Intelligent Assistants in the Era of LLMs: A New Methodology for Reusing Research Software from Documentation

Carlos Utrilla Guerrero

Ontology Engineering Group (OEG), Universidad Politécnica de Madrid (UPM), Madrid, Spain

Abstract

As intelligent assistants driven by generative Artificial Intelligent (AI) such as Large Language Model (LLM) have shown remarkable capabilities in supporting software development tasks, including code generation through natural language, a critical question arises: how can these intelligent assistants aid in the reuse of research software from its documentation? This thesis investigates how the research community can benefit from the LLM revolution, specifically in the context of research software (RS) reuse. To this end, this thesis will propose a new methodology that enables AI-based intelligent assistants to interpret, reason, plan and act upon reuse-oriented software documentation—such as README files and procedural guides. By extracting, transforming, and executing procedural instructions, such intelligent assistants have the potential to reduce cognitive and technical burdens on researchers, improve the sustainability of RS, and may alleviate certain pressures associated with modern scientific careers.

Keywords

Research Software Reuse, Multi-agent systems, Intelligent Assistant, Artificial Intelligent

1. Introduction

Today, many researchers are not only generating new knowledge using software but are also increasingly reusing digital infrastructure—such as tools, software, and services—developed by others. A key facilitator of this reuse process is human-generated documentation, particularly README files, which typically provide step-by-step instructions on how to install, configure, and run research software (RS)[1]. These instructions commonly follow established methods—such as package managers, containers, source builds, and/or setup scripts—intended to reduce friction and facilitate reuse[2, 3]. However, even when RS is encapsulated in portable environments like Docker containers or Python packages, researchers are often still required to manually inspect the documentation, make decisions about procedural steps, and resolve ambiguity. Executing these instructions accurately is not a trivial task: it involves modeling the structure of the installation process, locating where install commands may break down, and reasoning through multiple layers of document complexity without standards. These unstructured narratives present a major obstacle [4] to interpreting and executing instructions by both humans and machines, ultimately limiting the automation of research software reuse from documentation [5]. As a result, the full potential of automated research software reuse from documentation cannot be realized until we understand what set of *procedural* language processing, reasoning and planning capabilities are required to enable machines—or AI-based assistants—to effectively read, interpret, execute, and validate complex reuse instructions contained in human-generated documentation [6]. Here, we tackle this challenge by exploring a multidimensional approach encompassing these following tasks: (1) **extracting** install-related instructions from README files and (2) **transforming** them into a format that can be (3) **executed** by machines at a minimum cost.

The problem of automating software reuse (in general) has long been recognized in the field of AI and software engineering [7], but has come to prominence recently with the emergence of contemporary

Doctoral Symposium on Natural Language Processing, 25 September 2025, Zaragoza, Spain.
carlos.utrilla.guerrero@alumnos.upm.es (C. U. Guerrero)
https://carlosug.github.io/ (C. U. Guerrero)
0000-0002-9994-1462 (C. U. Guerrero)

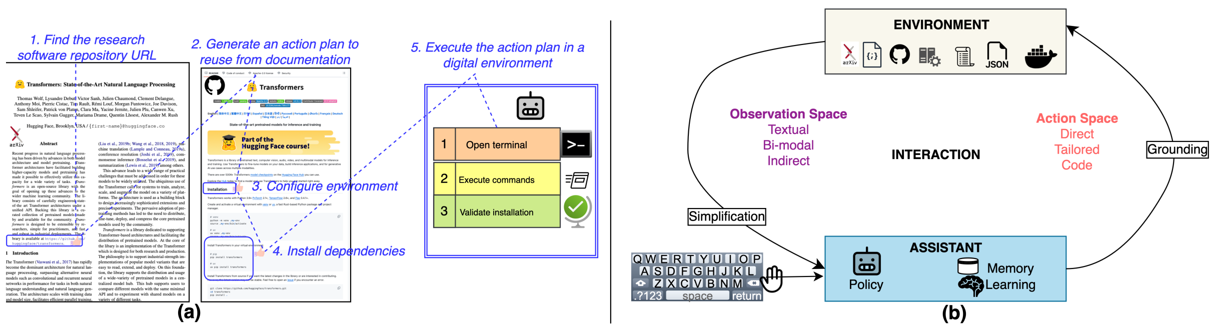


Figure 1: (a) An example task for reuse assistant: a researcher specifies a software project e.g., by an URL of a git repository, and the assistant generates an action plan and execute its commands for installation and reuse based on its documentation; (b) An example of Intelligent Assistant architecture: the shared environment properties, the means of interactions between assistant and the environment as manifested in the observation and action space, and the assistant components as how an assistant acts via its policy while tracking the past in memory and how it learns how to plan and act

generative AI, particularly Large Language Models (LLMs). The possibility of using networks of LLM-based Assistants [1] to solve of interacting with documentation, generating code and performing complex reasoning tasks. These LLM-powered assistants have greatly shown strong potential in automating scientific workflows [10] and software engineering activities such as bug fixing, test generation, and code synthesis [11].

Despite these potential advancements, the usage of IAs to automated reuse from documentation remains an open challenge (central research problem). It requires not only natural language understanding, but also robust planning, disambiguation, and execution capabilities. Documentation—especially when unstructured, unintelligible, or inconsistent—demands that assistants interpret complex instructions, fill in implicit semantic knowledge, and reason through a plan or conflicting sequential steps. Automating this process involves constructing executable representations of intent from loosely defined human-generated procedural narratives.

To date, there has been no research study of how we can develop and evaluate a method to aid in the RS reuse process from its documentation automatically. This gap produces a substantial obstacle for researchers who need a reliable and scalable solution for reusing RS across a wide variety of domains. Given the critical role that software plays in the scientific research-and the growing serious possibility of using networks of AI-based agents (particularly those equipped by contemporary LLMs), this research seeks to answer a key question: how can we develop and evaluate an AI-based Intelligent Assistant methodology capable of automatically supporting the reuse of research software exclusively from its documentation?

2. Background and related work

Prior related work can be divided under three major topics: Software reuse in the research community, automated approaches for RS reuse, and LLMs as Intelligent Assistants:

2.1. Software reuse in the research community

The benefits of reusing software (e.g., reducing duplication of effort) —are broadly acknowledged since early days in the software engineering field [7]. In spite of its promise, RS reuse has not become standard practice in RS development yet [12, 13]. Systematic efforts to enhance the reusability of digital scientific objects have evolved over time. In early 2000s, the Semantic Web initiative, under the power movement of World Wide Web Consortium (W3C), introduced standards such as the Resource Description Framework (RDF) and the Web Ontology Language (OWL) [14] to convert heterogeneous

[1]In this thesis, intelligent assistant (IA) (or Agent) is a computer system endowed with artificial intelligence and/or machine learning techniques capable of intelligently assisting researchers [8, 9]

knowledge across the web infrastructure into machine-readable format. Following these foundations, the 2014 Lorentz Conference on the FAIR Principles—Findable, Accessible, Interoperable, and Reusable—marked a single milestone in aligning good practices for representing digital scientific objects with the aim of solving their reuse problem [15]. In 2022, the research software community adapted the FAIR principles to create FAIR Principles for research software [16] in order to maximize its value.

Over a decade later, many individuals in the research community still regards RS reuse as *potentially* a powerful means of improving the research productivity and improving software engineering practices in scientific domains [17]. Among others, research initiatives such as Codemeta, SoFAIR, ADORE.software and EVERSE have emerged recently to enhance research software reuse by standardizing metadata, automating lifecycle management, and promoting software quality.

2.2. Automated approaches for RS reuse from documentation

In an effort to reduce human intervention in the reuse process, understanding how to enable machines to learn effective representation from natural language have always enjoyed academic interest using formal methods [4], traditional machine learning models [18], symbolic approaches [19, 20], and, more recently, generative models [2]. Most prior research efforts on the development of automated approaches for supporting software documentation tasks are focused on generating unit test [21], bugs [22] and issue [23, 24] reports with summarization techniques, generating pull requests [25], recommending good practices for ML [26], classifying README content [27] and its simplification [28], documenting program changes [29] as well as checking conflicts and libraries vulnerabilities [30].

2.3. LLMs as Intelligent Assistants

Recent work has initiated to explore the power of LLM-based Intelligent Assistants in software engineering tasks [31] and scientific discovery [32]. Notable among these are typically repository-level tasks [33] that aim to exploit the vast amount of code openly available in repositories. Several research projects explore automated solutions to generate unit test [21], bugs [22] and issue [24] reports with summarization techniques for README files [28] as well as checking conflicts and libraries vulnerabilities [30]. However, a key research challenge remains insufficiently understood, which is how suitable are LLM-powered agents to assist RS reuse from documentation. Recent work has initiated to explore the power of LLM-based IAs in software engineering tasks [34, 35] and scientific discovery [32, 11, 36]. Notable among these are typically classified as LLM-based using a single or multi-agent approach [35, 37, 38]. A newly research trend is shown into the transition from LLMs to Large Action Models (LAM) designed for action generation and execution in real-world scenario is promising, albeit robust empirical studies and formal evaluation framework remains an open question [39].

3. Description of the proposed research

3.1. Research Problem Statement

We define the machine problem of automatic reuse of Research Software (RS) as follows: given access to openly available RS documentation (e.g., a URL to a public Git repository), what actions should a system—such as an artificial intelligent assistant—perform to automatically convert human-generated installation instructions into actionable, machine-executable commands, and execute them within a virtual environment?

Unlike prior work that addresses isolated challenges or narrow tasks, our objective is to explore what kind of grounded language processing model is needed for enabling machines to support researchers in the reuse-tasks of RS, covering the full range of problems summarized in Table 1. A robust solution to enable RS reuse at scale would need to proceed as follows: to overcome the lack of standardization in RS documentation, including inconsistent (or siloed) machinery reuse-procedural narratives (P1 and P2), an machine first would need to extract all software metadata and alternative reuse methods

Table 1
Problems that machines need to address to accelerate scientific discovery through reuse of RS from documentation (e.g., README files): Brief Explanations and representative prior work.

id	Problem	Explanation	Prior Work
P1	Unstructured documentation	Machines lack a formal representation or efficient algorithm to extract install, configure, and execute commands reliably.	[40]
P2	Idiosyncrasies in procedural narratives	If one install-step in a procedure is not machinery-consumable, then these exceptions must be handled, and automation is harder or impossible.	[3]
P3	Complex documentation	This complexity requires substantial additional capabilities for machines (managing methods, dependencies, configurations robustly across systems).	[35]
P4	Lacking instruct-to-action mapping	No structured representation linking natural-language instructions to concise machine operations, preventing robust reuse workflow.	[41]

described in the README (and other files), then transforms each method's sequence of procedural steps into a structured format. To tackle P3 (e.g., automation in managing complex environments and configurations) and P4 (e.g., automated solutions for execution of RS), the research community utilises continuous and development solutions such as Github-based features; however these are relying on permission and might not be fully automated. Therefore, a machine would need to provide detailed thinking process (e.g., adopting the approach introduced in our previous work [2]: the *PlanStep* in which a machine first breaks down a complex install methods found in a README file such "from source" into several subtasks, and then apply reasoning to generate a plan for each subtask in a fixed, sequential order) before generating the two targeted outputs: i) an isolated environment (e.g., via Docker or virtual environments) and another to configure and install the RS within that environment. Finally, the machine should evaluate outcomes concerning correctness, reliability, and accuracy (see Figure 1).

3.2. Research Hypothesis

In light of the problems and potential solutions outlined above, the following central research hypothesis is proposed:

"An Intelligent Assistant, powered by AI methods, particularly Large Language Models, can be designed to autonomously interpret, reason and act upon research software documentation—such as README files—by extracting, transforming, and executing procedural instructions. The assistant can generalize across domains by modeling reuse tasks within documentation as sequences of machine commands and developing automation strategies to execute them accurately."

3.3. Research Questions

The main research question (RQ) addressed by this thesis can be formulated as follows:

RQ: *How can we develop and evaluate an AI-based Intelligent Assistant that is capable of automatically supporting the reuse of research software exclusively from its documentation?*

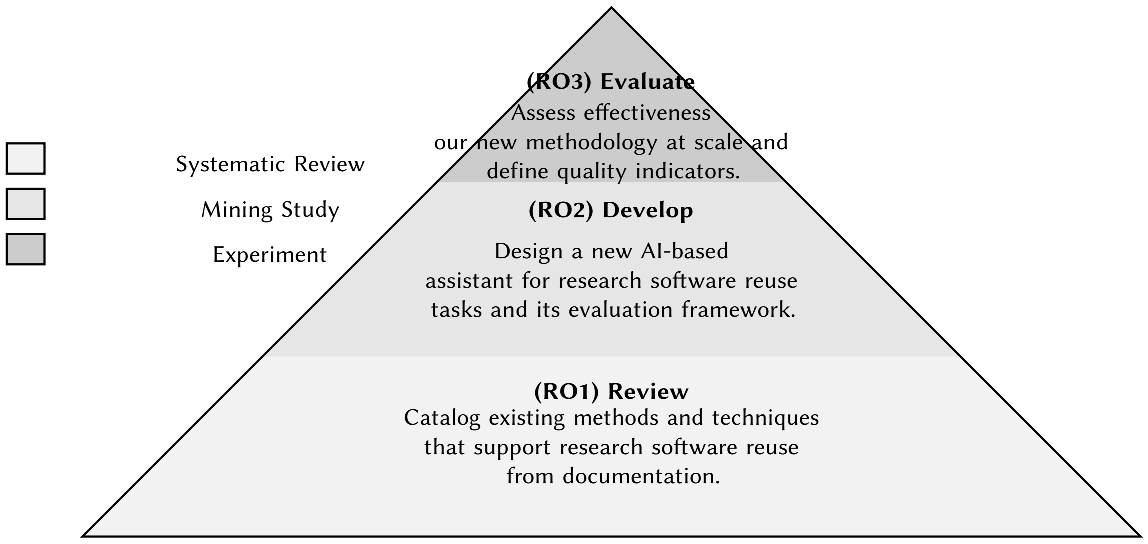
We further decompose the main RQ into various sub-questions (Sub-RQs):

- **Sub-RQ1:** What AI-based techniques and methodologies are suitable for automating the extraction, transformation, and execution of reuse instructions from research software documentation?
- **Sub-RQ2:** How can these techniques and methodologies be evaluated considering the hierarchical structure of reuse tasks and the complexity of documentation?

- **Sub-RQ3:** What system architecture is effective for enabling AI-based assistants to interpret, reason, plan, and execute reuse tasks from research software documentation?
- **Sub-RQ4:** What evaluation framework and quality indicators are needed to assess the AI-based assistant's performance across diverse software and documentation types?

3.4. Research Objectives

To address the main RQ and subsequent sub-research questions (Sub-RQs), this thesis will aim to achieve the following objectives (depicted in Figure 2): **(RO1) Review** and catalog existing methods that support software reuse from documentation, including their core techniques and evaluation strategies; **(RO2) Develop** and implement an AI-based Intelligent Assistant methodology capable of autonomously assisting with a range of reuse tasks from its documentation; and **(RO3) Evaluate** and validate empirically the methodology to assess its effectiveness in executing reuse-related tasks at scale as well as defining and applying relevant quality indicators to research software documentation and (re)-usability.



Systematic Review
Mining Study
Experiment
(RO3) Evaluate
Assess effectiveness our new methodology at scale and define quality indicators.
(RO2) Develop
Design a new AI-based assistant for research software reuse tasks and its evaluation framework.
(RO1) Review
Catalog existing methods and techniques that support research software reuse from documentation.

Figure 2: Research Objectives (RO's) and its proposed empirical methods (grey-scale).

3.5. Research Methodology

To tackle the above research questions, we apply the Design Science Research (DSR) methodology [42], which relies on an iterative process of investigating the problem, generating a solution, implementing it, and evaluating its effectiveness in a research context. Our research objectives previously presented, align with the DSR approach, as it strives to create a method designed to tackle the challenge of automating reuse processes in research software from documentation.

In our research context, this DSR methodology is structured into three unique phases (Ps) as illustrated in Figure 3: **(1) Review and collect prior work:** a comprehensive systematic literature review on existing methods that support software reuse from documentation, including their core techniques and evaluation strategies; **(2) design and implement an AI-based Intelligent Assistant** framework capable of autonomously assisting with a range of reuse tasks (e.g., installation, configuration and execution of research software); **(3) Evaluation and Validation:** empirically evaluate and validate the methodology to assess its effectiveness in executing reuse-related tasks at scale as well as defining and applying relevant quality indicators to research software documentation and (re)-usability.

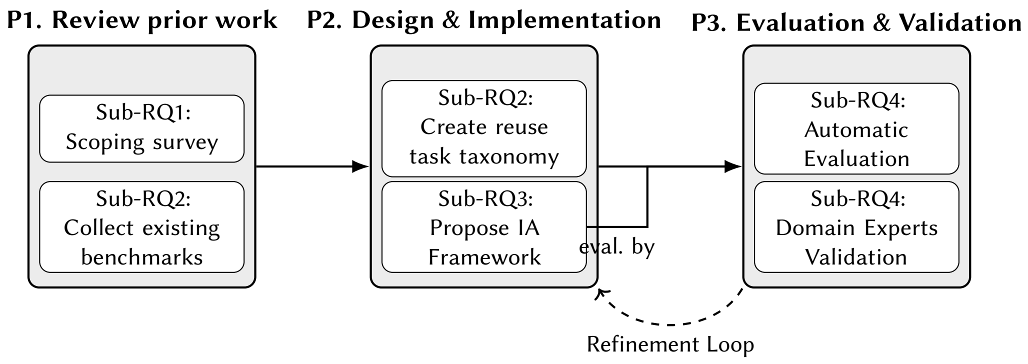

P1. Review prior work
P2. Design & Implementation
P3. Evaluation & Validation
Sub-RQ1: Scoping survey
Sub-RQ2: Collect existing benchmarks
Sub-RQ2: Create reuse task taxonomy
Sub-RQ3: Propose IA Framework
Sub-RQ4: Automatic Evaluation
Sub-RQ4: Domain Experts Validation
eval. by
Refinement Loop

Figure 3: PhD research workflow illustrating the modular stages and their corresponding sub-research questions (Sub-RQs)

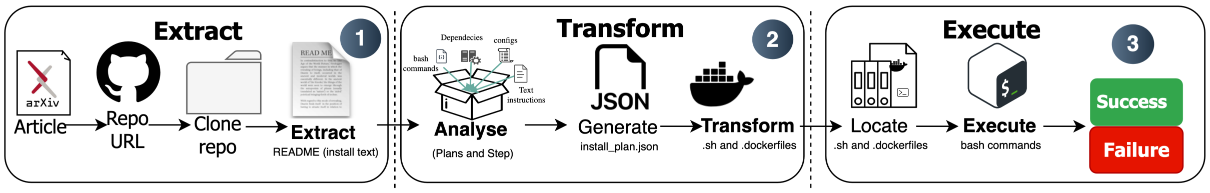


Figure 4: A high-level overview of ETE Agent Workflow - An illustration of how ETE agent make progress from initial extraction stage (1) through transformation of these outputs into machine executable files (2), and to subsequently the execution of these files into a local environment (3)

4. Proposed Research Methods

This thesis investigates the feasibility of adapting AI-based techniques for the reuse scenario from documentation, and how to evaluate them. To answer the previously defined research question, we propose the following empirical methods:

1. **Systematic Literature Review:** This review will provide a foundational and comprehensive overview of existing methods for automating research software reuse from documentation, propose a taxonomy of reuse tasks to be accomplished by an assistant, and define the evaluation criteria.
2. **Quantitative Analysis and Mining Study**: The goal of this study is to characterise research software documentation complexity, extract reuse-relevant elements, identify their properties such as installation plans and steps associated, and formulate a taxonomy of tasks that reflect real-world reuse activities across different scientific domains. Based on these findings, we will create a benchmark and evaluation corpus for assessing intelligent assistants in reuse scenarios. This benchmark will consist of an automated approach to annotate software documentation in real-world reuse scenarios. It is expected to use this benchmark to evaluate task performance in areas such as instruction extraction, interpretation, planning, reasoning and automated execution.
3. **Experiments**: The goal of this computational experiment is to evaluate the suitability of AI-based Intelligent Assistants to aid in the reuse-task of researchers in real-world scenarios. We have recently explored a first minimal prototype of a Large Language Model (LLM)-based agent—designed to assist researchers in research software reuse by extracting human-generated installation instructions from documentation (e.g., GitHub README files), transforming them into structured sequential steps, and executing them in a virtual environment (so called ETE-Agent [2] which architectural approach is shown in Figure 4).

[2]The proposed agent is publicly available at https://github.com/carlosug/agent.rse

Acknowledgments

This work is supported by the Ontology Engineering Group (OEG) under the PhD in Artificial Intelligence Program with Universidad Politécnica de Madrid, and through the exceptional support of the research team supervisor Dr. Prof. Daniel Garijo and director Dr. Prof. Oscar Corcho. The author would also like to warmly thank the mentors for their wise and thoughtful comments, which significantly helped improve the quality and clarity of this work.

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

[1] N. P. Chue Hong, D. S. Katz, M. Barker, A.-L. Lamprecht, C. Martinez, F. E. Psomopoulos, J. Harrow, L. J. Castro, M. Gruenpeter, P. A. Martinez, T. Honeyman, FAIR Principles for Research Software (FAIR4RS Principles) (2021). doi:10.15497/RDA00068.

[2] C. Utrilla Guerrero, O. Corcho, D. Garijo, Automated Extraction of Research Software Installation Instructions from README Files: An Initial Analysis, Lecture Notes in Computer Science 14770 LNAI (2024) 114–133. doi:10.1007/978-3-031-65794-8_8.

[3] H. Gao, C. Treude, M. Zahedi, "Add more config detail": A Taxonomy of Installation Instruction Changes, 2023. URL: http://arxiv.org/abs/2312.03250.

[4] L. Salerno, C. Treude, P. Thongtanunam, P. Thongtatunam, Challenges and Strategies For Novice Developers, Technical Report, 2024.

[5] S. Yuan, K. Song, J. Chen, X. Tan, Y. Shen, R. Kan, D. Li, D. Yang, EASYTOOL: Enhancing LLM-based Agents with Concise Tool Instruction, 2024. URL: http://arxiv.org/abs/2401.06201.

[6] M. M. Wagner, W. R. Hogan, J. D. Levander, M. Diller, Towards machine-fair: Representing software and datasets to facilitate reuse and scientific discovery by machines, Journal of biomedical informatics 154 (2024) 104647.

[7] P. Naur, B. Randell, Software engineering: Report on a conference by the nato science commitee, NATO Scientific Affairs Division, Brüssel (1968).

[8] R. S. Sutton, A. G. Barto, et al., Reinforcement learning: An introduction, volume 1, MIT press Cambridge, 1998.

[9] S. J. Russell, P. Norvig, Artificial intelligence: a modern approach, pearson, 2016.

[10] S. M. Narayanan, J. D. Braza, R.-R. Griffiths, A. Bou, G. Wellawatte, M. C. Ramos, L. Mitchener, S. G. Rodriques, A. D. White, Training a scientific reasoning model for chemistry, arXiv preprint arXiv:2506.17238 (2025).

[11] Z. Luo, Z. Yang, Z. Xu, W. Yang, X. Du, Llm4sr: A survey on large language models for scientific research, arXiv preprint arXiv:2501.04306 (2025).

[12] C. Goodwin, S. Woolley, Barriers to device longevity and reuse: A vintage device empirical study, Journal of Systems and Software 211 (2024) 111991.

[13] M. Karimzadeh, M. M. Hoffman, Top considerations for creating bioinformatics software documentation, Briefings in Bioinformatics 19 (2018) 693–699. doi:10.1093/bib/bbw134.

[14] N. Shadbolt, T. Berners-Lee, W. Hall, The semantic web revisited, IEEE intelligent systems 21 (2006) 96–101.

[15] M. D. Wilkinson, M. Dumontier, I. J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J.-W. Boiten, L. B. da Silva Santos, P. E. Bourne, et al., The fair guiding principles for scientific data management and stewardship, Scientific data 3 (2016) 1–9.

[16] M. Barker, N. P. Chue Hong, D. S. Katz, A. L. Lamprecht, C. Martinez-Ortiz, F. Psomopoulos, J. Harrow, L. J. Castro, M. Gruenpeter, P. A. Martinez, T. Honeyman, Introducing the FAIR Principles for research software, Scientific Data 9 (2022). doi:10.1038/s41597-022-01710-x.

[17] M. David, M. Colom, D. Garijo, L. J. Castro, V. Louvet, E. Ronchieri, M. Torquati, L. del Caño, L. Cerlane, M. Van den Bossche, I. Campos, R. Di Cosmo, Ensure Software Quality, Technical Report, 2024. doi:10.5281/ZENODO.10723608.

[18] G. A. Randrianaina, D. E. Khelladi, O. Zendra, M. Acher, Options Matter: Documenting and Fixing Non-Reproducible Builds in Highly-Configurable Systems (2024).

[19] R. Celebi, J. Rebelo Moreira, A. A. Hassan, S. Ayyar, L. Ridder, T. Kuhn, M. Dumontier, Towards FAIR protocols and workflows: the OpenPREDICT use case, PeerJ Computer Science 6 (2020) e281. doi:10.7717/peerj-cs.281.

[20] K. Belhajjame, J. Zhao, D. Garijo, M. Gamble, K. Hettne, R. Palma, E. Mina, O. Corcho, J. M. Gómez-Pérez, S. Bechhofer, G. Klyne, C. Goble, Using a suite of ontologies for preserving workflow-centric research objects, Journal of Web Semantics 32 (2015) 16–42. URL: https://www.sciencedirect.com/science/article/pii/S1570826815000049. doi:10.1016/j.websem.2015.01.003.

[21] B. Li, C. Vendome, M. Linares-Vasquez, D. Poshyvanyk, N. A. Kraft, Automatically Documenting Unit Test Cases, Proceedings - 2016 IEEE International Conference on Software Testing, Verification and Validation, ICST 2016 (2016) 341–352. doi:10.1109/ICST.2016.30.

[22] S. Rastkar, G. C. Murphy, G. Murray, Automatic summarization of bug reports, IEEE Transactions on Software Engineering 40 (2014) 366–380. doi:10.1109/TSE.2013.2297712.

[23] N. Nikeghbal, A. H. Kargaran, A. Heydarnoori, GIRT-Model: Automated Generation of Issue Report Templates, Proceedings - 2024 IEEE/ACM,, MSR 2024 (2024). doi:10.1145/3643991.3644906.

[24] G. Sridhara, E. Hill, D. Muppaneni, L. Pollock, K. Vijay-Shanker, Towards automatically generating summary comments for Java methods, ASE'10 - Proceedings of the IEEE/ACM International Conference on Automated Software Engineering (2010) 43–52. doi:10.1145/1858996.1859006.

[25] Z. Liu, X. Xia, C. Treude, D. Lo, S. Li, Automatic generation of pull request descriptions, Proceedings - 2019 34th IEEE/ACM International Conference on Automated Software Engineering (2019). URL: https://dl.acm.org/doi/10.1109/ASE.2019.00026.

[26] L. Cabra-Acela, A. Mojica-Hanke, M. Linares-Vásquez, S. Herbold, On using information retrieval to recommend machine learning good practices for software engineers, in: Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, 2023, pp. 2142–2146.

[27] S. Ikeda, �Akinori, I. �Raula, G. Kula, K. Matsumoto, An Empirical Study on README contents for JavaScript Packages (2018). URL: https://www.npmjs.com/package/express.

[28] H. Gao, C. Treude, M. Zahedi, Evaluating Transfer Learning for Simplifying GitHub READMEs, in: Proceedings of the 31st ACM Joint European Software Engineering Conference, 2023. doi:10.1145/3611643.3616291.

[29] R. P. L. Buse, W. R. Weimer, Automatically documenting program changes, in: Proceedings of the 25th IEEE/ACM International Conference on Automated Software Engineering, Association for Computing Machinery, 2010. doi:10.1145/1858996.1859005.

[30] H. O. Delicheh, A. Decan, T. Mens, Quantifying Security Issues in Reusable JavaScript Actions in GitHub Workflows, Proceedings - 2024 IEEE/ACM 21st International Conference on Mining Software Repositories, MSR 2024 (2024) 692–703. doi:10.1145/3643991.3644899.

[31] M. Pezzè, S. Abrahão, B. Penzenstadler, D. Poshyvanyk, A. Roychoudhury, T. Yue, A 2030 roadmap for software engineering, ACM Transactions on Software Engineering and Methodology (2025).

[32] A. Ghafarollahi, M. J. Buehler, Sciagents: Automating scientific discovery through multi-agent intelligent graph reasoning, arXiv preprint arXiv:2409.05556 (2024).

[33] R. Bairi, A. Sonwane, A. Kanade, V. D. C, A. Iyer, S. Parthasarathy, S. Rajamani, B. Ashok, S. Shet, CodePlan: Repository-level Coding using LLMs and Planning, 2023. URL: http://arxiv.org/abs/2309.12499.

[34] J. Liu, K. Wang, Y. Chen, X. Peng, Z. Chen, L. Zhang, Y. Lou, Large language model-based agents for software engineering: A survey, arXiv preprint arXiv:2409.02977 (2024).

[35] I. Bouzenia, P. Devanbu, M. Pradel, Repairagent: An autonomous, llm-based agent for program repair, arXiv preprint arXiv:2403.17134 (2024).

[36] Y. Yamada, R. T. Lange, C. Lu, S. Hu, C. Lu, J. Foerster, J. Clune, D. Ha, The ai scientist-v2: Workshop-

level automated scientific discovery via agentic tree search, arXiv preprint arXiv:2504.08066 (2025).

[37] B. Liu, X. Li, J. Zhang, J. Wang, T. He, S. Hong, H. Liu, S. Zhang, K. Song, K. Zhu, et al., Advances and challenges in foundation agents: From brain-inspired intelligence to evolutionary, collaborative, and safe systems, arXiv preprint arXiv:2504.01990 (2025).

[38] J. He, C. Treude, D. Lo, Llm-based multi-agent systems for software engineering: Literature review, vision and the road ahead, ACM Transactions on Software Engineering and Methodology (2024).

[39] L. Wang, F. Yang, C. Zhang, J. Lu, J. Qian, S. He, P. Zhao, B. Qiao, R. Huang, S. Qin, et al., Large action models: From inception to implementation, arXiv preprint arXiv:2412.10047 (2024).

[40] A. Mao, D. Garijo, S. Fakhraei, Somef: A framework for capturing scientific software metadata from its documentation, in: 2019 IEEE International Conference on Big Data (Big Data), 2019, pp. 3032–3037. doi:10.1109/BigData47090.2019.9006447.

[41] D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi, et al., Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, arXiv preprint arXiv:2501.12948 (2025).

[42] K. Peffers, T. Tuunanen, M. A. Rothenberger, S. Chatterjee, A design science research methodology for information systems research, Journal of management information systems 24 (2007) 45–77.