presents a valuable tool to increase talent acquisition efficiency in the software development domain. The proposed Accuracy Zones act as a safeguard, addressing potential drawbacks and ensuring the reliability of recommendations.

This chapter provides an evaluation of DevFinder's recommendation generation, focusing on complex network analysis and the integration of Large Language Models (LLMs). The analysis covered diverse industry domains and technologies, affirming DevFinder's efficacy in identifying specialists.

6 Conclusion

This work presents a body of knowledge regarding key concepts about systems that recommend software developers, ontologies, complex networks, and Large Language Models (LLMs). The work begins by introducing the domain of job recruitment scenarios and identifying the challenges related to leveraging version control systems data for decision-making in the recruitment process of high-specialty software developers.

Once finding software specialists poses a challenging task, this work proposed the answer for the following research question: ***"How can DevFinder support the search for software experts whose knowledge aligns specific industry domains with specific technologies?"*** The proposed architecture, DevFinder, was developed intending to support job recruiters in finding software by providing a ranked recommendation list of software specialists within input conditions.

Case studies were conducted to assess the proposed architecture, and evidence was presented on the feasibility of supporting decision-making. The case studies shed light on the downsides and advantages of integrating an LLM into the system. Despite initial concerns about accuracy and potential "hallucination", the findings indicate that, when complemented by proposed Accuracy Zones, the LLM improves the filtering process and insight generation.

As theoretical knowledge was produced throughout the study, it stands out subjects such as ontologies, complex networks, Large Language Models, and Machine Learning. When talking about technical and scientific knowledge produced, the following contributions highlight the value of this work:

- A systematic mapping study (SMS) that analyzes 1251 studies revealing the historical trajectory of the field and guiding future progress. Our research provides insights into current industry practices, highlighting key technologies and data sources. In addition, the SMS stands out by addressing the unique challenge of recommending software developers with expertise in both specific technologies and diverse industry domains. Lastly, we present a key factors list that serves as a practical guide for emerging researchers, offering essential guidelines and tools to design effective systems that recommend software developers;
- The proposal of an architecture model to extract, filter, rank, and generate insights about software developers from real-world repositories in order to support stakeholders of job recruitment tasks;
- The creation of a Contribution Factor metric that leverages collaboration aspects of software developers to generate a comprehensive ranked list;

- The assessment of the interaction between the proposed Contribution Factor to other classic metrics, such as closeness and degree centrality;
- The integration of Large Language Models (LLMs) on the system to perform tasks of filtering and insights generation.
- The assessment of the LLM integration through four case studies, considering real-world scenarios that provided data to verify the accuracy and precision of the LLM integration to the architecture;
- The implementation of the proposed architecture, capable of extracting software developers from repositories, filtering them using an LLM, ranking them using complex networks, enriching the presented data to stakeholders using an LLM, and providing data to the visualization of end users.

As a result from the scientific and technological contributions of this work, we have published a systematic mapping (138), the first DSR cycle of DevFinder model (136), the second DSR cycle of DevFinder with the complex network model (137), and the third DSR cycle considering the LLM integration to filter and generate insights (currently on submission process of the Information and Software Technology journal).

The results of this research have significant relevance in the rapidly evolving landscape of talent acquisition in the software development domain. Integrating Large Language Models (LLMs) into the DevFinder architecture shows promising results in addressing the challenges associated with identifying software experts aligned with specific industry domains and technologies. The iterative study development process and evaluation through diverse industry case studies contribute to advancing our understanding of the effectiveness of LLMs in systems that recommend software developers. The demonstrated success rate of 77.85% in the filtering process and the introduction of Accuracy Zones offer valuable insights for recruiters, providing a structured framework to evaluate the reliability of the insights generated. The improvements in filtering and generating insights, highlighted by the new iteration of DevFinder, mean a practical and effective tool for improving talent acquisition processes. These findings are critical for organizations looking for innovative solutions to streamline and optimize their recruitment strategies in the competitive software development landscape.

For future work, we intend to refine the filtering process further to address challenges related to false negatives, increasing the model's accuracy in distinguishing repositories relevant to specific industry domains. Furthermore, exploring alternative or complementary models and techniques for generating insights could contribute to a more comprehensive and robust architecture. Further investigation into the explainability and interpretability of LLMs in the context of recruitment decision-making could address concerns related to the "black box" nature of these models. As a way to enrich and refine data, the possibility

of employing ontology for semantic analysis aligned with the use of LLMs could be an interesting topic to be explored.

The context and a day-by-day usage of the proposed architecture could be assessed as a way to collect feedback from users on real-world scenarios. By doing so, we would also be able to evaluate the architecture performance across a broader range of industry domains and technologies. Also,incorporating feedback loops with recruiters to iteratively improve the system based on real-world application experiences is another avenue for future research.

The proposed future work could inspire researchers to further enhance architectural capabilities and tackle emerging challenges, driving the ongoing evolution of talent acquisition technologies within the software development industry.

REFERENCES

1 FÄLL, Patrick and HAUSER, Matthias and THIESSE, Frän and others. **Identifying the skills expected of IS graduates by industry: A text mining approach**. 2018.

2 JONES, Jack William. **Management Decision Support Systems**. JSTOR, 1980.

3 TERRIBILE, Fabio, AGRILLO, Antonietta, BONFANTE, A., BUSCEMI, Gilda, COLANDREA, M., D'ANTONIO, A., DE MASCELLIS, R., DE MICHELE, C., LANGELLA, G., MANNA, P., et al. **A Web-based spatial decision supporting system for land management and soil conservation**. *Solid Earth*, vol. 6, no. 3, pp. 903–928, 2015. Copernicus GmbH.

4 CAMPION, Michael C., CAMPION, Michael A., CAMPION, Emily D., and REIDER, Matthew H. **Initial Investigation into Computer Scoring of Candidate Essays for Personnel Selection**. *Journal of Applied Psychology*, vol. 101, no. 7, pp. 958, 2016. American Psychological Association.

5 GAO, Yunfan, SHENG, Tao, XIANG, Youlin, XIONG, Yun, WANG, Haofen, and ZHANG, Jiawei. **Chat-rec: Towards Interactive and Explainable LLMS-Augmented Recommender System**. arXiv preprint arXiv:2303.14524, 2023.

6 ZHANG, Junjie, XIE, Ruobing, HOU, Yupeng, ZHAO, Wayne Xin, LIN, Leyu, and WEN, Ji-Rong. **Recommendation as Instruction Following: A Large Language Model Empowered Recommendation Approach**. arXiv preprint arXiv:2305.07001, 2023.

7 KALLIAMVAKOU, Eirini, GOUSIOS, Georgios, BLINCOE, Kelly, SINGER, Leif, GERMAN, Daniel, and DAMIAN, Daniela. **The Promises and Perils of Mining GitHub (Extended Version)**. *Empirical Software Engineering*, 2015, January.

8 FAN, Wenqi, ZHAO, Zihuai, LI, Jiatong, LIU, Yunqing, MEI, Xiaowei, WANG, Yiqi, TANG, Jiliang, and LI, Qing. **Recommender Systems in the Era of Large Language Models (LLMS)**. arXiv preprint arXiv:2307.02046, 2023.

9 LIU, Junling, LIU, Chao, LV, Renjie, ZHOU, Kang, and ZHANG, Yan. **Is ChatGPT a Good Recommender? A Preliminary Study**. arXiv preprint arXiv:2304.10149, 2023.

10 ZHIYULI, Aakas, CHEN, Yanfang, ZHANG, Xuan, and LIANG, Xun. **BookGPT: A General Framework for Book Recommendation Empowered by Large Language Model**. arXiv preprint arXiv:2305.15673, 2023.

11 MANAKUL, Potsawee, LIUSIE, Adian, and GALES, Mark JF. **Selfcheckgpt: Zero-resource Black-box Hallucination Detection for Generative Large Language Models**. arXiv preprint arXiv:2303.08896, 2023.

12 MCKENNA, N., LI, T., CHENG, L., HOSSEINI, MJ, JOHNSON, M., and STEEDMAN, M. **Sources of Hallucination by Large Language Models on Inference Tasks (2023)**. arXiv preprint arXiv:2305.14552, 2023.

13 JI, Ziwei, LEE, Nayeon, FRIESKE, Rita, YU, Tiezheng, SU, Dan, XU, Yan, ISHII, Etsuko, BANG, Ye Jin, MADOTTO, Andrea, and FUNG, Pascale. **Survey of Hallucination in Natural Language Generation**. *ACM Computing Surveys*, vol. 55, no. 12, pp. 1–38, 2023. ACM New York, NY.

14 ZHANG, Zhen, ZHANG, Guanhua, HOU, Bairu, FAN, Wenqi, LI, Qing, LIU, Sijia, ZHANG, Yang, and CHANG, Shiyu. **Certified Robustness for Large Language Models with Self-Denoising**. arXiv preprint arXiv:2307.07171, 2023.

15 ZHUO, TY, HUANG, Y, CHEN, C, and XING, Z. **Exploring AI Ethics of ChatGPT: A Diagnostic Analysis. ArXiv**. arXiv preprint arXiv:2301.12867, 2023.

16 ZHANG, Guanhua, ZHANG, Yihua, ZHANG, Yang, FAN, Wenqi, LI, Qing, LIU, Sijia, and CHANG, Shiyu. **Fairness Reprogramming**. *Advances in Neural Information Processing Systems*, vol. 35, pp. 34347–34362, 2022.

17 LIU, Haochen, DACON, Jamell, FAN, Wenqi, LIU, Hui, LIU, Zitao, and TANG, Jiliang. **Does Gender Matter? Towards Fairness in Dialogue Systems**. arXiv preprint arXiv:1910.10486, 2019.

18 BILLS, Steven, CAMMARATA, Nick, MOSSING, Dan, TILLMAN, Henk, GAO, Leo, GOH, Gabriel, SUTSKEVER, Ilya, LEIKE, Jan, WU, Jeff, and SAUNDERS, William. **Language Models Can Explain Neurons in Language Models**. URL `https://openaipublic.blob.core.windows.net/neuron-explainer/paper/index.html` (Date accessed: 14.05.2023), 2023.

19 CARLINI, Nicholas, TRAMER, Florian, WALLACE, Eric, JAGIELSKI, Matthew, HERBERT-VOSS, Ariel, LEE, Katherine, ROBERTS, Adam, BROWN, Tom, SONG, Dawn, ERLINGSSON, Ulfar, et al. **Extracting Training Data from Large Language Models**. In *30th USENIX Security Symposium (USENIX Security 21)*, pp. 2633–2650, 2021.

20 LI, Yansong, TAN, Zhixing, and LIU, Yang. **Privacy-preserving Prompt Tuning for Large Language Model Services**. arXiv preprint arXiv:2305.06212, 2023.

21 FAN, Wenqi, MA, Yao, LI, Qing, WANG, Jianping, CAI, Guoyong, TANG, Jiliang, and YIN, Dawei. **A Graph Neural Network Framework for Social Recommendations**. *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 5, pp. 2033–2047, 2020. IEEE.

22 DIJKMAN, Remco M., DUMAS, Marlon, and OUYANG, Chun. **Semantics and Analysis of Business Process Models in BPMN**. *Information and Software Technology*, vol. 50, no. 12, pp. 1281–1294, 2008. Elsevier.

23 ROSEMANN, Michael and VOM BROCKE, Jan. **The Six Core Elements of Business Process Management**. In *Handbook on Business Process Management 1: Introduction, Methods, and Information Systems*, pp. 105–122, 2014. Springer.

24 PEARSON, Karl. **VII. Mathematical Contributions to the Theory of Evolution.—III. Regression, Heredity, and Panmixia**. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, no. 187, pp. 253–318, 1896. The Royal Society London.

25 WATTS, Duncan J. and STROGATZ, Steven H. **Collective Dynamics of 'Small-World' Networks**. *Nature*, vol. 393, no. 6684, pp. 440–442, 1998. Nature Publishing Group.

26 BARABÁSİ, Albert-László and ALBERT, Réka. **Emergence of Scaling in Random Networks**. *Science*, vol. 286, no. 5439, pp. 509–512, 1999. American Association for the Advancement of Science.

27 DIESTEL, R. **Graph Theory**. Electronic Library of Mathematics. ISBN 9783540261834, LCCN 2005928165. URL `https://books.google.com.br/books?id=aR2TMYQr2CMC`, 2005. Springer.

28 ASHISH, Vaswani, NOAM, Shazeer, NIKI, Parmar, JAKOB, Uszkoreit, LLION, Jones, AIDAN, N. Gomez, LUKASZ, Kaiser, and ILLIA, Polosukhin. **Attention Is All You Need**. arXiv preprint arXiv:1706.03762, 2023.

29 JACOB, Devlin, MING-WEI, Chang, KENTON, Lee, and KRISTINA, Toutanova. **BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding**. arXiv preprint arXiv:1810.04805, 2019.

30 BROWN, Tom, MANN, Benjamin, RYDER, Nick, SUBBIAH, Melanie, KAPLAN, Jared D., DHARIWAL, Prafulla, NEELAKANTAN, Arvind, SHYAM, Pranav, SASTRY, Girish, ASKELL, Amanda, et al. **Language models are few-shot learners**. *Advances in Neural Information Processing Systems*, vol. 33, pp. 1877–1901, 2020.

31 BENDER, Emily M., GEBRU, Timnit, MCMILLAN-MAJOR, Angelina, and SHMITCHELL, Shmargaret. **On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?** In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency (FAccT '21)*, pp. 610–623, 2021. Association for Computing Machinery. New York, NY, USA. DOI: `10.1145/3442188.3445922`. URL: `https://doi.org/10.1145/3442188.3445922`.

32 CUMMINGS, Maeve and MCCUBBREY, Donald J. **Management Information Systems for the Information Age**. 2004. McGraw-Hill Irwin.

33 HAN, Jiawei, PEI, Jian, and TONG, Hanghang. **Data Mining: Concepts and Techniques**. 2022. Morgan Kaufmann.

34 LAUDON, Kenneth C and LAUDON, Jane Price. **Management Information Systems: Managing the Digital Firm**. 2004. Pearson Educación.

35 POWER, Daniel J. **A Brief History of Decision Support Systems**. *DSSResources.com*, vol. 3, 2007.

36 BERGER, James. **Statistical Decision Theory: Foundations, Concepts, and Methods**. 2013. Springer Science & Business Media.

37 RAIFFA, Howard and SCHLAIFER, Robert, et al. **Applied Statistical Decision Theory**. 1961. Wiley New York.

38 ARONSON, Jay E, LIANG, Ting-Peng, and MACCARTHY, Richard V. **Decision Support Systems and Intelligent Systems**. vol. 4, 2005. Pearson Prentice-Hall Upper Saddle River, NJ, USA.

39 ZHANG, Zhenyu, SUN, Hailong, and ZHANG, Hongyu. **Developer Recommendation for Topcoder Through a Meta-Learning Based Policy Model**. *Empirical Software Engineering*, vol. 25, pp. 859–889, 2020. Springer.

40 LIU, Xiaomo, WANG, G Alan, JOHRI, Aditya, ZHOU, Mi, and FAN, Weiguo. **Harnessing global expertise: A comparative study of expertise profiling methods for online communities**. *Information Systems Frontiers*, vol. 16, pp. 715–727, 2014. Springer.

41 LI, Liangyue, TONG, Hanghang, CAO, Nan, EHRLICH, Kate, LIN, Yu-Ru, and BUCHLER, Norbou. **Replacing the Irreplaceable: Fast Algorithms for Team Member Recommendation**. In *Proceedings of the 24th International Conference on World Wide Web*, pp. 636–646, 2015.

42 DATTA, Anwitaman, YONG, Jackson Tan Teck, and BRAGHIN, Stefano. **The Zen of Multidisciplinary Team Recommendation**. *Journal of the Association for Information Science and Technology*, vol. 65, no. 12, pp. 2518–2533, 2014. Wiley Online Library.

43 AZMEH, Zeina and MIRBEL, Isabelle. **Identifying Reputable Contributors in Collaborative Software Development Platforms**. In *2015 IEEE 9th International Conference on Research Challenges in Information Science (RCIS)*, pp. 374–381, 2015. IEEE.

44 HORTA, Vitor, STRÖELE, Victor, SCHETTINO, Vinicius, OLIVEIRA, Jonice, DAVID, José Maria, and ARAÚJO, Marco Antônio P. **Collaboration Analysis in Global Software Development**. In *2019 IEEE 23rd International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pp. 464–469, 2019. IEEE.

45 OLIVEIRA JR, Marcio, BRAGA, Regina, GHIOTTO, Gleiph, and DAVID, José. **Recommending External Developers to Software Projects Based on Historical Analysis of Previous Contributions**. In *Proceedings of the XXXIII Brazilian Symposium on Software Engineering*, pp. 417–426, 2019.

46 TUAROB, Suppawong, ASSAVAKAMHAENGHAN, Noppadol, TANAPHANTARUK, Waralee, SUWANWORABOON, Ponlakit, HASSAN, Saeed-Ul, and CHOETKIERTIKUL, Morakot. **Automatic Team Recommendation for Collaborative Software Development**. *Empirical Software Engineering*, vol. 26, no. 4, pp. 64, 2021. Springer.

47 CONSTANTINO, Kattiana and FIGUEIREDO, Eduardo. **CoopFinder: Finding Collaborators Based on Co–Changed Files**. In *2022 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pp. 1–3, 2022. IEEE.

48 ROSTAMI, Peyman and SHAKERY, Azadeh. **A Deep Learning-Based Expert Finding Method to Retrieve Agile Software Teams from CQAs**. *Information Processing & Management*, vol. 60, no. 2, pp. 103144, 2023. Elsevier.

49 VERGNE, Matthieu and SUSI, Angelo. **Expert Finding Using Markov Networks in Open Source Communities**. In *Advanced Information Systems Engineering: 26th International Conference, CAiSE 2014, Thessaloniki, Greece, June 16-20, 2014. Proceedings 26*, pp. 196–210, 2014. Springer.

50 LOPES, Tales, STRÖELE, Victor, BRAGA, Regina, DAVID, José Maria N, and BAUER, Michael. **A Broad Approach to Expert Detection Using Syntactic and Semantic Social Networks Analysis in the Context of Global Software Development**. *Journal of Computational Science*, vol. 66, pp. 101928, 2023. Elsevier.

51 HAUFF, Claudia and GOUSIOS, Georgios. **Matching GitHub Developer Profiles to Job Advertisements**. In *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*, pp. 362–366, 2015. IEEE.

52 FU, Chenbo, ZHOU, Mingming, XUAN, Qi, and HU, Hong-Xiang. **Expert Recommendation in OSS Projects Based on Knowledge Embedding**. In *2017 International Workshop on Complex Systems and Networks (IWCSN)*, pp. 149–155, 2017. IEEE.

53 DEHGHAN, Mahdi, RAHMANI, Hossein Ali, ABIN, Ahmad Ali, and VU, Viet-Vu. **Mining Shape of Expertise: A Novel Approach Based on Convolutional Neural Network**. *Information Processing & Management*, vol. 57, no. 4, pp. 102239, 2020. Elsevier.

54 XIE, Xinqiang, YANG, Xiaochun, WANG, Bin, and HE, Qiang. **DevRec: Multi-Relationship Embedded Software Developer Recommendation**. *IEEE Transactions on Software Engineering*, vol. 48, no. 11, pp. 4357–4379, 2021. IEEE.

55 MONTANDON, Joao Eduardo, SILVA, Luciana Lourdes, and VALENTE, Marco Tulio. **Identifying Experts in Software Libraries and Frameworks Among GitHub Users**. In *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*, pp. 276–287, 2019. IEEE.

56 ROSTAMI, Peyman and NESHATI, Mahmood. **Intern Retrieval from Community Question Answering Websites: A New Variation of Expert Finding Problem**. *Expert Systems with Applications*, vol. 181, pp. 115044, 2021. Elsevier.

57 MORADI DAKHEL, Arghavan, DESMARAIS, Michel C., and KHOMH, Foutse. **Assessing Developer Expertise from the Statistical Distribution of Programming Syntax Patterns**. In *Evaluation and Assessment in Software Engineering*, pp. 90–99, 2021.

58 PEREZ, Quentin, URTADO, Christelle, and VAUTTIER, Sylvain. **Mining Experienced Developers in Open-Source Projects**. In *ENASE 2022-17th International Conference on Evaluation of Novel Approaches to Software Engineering*, pp. 443–452, 2022. SCITEPRESS-Science and Technology Publications.

59 ATZBERGER, Daniel, SCORDIALO, Nico, CECH, Tim, SCHEIBEL, Willy, TRAPP, Matthias, and DÖLLNER, Jürgen. **CodeCV: Mining Expertise of GitHub Users from Coding Activities**. In *2022 IEEE 22nd International Working Conference on Source Code Analysis and Manipulation (SCAM)*, pp. 143–147, 2022. IEEE.

60 HAYMOND, Shannon and MASTER, Stephen R. **How Can We Ensure Reproducibility and Clinical Translation of Machine Learning Applications in Laboratory Medicine?**. *Clinical Chemistry*, vol. 68, no. 3, pp. 392–395, 2022. Oxford University Press.

61 HERBSLEB, James D. and MOITRA, Deependra. **Global Software Development**. *IEEE Software*, vol. 18, no. 2, pp. 16–20, 2001. IEEE.

62 Rubin, Julia and Rinard, Martin. **The Challenges of Staying Together While Moving Fast: An Exploratory Study**. In *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)*, pp. 982–993, 2016. IEEE. doi:10.1145/2884781.2884871.

63 LOPES, Tales, STRÖELE, Victor, BRAGA, Regina, and BAUER, Michael. **Unraveling the Semantic Evolution of Core Nodes in a Global Contribution Network**. In *2020 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pp. 594–601, 2020. IEEE.

64 Wang, Xianzhi, Huang, Chaoran, Yao, Lina, Benatallah, Boualem, and Dong, Manqing. **A Survey on Expert Recommendation in Community Question Answering**. *Journal of Computer Science and Technology*, vol. 33, no. 4, pp. 625-653, Jul. 2018. ISSN 1860-4749. doi:10.1007/s11390-018-1845-0. Available online: `https://doi.org/10.1007/s11390-018-1845-0`.

65 RICH, Elaine. **User Modeling via Stereotypes**. *Cognitive Science*, vol. 3, no. 4, pp. 329–354, 1979. Elsevier.

66 SANDERSON, Mark and CROFT, W. Bruce. **The History of Information Retrieval Research**. *Proceedings of the IEEE*, vol. 100, Special Centennial Issue, pp. 1444–1451, 2012. IEEE.

67 SHANI, Guy and GUNAWARDANA, Asela. **Evaluating Recommendation Systems**. In *Recommender Systems Handbook*, pp. 257–297, 2011. Springer.

68 PETERSEN, Kai, VAKKALANKA, Sairam, and KUZNARZ, Ludwik. **Guidelines for conducting systematic mapping studies in software engineering: An update**. *Information and Software Technology*, vol. 64, pp. 1–18, 2015. Elsevier.

69 HYRYNSALMI, Sonja M., RANTANEN, Minna M., and HYRYNSALMI, Sami. **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)*, pp. 1-10, 2021. doi:10.1109/ICE/ITMC52061.2021.9570207.

70 KITCHENHAM, Barbara. **Procedures for Performing Systematic Reviews**. *Keele, UK, Keele Univ.*, vol. 33, 2004.

71 KITCHENHAM, Barbara and CHARTERS, S. **Guidelines for performing Systematic Literature Reviews in Software Engineering**, 2007.

72 KITCHENHAM, Barbara A., BUDGEN, David, and BRERETON, Pearl. **Using mapping studies as the basis for further research - A participant-observer case study**. *Information Software Technology*, vol. 53, pp. 638-651, 2011. doi:10.1016/j.infsof.2010.12.011.

73 STEINMACHER, Igor, CHAVES STEINMACHER, Ana Paula, and GEROSA, Marco Aurelio. **Awareness Support in Distributed Software Development: A Systematic Review and Mapping of the Literature**. *Computer Supported Cooperative Work (CSCW)*, vol. 22, 2013. doi:10.1007/s10606-012-9164-4.

74 MOURÃO, Erica, KALINOWSKI, Marcos, MURTA, Leonardo, MENDES, Emilia, and WOHLIN, Claes. **Investigating the use of a hybrid search strategy for systematic reviews**. In *2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pp. 193–198, 2017. IEEE.

75 WOHLIN, Claes, KALINOWSKI, Marcos, FELIZARDO, Katia Romero, and MENDES, Emilia. **Successful combination of database search and snowballing for identification of primary studies in systematic literature studies**. *Information and Software Technology*, vol. 147, pp. 106908, 2022. Elsevier.

76 PETTICREW, Mark and ROBERTS, Helen. **Systematic reviews in the social sciences: A practical guide**. John Wiley & Sons, 2008.

77 ZHANG, He and ALI BABAR, Muhammad. **On Searching Relevant Studies in Software Engineering**. In *Proceedings of the 14th International Conference on Evaluation and Assessment in Software Engineering*, pp. 111–120, 2010. UK, EASE'10.

78 LIU, Xiaomo, WANG, G Alan, JOHRI, Aditya, ZHOU, Mi, and FAN, Weiguo. **Harnessing global expertise: A comparative study of expertise profiling methods for online communities**. *Information Systems Frontiers*, vol. 16, pp. 715–727, 2014. Springer.

79 WOHLIN, Claes. **Guidelines for snowballing in systematic literature studies and a replication in software engineering**. In *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*, pp. 1–10, 2014.

80 VADLAMANI, Sri Lakshmi and BAYSAL, Olga. **Studying Software Developer Expertise and Contributions in Stack Overflow and GitHub**. In *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pp. 312-323, 2020. doi:10.1109/ICSME46990.2020.00038.

81 MOHEBZADA, Jamshaid G, RUHE, Guenther, and EBERLEIN, Armin. **Systematic mapping of recommendation systems for requirements engineering**. In *2012 International Conference on Software and System Process (ICSSP)*, pp. 200–209, 2012. IEEE.

82 BRASIL-SILVA, Renata and SIQUEIRA, Fábio Levy. **Metrics to quantify software developer experience: a systematic mapping**. In *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing*, pp. 1562–1569, 2022.

83 COSTA, Alexandre, RAMOS, Felipe, PERKUSICH, Mirko, DANTAS, Emanuel, DILORENZO, Ednaldo, CHAGAS, Ferdinandy, MEIRELES, André, ALBUQUERQUE, Danyllo, SILVA, Luiz, ALMEIDA, Hyggo, et al. **Team formation in software engineering: a systematic mapping study**. *IEEE Access*, vol. 8, pp. 145687–145712, 2020. IEEE.

84 PAPOUTSOGLU, Maria, AMPATZOGLU, Apostolos, MITTAS, Nikolaos, and ANGELIS, Lefteris. **Extracting knowledge from on-line sources for software engineering labor market: A mapping study**. *IEEE Access*, vol. 7, pp. 157595–157613, 2019. IEEE.

85 SAYAGO-HEREDIA, Jaime, PÉREZ-CASTILLO, Ricardo, and PIATTINI, Mario. **A Systematic Mapping Study on Analysis of Code Repositories**. *Informatica*, vol. 32, no. 3, pp. 619–660, 2021. Vilnius University.

86 TUKUR, Muhammad, UMAR, Sani, and HASSINE, Jameleddine. **Requirement engineering challenges: A systematic mapping study on the academic and the industrial perspective**. *Arabian Journal for Science and Engineering*, vol. 46, pp. 3723–3748, 2021. Springer.

87 NEIRA, Anderson, STEINMACHER, Igor, and WIESE, Igor. **Characterizing the hyperspecialists in the context of crowdsourcing software development**. *Journal of the Brazilian Computer Society*, vol. 24, pp. 17, 2018. doi:10.1186/s13173-018-0082-2.

88 BEECHAM, Sarah, CARROLL, Noel, and NOLL, John. **A Decision Support System for Global Team Management: Expert Evaluation REMIDI**. In *Proceedings - 2012 IEEE 7th International Conference on Global Software Engineering Workshops, ICGSEW 2012*, 2012. doi:10.1109/ICGSEW.2012.14.

89 ZHANG, Xunhui, WANG, Tao, YIN, Gang, YANG, Cheng, YU, Yue, and WANG, Huaimin. **DevRec: A Developer Recommendation System for Open Source Repositories**. In *Mastering Scale and Complexity in Software Reuse*, edited by BOTTERWECK, Goetz and WERNER, Claudia, pp. 3–11, 2017. Springer International Publishing, Cham. ISBN 978-3-319-56856-0.

90 PEREIRA, Thaís Alves Burity, dos SANTOS, Vinicius Souza, RIBEIRO, Bruno Luna, and ELIAS, Glêdson. **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*, pp. 36–40, 2010. Association for Computing Machinery, New York, NY, USA. ISBN 9781605589749. doi:10.1145/1808920.1808928.

91 MCAFEE, Andrew and BRYNJOLFSSON, Erik. **Machine, platform, crowd: Harnessing our digital future**. WW Norton & Company, 2017.

92 TRANFIELD, David, DENYER, David, and SMART, Palminder. **Towards a methodology for developing evidence-informed management knowledge by means of systematic review**. *British Journal of Management*, vol. 14, no. 3, pp. 207–222, 2003. Wiley Online Library.

93 SIEVI-KORTE, Outi, BEECHAM, Sarah, and RICHARDSON, Ita. **Challenges and recommended practices for software architecting in global software development**. *Information and Software Technology*, vol. 106, pp. 234-253, 2019. ISSN 0950-5849. doi: `https://doi.org/10.1016/j.infsof.2018.10.008`. `https://www.sciencedirect.com/science/article/pii/S0950584918302209`.

94 HORTA, Vitor, STRÖELE, Victor, SCHETTINO, Vinicius, OLIVEIRA, Jonice, MARIA DAVID, José, and ARAÚJO, Marco Antônio P. **Collaboration Analysis in Global Software Development**. In *2019 IEEE 23rd International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pp. 464-469, 2019. doi:10.1109/CSCWD.2019.8791857.

95 HOEHNDORF, Robert, GKOUTOS, Georgios V, and SCHOFIELD, Paul N. **Datamining with Ontologies**. *Methods Mol Biol*, vol. 1415, pp. 385–397, 2016. United States.

96 HERRE, Heinrich. **General Formal Ontology (GFO): A Foundational Ontology for Conceptual Modelling**. In *Theory and Applications of Ontology: Computer Applications*, edited by POLI, Roberto, HEALY, Michael, and KAMEAS, Achilles, pp. 297–345. Springer Netherlands, Dordrecht, 2010. ISBN 978-90-481-8847-5. https://doi.org/10.1007/978-90-481-8847-5_14.

97 FREY, C.B. and OSBORNE, M. **Technology at Work: The Future of Innovation and Employment**. 2015. https://books.google.com.br/books?id=q9XRjgEACAAJ.

98 SCHWAB, Klaus. **The Fourth Industrial Revolution**. *Foreign Affairs*, Jan 2016. https://www.foreignaffairs.com/articles/2015-12-12/fourth-industrial-revolution.

99 RICCI, Francesco, ROKACH, Lior, and SHAPIRA, Bracha, editors. **Recommender Systems: Techniques, Applications, and Challenges**. In *Recommender Systems Handbook*, pp. 1–35. Springer US, New York, NY, 2022. ISBN 978-1-0716-2197-4. doi: https://doi.org/10.1007/978-1-0716-2197-4_1.

100 KNOKE, David and YANG, Song. **Social Network Analysis**. Volume 154, 2008.

101 GROTH, Paul and MOREAU, Luc, editors. **PROV-Overview**. 2013. https://www.w3.org/TR/prov-overview/.

102 GHAISAS, Smita. **Recommendation system for agile software development**. U.S. Patent 9262126B2, 2010.

103 MOREIRA, Catarina and WICHERT, Andreas. **Finding academic experts on a multisensor approach using Shannon's entropy**. *Expert Systems with Applications*, vol. 40, no. 14, pp. 5740–5754, Oct 2013. Elsevier BV. doi: https://doi.org/10.1016/j.eswa.2013.04.001.

104 LIN, Shuyi, HONG, Wenxing, WANG, Dingding, and LI, Tao. **A survey on expert finding techniques**. *Journal of Intelligent Information Systems*, vol. 49, Oct 2017. doi:10.1007/s10844-016-0440-5.

105 ALARFAJ, Fawaz, KRUSCHWITZ, Udo, HUNTER, David, and FOX, Chris. **Finding the Right Supervisor: Expert-Finding in a University Domain**. In *Proceedings of the 2012 Conference*, pp. 1–6, 2012, Jun.

106 LOPES, Tales, STRÖELE, Victor, BRAGA, Regina, DAVID, José Maria N., and BAUER, Michael. **Identifying and Recommending Experts Using a Syntactic-Semantic Analysis Approach**. In *2021 IEEE 24th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pp. 739–744, 2021. doi:10.1109/CSCWD49262.2021.9437785.

107 SCHETTINO, Vinicius, HORTA, Vitor, ARAÚJO, Marco Antônio P., and STRÖELE, Victor. **Towards Community and Expert Detection in Open Source Global Development**. In *2019 IEEE 23rd International Conference on*

Computer Supported Cooperative Work in Design (CSCWD), pp. 350–355, 2019. doi:10.1109/CSCWD.2019.8791872.

108 MUSEN, M.A. **The Protégé project: A look back and a look forward**. *AI Matters. Association of Computing Machinery Specific Interest Group in Artificial Intelligence*, Jun 2015. doi:10.1145/2557001.25757003.

109 MATTURRO, Gerardo, RASCHETTI, Florencia, and FONTÁN, Carina. **A Systematic Mapping Study on Soft Skills in Software Engineering**. *Journal of Universal Computer Science*, vol. 25, pp. 16–41, Jan 2019. doi: 10.3217/jucs-025-01-0016.

110 KITCHENHAM, B. and CHARTERS, S. **Guidelines for performing Systematic Literature Reviews in Software Engineering**. 2007.

111 POURHEIDARI, Vahid, MOLLASHAHI, Ehsan Sotoodeh, VASSILEVA, Julita, and DETERS, Ralph. **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)*, pp. 215–222, 2018. doi:10.1109/IEMCON.2018.8614793.

112 MARTÍNEZ-GARCÍA, Jose R., CASTILLO-BARRERA, Francisco-Edgar, PALACIO, Ramon R., BORREGO, Gilberto, and CUEVAS-TELLO, Juan C. **Ontology for knowledge condensation to support expertise location in the code phase during software development process**. *IET Software*, vol. 14, no. 3, pp. 234–241, 2020. doi: https://doi.org/10.1049/iet-sen.2019.0272. https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/iet-sen.2019.0272. https://ietresearch.onlinelibrary.wiley.com/doi/pdf/10.1049/iet-sen.2019.0272.

113 US Environmental Protection Agency. **Final aquatic life ambient water quality criteria for aluminum 2018**. 2018.

114 SIMON, Herbert A. **The sciences of the artificial**. MIT Press, 1996.

115 COLE, Robert, PURAO, Sandeep, ROSSI, Matti, and SEIN, Maung. **Being proactive: where action research meets design research**. In *ICIS 2005 proceedings*, pp. 27, 2005.

116 JÄRVINEN, Pertti. **Action research is similar to design science**. *Quality & Quantity*, vol. 41, pp. 37–54, 2007. Springer.

117 HEVNER, Alan R. **A three cycle view of design science research**. *Scandinavian Journal of Information Systems*, vol. 19, no. 2, pp. 4, 2007.

118 ÖZSU, M Tamer and VALDURIEZ, Patrick. **Principles of distributed database systems**. Vol. 2, Springer, 1999.

119 MILOSLAVSKAYA, Natalia and TOLSTOY, Alexander. **Big data, fast data and data lake concepts**. *Procedia Computer Science*, vol. 88, pp. 300–305, 2016. Elsevier.

120 BUNEMAN, Peter, KHANNA, Sanjeev, and WANG-CHIEW, Tan. **Why and where: A characterization of data provenance**. In *International Conference on Database Theory*, pp. 316–330, 2001. Springer.

121 SIEVI-KORTE, Outi, BEECHAM, Sarah, and RICHARDSON, Ita. **Challenges and recommended practices for software architecting in global software development**. *Information and Software Technology*, vol. 106, pp. 234–253, 2019. doi: https://doi.org/10.1016/j.infsof.2018.10.008. https://www.sciencedirect.com/science/article/pii/S0950584918302209.

122 HORTA, Vitor, STRÖELE, Victor, SCHETTINO, Vinicius, OLIVEIRA, Jonice, MARIA DAVID, José, and ARAÚJO, Marco Antônio P. **Collaboration Analysis in Global Software Development**. In *2019 IEEE 23rd International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pp. 464–469, 2019. doi:10.1109/CSCWD.2019.8791857.

123 HOEHNDORF, Robert, GKOUTOS, Georgios V, and SCHOFIELD, Paul N. **Datamining with Ontologies**. *Methods Mol Biol*, vol. 1415, pp. 385–397, 2016. United States. Keywords: Automated reasoning; Data integration; Enrichment; Graph algorithms; Ontology; Semantic Web; Semantic similarity; Web Ontology Language (OWL).

124 HERRE, Heinrich. **General Formal Ontology (GFO): A Foundational Ontology for Conceptual Modelling**. In *Theory and Applications of Ontology: Computer Applications*, Poli, Roberto, Healy, Michael, and Kameas, Achilles (eds.), pp. 297–345, Springer Netherlands, Dordrecht, 2010. doi: 10.1007/978-90-481-8847-5_14.

125 HYRYNSALMI, Sonja M., RANTANEN, Minna M., and HYRYNSALMI, Sami. **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)*, pp. 1–10, 2021. doi:10.1109/ICE/ITMC52061.2021.9570207.

126 FREY, C.B. and OSBORNE, M. **Technology at Work: The Future of Innovation and Employment**. 2015. https://books.google.com.br/books?id=q9XRjgEACAAJ.

127 SCHWAB, Klaus. **The Fourth Industrial Revolution**. *Foreign Affairs*, Jan 2016. https://www.foreignaffairs.com/articles/2015-12-12/fourth-industrial-revolution.

128 RICCI, Francesco, ROKACH, Lior, and SHAPIRA, Bracha. **Recommender Systems: Techniques, Applications, and Challenges**. In *Recommender Systems Handbook*, Ricci, Francesco, Rokach, Lior, and Shapira, Bracha (eds.), pp. 1–35, Springer US, New York, NY, 2022. doi: 10.1007/978-1-0716-2197-4_1. https://doi.org/10.1007/978-1-0716-2197-4_1.

129 GUARINO, Nicola, OBERLE, Daniel, and STAAB, Steffen. **What Is an Ontology?** In *Handbook on Ontologies*, Staab, Steffen and Studer, Rudi (eds.), pp. 1–17, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. doi: 10.1007/978-3-540-92673-3_0. https://doi.org/10.1007/978-3-540-92673-3_0.

130 AL-TAIE, Mohammed Zuhair, KADRY, Seifedine, and OBASA, Adekunle Isiaka. **Understanding expert finding systems: domains and techniques**. *Social Network Analysis and Mining*, vol. 8, no. 1, p. 57, Aug 2018. doi: 10.1007/s13278-018-0534-x. https://doi.org/10.1007/s13278-018-0534-x.

131 ZHANG, Xunhui, WANG, Tao, YIN, Gang, YANG, Cheng, YU, Yue, and WANG, Huaimin. **DevRec: A Developer Recommendation System for Open Source Repositories**. In *Mastering Scale and Complexity in Software Reuse*, Botterweck, Goetz and Werner, Claudia (eds.), pp. 3–11, Springer International Publishing, Cham, 2017. ISBN: 978-3-319-56856-0.

132 PEREIRA, Thaís Alves Burity, DOS SANTOS, Vinicius Souza, RIBEIRO, Bruno Luna, and ELIAS, Glédson. **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*, pp. 36–40, Association for Computing Machinery, New York, NY, USA, 2010. doi: 10.1145/1808920.1808928. https://doi.org/10.1145/1808920.1808928.

133 BEECHAM, Sarah, CARROLL, Noel, and NOLL, John. **A Decision Support System for Global Team Management: Expert Evaluation REMIDI**. In *Proceedings - 2012 IEEE 7th International Conference on Global Software Engineering Workshops, ICGSEW 2012*, 2012, pp. –. doi: 10.1109/ICGSEW.2012.14.

134 CIFARIELLO, Paolo, FERRAGINA, Paolo, and PONZA, Marco. **Wiser: A semantic approach for expert finding in academia based on entity linking**. *Information Systems*, vol. 82, pp. 1-16, 2019. ISSN: 0306-4379. doi: https://doi.org/10.1016/j.is.2018.12.003. https://www.sciencedirect.com/science/article/pii/S0306437918302515.

135 NEIRA, Anderson, STEINMACHER, Igor, and WIESE, Igor. **Characterizing the hyperspecialists in the context of crowdsourcing software development**. *Journal of the Brazilian Computer Society*, vol. 24, p. 17, Dec 2018. doi: 10.1186/s13173-018-0082-2.

136 DE CAMPOS, Vitor, DAVID, José Maria N, STRÖELE, Victor, and BRAGA, Regina. **Supporting the recruitment of software development experts: aligning technical knowledge to an industry domain**. In *Anais do XVIII Simpósio Brasileiro de Sistemas Colaborativos*, pp. 183–192, SBC, 2023.

137 DE CAMPOS, Vitor, DAVID, José Maria N, STRÖELE, Victor, BRAGA, Regina, and OLIVEIRA, Alessandreia M. **DevFinder: An approach to finding expert software developers considering desired technologies and industry domains**. In *2023 26th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pp. 1446–1451, IEEE, 2023.

138 DE CAMPOS, Vitor, DAVID, José Maria N., STRÖELE, Victor, and BRAGA, Regina. **Aligning technical knowledge to an industry domain in global software development: A systematic mapping**. *Journal of Software: Evolution and Process*, vol. n/a, no. n/a, p. e2713, 2024. doi: 10.1002/smr.2713. https://onlinelibrary.wiley.com/doi/abs/10.1002/smr.2713. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/smr.2713.

139 GRUBER, Thomas R. **Toward principles for the design of ontologies used for knowledge sharing**. *International Journal of Human-Computer Studies*, vol. 43, no. 5-6, pp. 907–928, 1995. Publisher: Elsevier.

140 LOUIE, Brenton, MORK, Peter, MARTIN-SANCHEZ, Fernando, HALEVY, Alon, and TARCZY-HORNOCH, Peter. **Data integration and genomic medicine**. *Journal of Biomedical Informatics*, vol. 40, no. 1, pp. 5–16, 2007. Publisher: Elsevier.

141 ZITNIK, Marinka, NGUYEN, Francis, WANG, Bo, LESKOVEC, Jure, GOLDENBERG, Anna, and HOFFMAN, Michael M. **Machine learning for integrating data in biology and medicine: Principles, practice, and opportunities**. *Information Fusion*, vol. 50, pp. 71–91, 2019. Publisher: Elsevier.

142 McGUINNESS, Deborah L. and VAN HARMELEN, Frank, et al. **OWL web ontology language overview**. *W3C recommendation*, vol. 10, no. 10, pp. 2004, 2004.

143 O'CONNOR, Martin J., SHANKAR, Ravi D., NYULAS, Csongor, TU, Samson W., and DAS, Amar K. **Developing a Web-Based Application using OWL and SWRL**. In *AAAI Spring Symposium: AI Meets Business Rules and Process Management*, pp. 93–98, 2008.

144 HORROCKS, Ian, PATEL-SCHNEIDER, Peter F., BOLEY, Harold, TABET, Said, GROSOF, Benjamin, DEAN, Mike, et al. **SWRL: A semantic web rule language combining OWL and RuleML**. *W3C Member Submission*, vol. 21, no. 79, pp. 1–31, 2004.

145 HYRYNSALMI, Sonja M., RANTANEN, Minna M., and HYRYNSALMI, Sami. **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)*, pp. 1–10, 2021. doi: 10.1109/ICE/ITMC52061.2021.9570207.

146 WELTY, Christopher and GUARINO, Nicola. **Supporting ontological analysis of taxonomic relationships**. *Data & Knowledge Engineering*, vol. 39, no. 1, pp. 51–74, 2001. Publisher: Elsevier.

147 MOTIK, Boris. **On the properties of metamodeling in OWL**. *Journal of Logic and Computation*, vol. 17, no. 4, pp. 617–637, 2007. Publisher: OUP.

148 CHACON, Scott and STRAUB, Ben. **Pro Git**. Springer Nature, 2014.

149 LOELIGER, Jon and MCCULLOUGH, Matthew. **Version Control with Git: Powerful tools and techniques for collaborative software development**. O'Reilly Media, Inc., 2012.

150 DABBISH, Laura, STUART, Colleen, TSAY, Jason, and HERBSLEB, Jim. **Social coding in GitHub: transparency and collaboration in an open software repository**. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work*, pp. 1277–1286, 2012.

151 HUA, Wenyue, LI, Lei, XU, Shuyuan, CHEN, Li, and ZHANG, Yongfeng. **Tutorial on Large Language Models for Recommendation**. In *Proceedings of the 17th ACM Conference on Recommender Systems*, pp. 1281–1283, 2023.

152 DI PALMA, Dario. **Retrieval-augmented recommender system: Enhancing recommender systems with large language models**. In *Proceedings of the 17th ACM Conference on Recommender Systems*, pp. 1369–1373, 2023.

153 HENDERSON, John C. and VENKATRAMAN, Harihara. **Strategic alignment: Leveraging information technology for transforming organizations**. *IBM Systems Journal*, vol. 38, no. 2.3, pp. 472–484, 1999. Publisher: IBM.

154 ROSS, Jeanne W. and VITALE, Michael R. **The ERP revolution: surviving vs. thriving**. *Information Systems Frontiers*, vol. 2, pp. 233–241, 2000. Publisher: Springer.

155 BLANKA, Christine, KRUMAY, Barbara, and RUECKEL, David. **The interplay of digital transformation and employee competency: A design science approach**. *Technological Forecasting and Social Change*, vol. 178, p. 121575, 2022. Publisher: Elsevier.

156 ISKOUJINA, Zilia and ROBERTS, Joanne. **Knowledge sharing in open source software communities: motivations and management**. *Journal of Knowledge Management*, vol. 19, no. 4, pp. 791–813, 2015. Publisher: Emerald Group Publishing Limited.

157 BROOKS, Frederick. **The mythical man-month (anniversary ed.)**. Addison-Wesley Longman Publishing Co., Inc., 1995.

158 ROTHWELL, Jonathan T. **Defining skilled technical work**. Available at SSRN 2709141, 2015.

159 TARAFDAR, Monideepa and GORDON, Steven R. **Understanding the influence of information systems competencies on process innovation: A resource-based view**. *The Journal of Strategic Information Systems*, vol. 16, no. 4, pp. 353–392, 2007. Publisher: Elsevier.

160 ZHANG, Junjie, XIE, Ruobing, HOU, Yupeng, ZHAO, Wayne Xin, LIN, Leyu, and WEN, Ji-Rong. **Recommendation as instruction following: A large language model empowered recommendation approach**. *arXiv preprint arXiv:2305.07001*, 2023.

161 CALDIERA, Victor R Basili1 Gianluigi and ROMBACH, H Dieter. **The goal question metric approach**. In *Encyclopedia of Software Engineering*, pp. 528–532, 1994.

162 FÖLL, Patrick and HAUSER, Matthias and THIESSE, Frederic. **Identifying the Skills Expected of IS Graduates by Industry: A Text Mining Approach**. In Proceedings, 2018.