Combining Knowledge Graphs and Large Language Models to Ease Knowledge Access in Software Architecture Research

Angelika Kaplan[1], Jan Keim[1], Marco Schneider[1], Anne Koziolek[1] and Ralf Reussner[1]

[1]*Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany*

Abstract

Science enables progress. Software engineering research and its sub-research fields like software architecture have high economic relevance (cf. "software is eating the world") and serve as an enabler for technical innovations in other research fields as well. The ever-increasing amount of research results and insights creates the need for efficient knowledge documentation and search functions to access relevant information. Till now, scientific papers have been published digitally as PDF documents. Researchers and practitioners can access these documents via digital libraries and scientific search engines mostly with a simple keyword-based search. Literature studies in software engineering show that this procedure (a) requires enhanced information literacy skills even for domain experts and (b) is very time-consuming for aggregating evidence in written PDF articles to a specific field of interest. Consequently, we can derive a need for speeding up the information-finding process and effective knowledge representation as well as aggregation for evidence-based literature studies. As a result, we present our approach idea for effective knowledge access and discovery in software architecture research by combining knowledge graphs and large language models, using the complementary advantages of both concepts.

Keywords

Knowledge Graphs, Large Language Models, Scientific Knowledge Access, Software Architecture Research

1. Introduction

In the rapidly evolving field of software engineering (SE) and software architecture (SWA), researchers and practitioners alike are confronted with the task of navigating an ever-expanding universe of knowledge and the immense growth of the SE literature (cf. Figure 3 in [1]). As there is no dedicated database for scientific software architecture knowledge, one has to use academic search engines to access relevant information while struggling with bugs (i.e., faults in the search engines) and usability issues [2]. It is a challenging and time-consuming task to find the required information for each unique inquiry. There are ways, however, to speed up the information-finding process. Traditional methods of information retrieval are heavily reliant on keyword-based searches that also require manual sifting through matching documents. Additionally, this method is reliant on exact keyword matches as it lacks the sophistication needed to understand the context of queries, overall leading to time-consuming searches and incomplete or irrelevant results.

In this context, knowledge graphs (KGs) represent a leap forward in the way we manage and retrieve information [3]. By structuring data in an interconnected network of entities and their relationships, KGs can offer a context-aware system that can help leverage the limitations of traditional keyword-based searches. The key strength of KGs lies in their ability to semantically connect information, which means they can set the context and relationships between different pieces of data. This semantic understanding enables more accurate and relevant search results, as the system can interpret the intent behind a query rather than merely matching keywords. In the domain of software engineering research, KGs can streamline the process of literature reviews and state-of-the-art surveys (cf. [4]). With the connections presented by the KG, researchers can quickly identify influential works, related research topics, and the development trajectory of specific technologies or methodologies. This is made possible by analyzing

2nd Intl. WS on Semantic Technologies and Deep Learning Models for Scientific, Technical and Legal Data (SemTech4STLD)

angelika.kaplan@kit.edu (A. Kaplan); jan.keim@kit.edu (J. Keim); marco.schneider@student.kit.edu (M. Schneider); koziolek@kit.edu (A. Koziolek); reussner@kit.edu (R. Reussner)

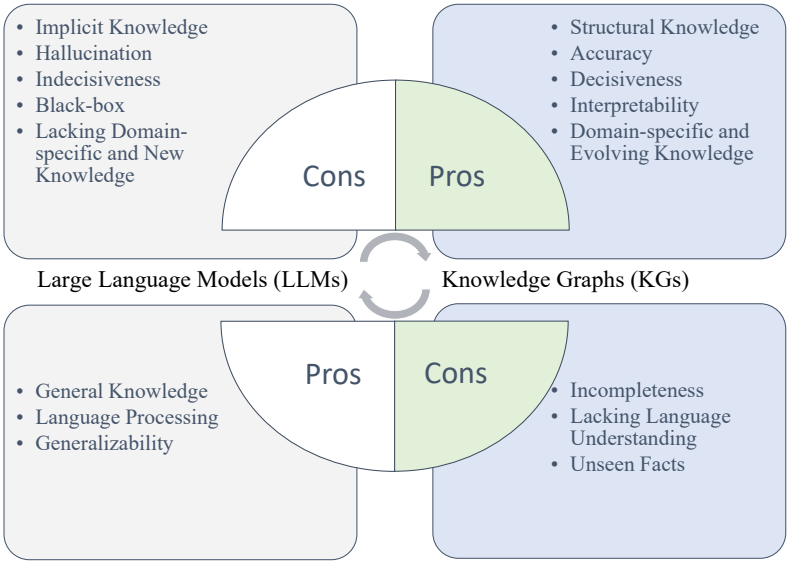

Figure 1: Summarization of the pros and cons for LLMs and KGs according to Pan et al. [3]

the connections between research papers, authors, and their affiliations, thereby uncovering the network of knowledge that shapes the field. One example of such a KG is the knowledge management system Open Research Knowledge Graph (ORKG). The ORKG offers different views and management options to extract and organize scientific knowledge from published papers. For this, the ORKG relies on users who (manually) add papers to the system. As a core object, ORKG uses a domain model that specifies a *Research Contribution* [5]. A *Research Contribution* is linked to a *Research Problem*, a *Research Method*, a *Research Result* and to *Research Material*. To create domain-specific descriptions of publications in a dedicated research field, users of ORKG can use templates. Users can then perform subsequent tasks like comparisons and (literature) reviews. However, these tasks are manual.

With the uprising of Large Language Models (LLMs), another transformative tool is being integrated into the knowledge retrieval process [3]. LLMs provide an advanced capacity for natural language understanding and generation, which complements the structured, semantic connections provided by KGs. LLMs excel in the interpretation of nuances of natural language queries. This allows the user to ask complex questions in their own words rather than relying on specific keywords. This ability reduces the barriers to accessing information and, thus, speeds up research tasks. However, LLMs tend to hallucinate and fabricate information in their responses, making them less reliable. This is also reflected in current state-of-practice tools (i.e., closed source with unknown database) like Elicit [6], SciSpace [7, 8], and Consensus [9] that lack of transparency due to the black-box nature. They suffer from the aforementioned limitations, as summarized by Pan et al. [3].

Our idea is to combine the capabilities of LLMs with the structured data of KGs and, thus, use the complementary advantages of each concept (as depicted in Figure 1). Combined, LLMs will be able to use the semantic networks to provide informed, more precise and contextually relevant responses to queries. Furthermore, we can cross-check and verify responses with the data of the KGs, reducing well-known negative behaviors of LLMs like hallucinating. This synergy enhances the way information is processed, understood, and retrieved by leveraging the search for information to a semantic, context-aware level.

Consequently, we address the research question (RQ): *How to construct an approach that combines the advantages of knowledge graphs and large language models to ease knowledge access (in software architecture research)?*

The paper is structured as follows. Section 2 introduces example use cases where we plan to use the approach and where it can be beneficial. In Section 3, we detail the approach by answering our RQ and outlining preliminary work and results. Related work is discussed in Section 4. Lastly, we conclude the paper in Section 5.

2. Example Use Cases

In this section, we present example use cases for our approach, where we think it can be beneficial.

1. Use Case (Find Papers by Topic): The user is interested in topic α and wishes to find a list of papers to read about it.
Prompt/Natural language query: Please find papers that are about topic α.
Expectation: The system generates a list of papers including persistence identifier (DOI) that are related to the specified topic α. The list is sorted by their relevance to topic α.

2. Use Case (Find Papers by quoted Sentence): The user recalls reading a particular sentence in a paper but cannot remember which paper it was. They can only memorize it semantically and cannot reproduce the full sentence.
Prompt/Natural language query: Please find papers that are related to the following sentence β.
Expectation: The system generates a list of papers containing related sentences, sorted by relevance.

3. Use Case (Paper Summarization): The user has a lot of papers they need to read through and would like to receive help to capture the essence of each paper quickly.
Prompt/Natural language query: Please summarize the key points of the paper with title γ.
Expectation: The system generates a structured summary with the key aspects of the paper titled γ.

4. Use Case (Search for Methods used in Topic): The user wants to do research for topic δ. They would like to find out about state-of-the-art methodologies that are applied in this topic.
Prompt/Natural language query: Tell me more about methodologies that are used for topic δ.
Expectation: The system returns a structured answer that lists and explains each methodology.

5. Use Case (Search for Benchmarks / Datasets for Topic): The user searches for benchmarks, datasets, or input tools for topic ϵ. They would like to find external resources and artifacts to evaluate the new solution approach.
Prompt/Natural language query: Which replication artifacts are commonly used for topic ϵ.
Expectation: The system returns a structured answer that lists and explains each kind of used artifact.

3. Approach

In this section, we present our idea and architecture to combine KGs and LLMs to enhance and ease scientific knowledge access in software architecture research for researchers and practitioners by answering our research question (cf. Section 1). The process is shown in Figure 2. There are two main parts of the approach: (1) Knowledge graph generation and population, and (2) knowledge retrieval. We call this approach *Knowledge Augmented Retrieval And Generation ENgine (KARAGEN)*. The corresponding open-source repository [10] comprises artifacts and projects related to our approach.

3.1. Architecture

First, we plan to support generating and populating KGs like ORKG by aiding the users with the help of LLM-based classification. In this semi-automated process, when users add papers to the KG, the classification approach recommends the required information to add research works and publications to the KG. For example for ORKG, this process includes filling out the respective template or data schema. We describe a preliminary example for this in Section 3.2.

Second, KARAGEN supports knowledge retrieval by querying and processing the information from the KGs using LLMs. Users can state their query in natural language. An LLM-based component processes the query. The processing can include several processing steps (e.g., Named Entity Recognition) as well as generating queries to the KG to retrieve the relevant data. For each of the retrieved dates, a knowledge enhancement component can (optionally) further use LLMs to transform and generate information. For example, the component can summarize research content, classify it or use similar enhancement strategies. Finally, KARAGEN uses all this information to generate an output for the

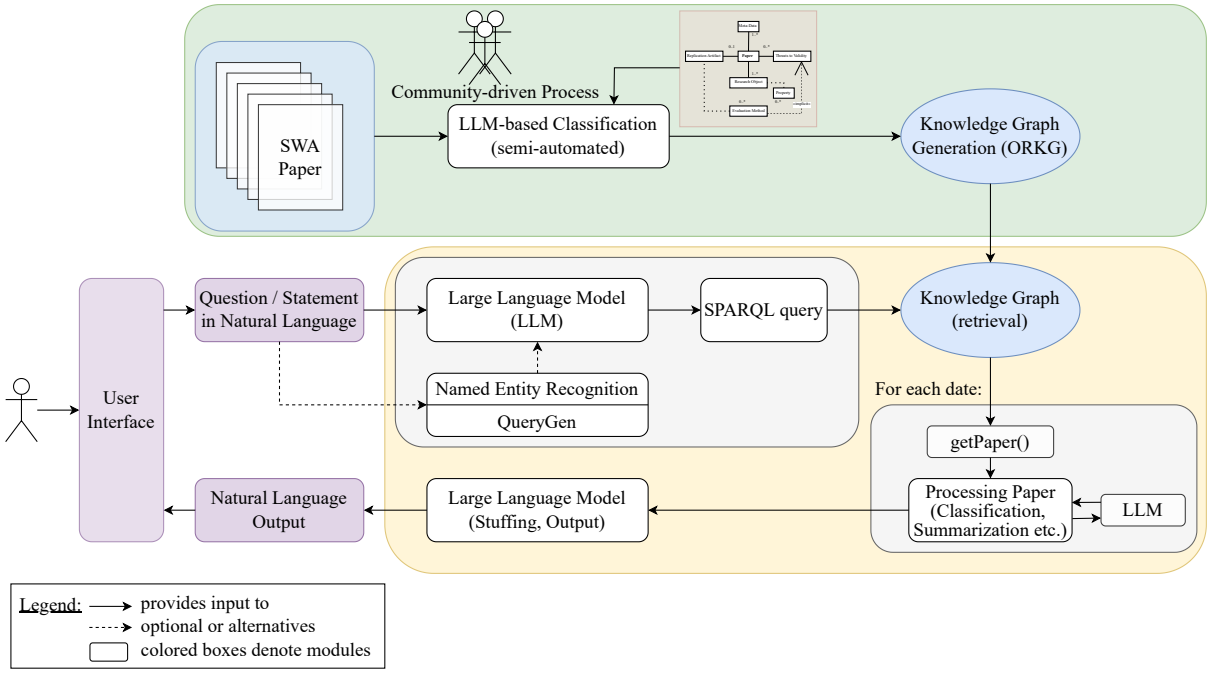

Figure 2: Overview of the Knowledge Augmented Retrieval And Generation ENgine (KARAGEN)

users, ranking data initially based on information from the KG and later including feedback from the LLM. KARAGEN can also add sources to the generated output as the origin of the data is known.

While the approach has similarities to existing retrieval augmented generation (RAG) approaches, there are notable differences. RAG approaches use vector stores that contain natural language texts together with their embeddings. These embeddings are used to retrieve entries with similar embeddings to a given query. We implemented such an approach in a prototype that is briefly introduced in Section 3.3. We plan to enhance the prototype by adding information retrieval from the KG. The approach can then enhance this information and, thus, potentially increase the usefulness and correctness of the outputs as depicted in Figure 2. Adding KG into the process allows us to employ error detection strategies and proof-checking of results before returning them to the users to strengthen correctness. These strategies include cross-checking LLM-generated results with the KG, where possible. Additionally, confidence scores, calculated from the confidence of the LLM and/or KG towards some data, allow us to filter low-confidence results based on an individual threshold that can be set by the user.

3.2. Data Schema for Software Architecture Research

Konersmann, Kaplan, and Kühn et al. [11] introduced a data schema for the description of SWA research. They provide a compact description and characterization of this research field and give an overview of the evaluation and replicability of SWA research objects. In a community effort, they labeled 153 papers in SWA research according to the data schema depicted in Figure 3. This schema offers a description approach for a paper's content as well as the respective meta-data such as bibliographic information. Central is the *Research Object* under investigation w.r.t. a certain *Property*, evaluated with an *Evaluation Method*. This schema fosters knowledge description and representation in SWA research and contributes to an ORKG template (cf. Section 1) in this research field to support the corresponding ORKG observatory[1]. We already constructed a prototype for semi-automatic classification as depicted in Figure 2 (green box). For this, we implemented a classification framework based on a blackboard pattern (cf. Schmidt et al. [12]) supporting (i) prompting techniques and (ii) fine-tuning language models with hyperparameter searches. This approach is universal by flexibly allowing various input data and

[1] https://orkg.org/observatory/Software_Architecture_Research [Last accessed on 2024-03-18]

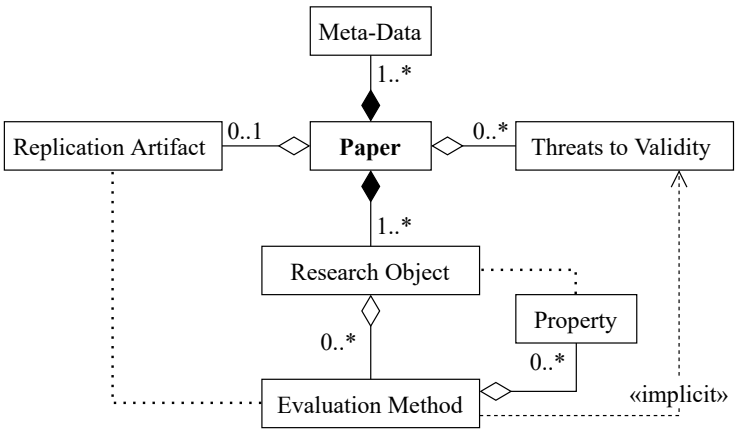

Figure 3: Data schema for describing software architecture research papers. Dashed arrows indicate inferred relations. Extended from [11].

schemas, so it enables handling evolutionary changes and extensions in the data schema. Moreover, we can specify variations points (w.r.t. among others data split and data augmentation techniques as well as the aforementioned classification paradigms) in the experimental setup with focus on open-source language models like Llama2, Mistral, and Mixtral. Initial experiments with small datasets (based on [13]) show promising results, but we require further experiments with more data for reliable results and conclusive statements.

3.3. Software Architecture Research Discovery Engine

The Software Architecture Research Discovery Engine project is aimed at enhancing the capabilities of an LLM through a *modular functional approach* (cf. parts of the yellow box as depicted in Figure 2). In this project, we want to leverage LLMs for a variety of tasks, such as summarizing papers, locating specific information, and conducting semantic searches within scientific literature (cf. example use cases in Section 2). The project is currently a work in progress and is in an experimental stage. The prototype uses papers from the European Conference on Software Architecture (ECSA)[2]. This enables us to explore the capabilities of synergy between the LLM and its connection to a scientific data source. Although this initial dataset does not constitute a KG, we will include a structured KG in the future. Furthermore, the *modular functional approach*, facilitated by LangChain [14], enables the creation of an agent equipped with a diverse toolkit, overseen by an LLM. This LLM is tasked with evaluating the toolkit and selecting the most appropriate tool based on the given prompt's intention. Our implementation leverages this framework to offer varied functionalities that address the mentioned use cases in Section 2. This design philosophy ensures that, as new use cases emerge, the system can be seamlessly expanded. We plan to further enhance the agent's capabilities by integrating a KG, thereby broadening the scope of its functionality and improving its effectiveness in navigating and processing scientific literature.

4. Related Work

In the following, we identify and present three main research areas related to our work.

Scientific Knowledge Graphs in Related Research Fields. First, we regard an approach that builds upon the ORKG in the research field of requirements engineering (RE) as a related field to SWA. Karras et al. [4] investigate this KG as a technical infrastructure by building, publishing, and evaluating an initial KG for empirical research in RE – the so-called KG-EmpiRE. For this, they collected papers from the research track of the IEEE International Requirements Engineering Conference and extracted

[2] https://link.springer.com/conference/ecsa [Last accessed on 2024-03-18]

data from 570 papers that report on empirical research. Derived from this, they build and publish the initial KG-EmpiRE that the research community can constantly maintain, (re-)use, update, and expand.

Structured Forms and Description Approaches for Scientific Knowledge Representation in Software Architecture. Second, we consider structured forms and description approaches for software architecture research and its superordinate research field software engineering. In this context, we refer to the data schema for software architecture research of Konersmann, Kaplan, and Kühn et al. [11]. The authors made a first attempt to describe this research field. Based on the labeling in a community effort, there are already limitations of the schema identified. However, there is existing work by Kaplan et al. [15] to overcome these limitations and holistically evolve the initial aforementioned schema by enhancing the respective entities and relations to enrich the corresponding KG.

Access to Software Architecture Knowledge. Lastly, we present state-of-the-art/state-of-practice access approaches to software architecture knowledge. In this context, we differentiate between two main knowledge sources: (1) scientific, i.e., evidence-based knowledge and (2) practical knowledge in an industrial context. For the first knowledge source, we refer to the serverless pipeline that is based on the replication package [13] of the work of Konersmann, Kaplan, and Kühn et al. [11]. The authors provide – based on the specification of the data schema – a React webpage Visulite (Visualization of Literature) [16] to enable researchers to have a better overview by browsing scientific SWA literature in a graphical presentation and interactive analysis of the data. In addition, we regard tools based on *retrieval augmented generation (RAG)* like Elicit [6], SciSpace [7, 8], and Consensus [9] as related (cf. Section 1). However, these tools are closed source, only rely on open-use language models (like GPT from OpenAI), and lack domain-specific knowledge (i.e., the underlying database is unknown and the range of the literature corpus in SWA is vague). For the second knowledge source, we refer to the work of Soliman et al. [17]. The authors introduced a new approach for improving knowledge access and search functions for architecturally relevant information in online developer communities. They implemented their approach as a web-based search engine. For this, the authors used a combination of keyword-based search and a specification of the current step in the software architecture to present better results to the user. As an underlying technology, they used the index-based search engine Apache Lucene. The data and information that is used to present new architecture knowledge is based on the content of Stack Overflow, i.e., not considering scientific software architecture knowledge.

5. Conclusion

In this paper, we outlined our approach idea KARAGEN, the Knowledge Augmented Retrieval And Generation ENgine, that combines knowledge graphs (e.g., ORKG) and LLMs to ease knowledge access, e.g., in the research field of software architecture. We expect to combine the benefits of knowledge graphs and LLMs to improve the overall usefulness and correctness of the answers to user queries in comparison to current state-of-practice tools. From LLMs, we expect to benefit from their capabilities to understand and interpret nuances of natural language queries as well as generate coherent natural language output. From KGs, we expect to benefit from the curated and structured knowledge within the KGs to retrieve and check information. In the next steps, we plan to integrate a prototype into our existing work. We then will evaluate the prototype. While we expect the approach, in general, to work for any research area, we will initially limit the study to software architecture research.

Acknowledgments

This work was created as part of the NFDI consortium NFDIxCS (nfdixcs.org). This work is funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under the National Research Data Infrastructure – NFDI 52/1 – project number 501930651 and supported by funding from the pilot program Core Informatics at KIT (KiKIT) of the Helmholtz Association (HGF).

References

[1] V. Garousi, J. M. Fernandes, Highly-cited papers in software engineering: The top-100, Information and Software Technology 71 (2016) 108–128. doi:10.1016/J.INFSOF.2015.11.003.

[2] Z. Li, A. Rainer, Academic search engines: constraints, bugs, and recommendations, in: Á. Kiss, B. Marín, M. Saadatmand (Eds.), Proceedings of the 13th International Workshop on Automating Test Case Design, Selection and Evaluation, A-TEST 2022, Singapore, Singapore, November 17-18, 2022, ACM, 2022, pp. 25–32. doi:10.1145/3548659.3561310.

[3] S. Pan, L. Luo, Y. Wang, C. Chen, J. Wang, X. Wu, Unifying large language models and knowledge graphs: A roadmap (2023). doi:10.48550/ARXIV.2306.08302. arXiv:2306.08302.

[4] O. Karras, F. Wernlein, J. Klünder, S. Auer, Divide and conquer the empire: A community-maintainable knowledge graph of empirical research in requirements engineering, in: ACM/IEEE ESEM 2023, IEEE, 2023, pp. 1–12. doi:10.1109/ESEM56168.2023.10304795.

[5] M. Y. Jaradeh, A. Oelen, K. E. Farfar, M. Prinz, J. D'Souza, G. Kismihók, M. Stocker, S. Auer, Open Research Knowledge Graph: Next Generation Infrastructure for Semantic Scholarly Knowledge, in: 10th K-CAP, ACM, 2019, pp. 243–246. doi:10.1145/3360901.3364435.

[6] J. Byun, A. Stuhlmüller, Elicit: Language models as research tools, OECD Publishing, Paris, 2023. doi:https://doi.org/10.1787/174aee8f-en.

[7] T. Roy, A. Kumar, D. Raghuvanshi, S. Jain, G. Vignesh, K. Shinde, R. Tondulkar, SciSpace Copilot: Empowering Researchers through Intelligent Reading Assistance, Proceedings of the AAAI Conference on Artificial Intelligence 38 (2024) 23826–23828. URL: https://ojs.aaai.org/index.php/AAAI/article/view/30578. doi:10.1609/aaai.v38i21.30578.

[8] S. Jain, A. Kumar, T. Roy, K. Shinde, G. Vignesh, R. Tondulkar, SciSpace Literature Review: Harnessing AI for Effortless Scientific Discovery, in: N. Goharian, N. Tonellotto, Y. He, A. Lipani, G. McDonald, C. Macdonald, I. Ounis (Eds.), Advances in Information Retrieval, Springer Nature Switzerland, Cham, 2024, pp. 256–260.

[9] Consensus: AI search engine for research, 2024. URL: https://consensus.app/.

[10] Knowledge Augmented Retrieval And Generation ENgine (KARAGEN), 2024. URL: https://gitlab.com/software-engineering-meta-research/karagen.

[11] M. Konersmann, A. Kaplan, T. Kühn, R. Heinrich, A. Koziolek, R. H. Reussner, J. Jürjens, M. al-Doori, N. Boltz, M. Ehl, D. Fuchß, K. Großer, S. Hahner, J. Keim, M. Lohr, T. Saglam, S. Schulz, J. Töberg, Evaluation methods and replicability of software architecture research objects, in: 19th IEEE ICSA 2022, IEEE, 2022, pp. 157–168. doi:10.1109/ICSA53651.2022.00023.

[12] D. Schmidt, M. Stal, H. Rohnert, F. Buschmann, Pattern-oriented software architecture, volume 1: a system of patterns, 1996.

[13] M. Konersmann, A. Kaplan, T. Kühn, R. Heinrich, A. Koziolek, R. H. Reussner, J. Jürjens, M. al-Doori, N. Boltz, M. Ehl, D. Fuchß, K. Großer, S. Hahner, J. Keim, M. Lohr, T. Saglam, S. Schulz, J. Töberg, Replication package of "evaluation methods and replicability of software architecture research objects", in: IEEE 19th ICSA 2022, IEEE, 2022, p. 58. doi:10.1109/ICSA-C54293.2022.00021.

[14] LangChain, 2024. URL: https://langchain.com/.

[15] A. Kaplan, T. Kühn, R. H. Reussner, Unifying classification schemes for software engineering meta-research, CoRR abs/2209.10491 (2022). doi:10.48550/ARXIV.2209.10491. arXiv:2209.10491.

[16] Visulite (visualization of literature) based on data schema of 10.1109/ICSA53651.2022.00023, 2023. URL: https://softwarearchitectureresearch.github.io/visulite/.

[17] M. Soliman, A. Rekaby Salama, M. Galster, O. Zimmermann, M. Riebisch, Improving the Search for Architecture Knowledge in Online Developer Communities, in: Proceedings - 2018 IEEE 15th International Conference on Software Architecture, ICSA 2018, Institute of Electrical and Electronics Engineers Inc., 2018, pp. 186–195. doi:10.1109/ICSA.2018.00028.