Envisioning Intelligent Requirements Engineering via Knowledge-Guided Multi-Agent Collaboration

Jiangping Huang[1*], Dongmin Jin[2*], Weisong Sun[3], Yang Liu[3], Zhi Jin[4†]
[1]School of Computer Science and Technology, Chongqing University of Posts and Telecommunications, China
[2]Key Lab of High Confidence Software Technology, Ministry of Education, Peking University, China
[3]College of Computing and Data Science, Nanyang Technological University, Singapore
[4]School of Computer Science, Wuhan University, China
huangjp@cqupt.edu.cn, dmjin@stu.pku.edu.cn, {weisong.sun, yangliu}@ntu.edu.sg, zhijin@whu.edu.cn

***Abstract*—Requirements Engineering (RE) is an initial and critical phase in software development, with the aim of producing well-defined software requirements specifications (SRSs) from rough ideas of clients. It involves multiple tasks (e.g., elicitation, analysis) and roles (e.g., interviewer, analyst). With the rise of Large Language Models (LLMs), many studies have leveraged LLMs to support specific RE tasks. However, existing LLM-based agents often lack domain knowledge integration and fall short in simulating the complex collaboration of human experts across the full RE process. To address this gap, we propose KGMAF, a knowledge-guided multi-agent framework designed to assist requirements engineers in developing high-quality SRSs. KGMAF comprises six LLM-based agents and a shared artifact pool. Each agent is equipped with predefined actions, dedicated functions, and injected knowledge tailored to specific RE tasks. The artifact pool stores both intermediate and final artifacts, serving as a communication channel for inter-agent collaboration. A human-in-the-loop (HITL) mechanism is embedded to guide and validate agent outputs. We present the design of KGMAF, along with preliminary experiments and a case study to demonstrate its practicality. This work lays the foundation for future research on knowledge-driven multi-agent collaboration in RE and highlights key challenges in building trustworthy intelligent assistants for real-world RE tasks.**

***Index Terms*—intelligent requirements development, requirements engineering, large language models, multi-agent collaboration, knowledge injection, human-in-the-loop.**

I. Introduction

Requirements Engineering (RE) plays a foundational role in the success of software systems, aiming to transform clients' vague ideas into structured, verifiable software requirements specifications (SRSs) [1]–[4]. However, RE remains a cognitively demanding and labor-intensive process that spans multiple phases (e.g., elicitation, analysis, specification, and validation) and involves diverse roles (e.g., analyst, stakeholder, verifier). The high cost of manual RE hinders the realization of end-to-end intelligent software engineering.

The emergence of Large Language Models (LLMs) has reshaped many software engineering tasks, from code generation [5] to requirements modeling [6]. Recently, autonomous LLM-based agents have been developed to simulate human workflows in software development pipelines. Notable examples include ChatDev [7] and MetaGPT [8], which orchestrate agent collaboration across design, implementation, and testing. Despite their innovation, these systems marginally support RE tasks or treat them superficially. Some recent work, such as Elicitron [9] and MARE [10], begins to explore agent-based RE automation. However, these efforts often focus on individual RE activities and neglect the integration of domain knowledge that is essential for capturing intent, uncovering implicit requirements, and ensuring artifact quality.

In contrast to earlier approaches that either underrepresent RE or overlook the role of knowledge, we argue that effective automation of RE requires: (1) multi-agent collaboration to emulate human team dynamics, and (2) the infusion of professional and domain knowledge into the reasoning processes of each agent. In the absence of both coordinated agent collaboration and integrated domain knowledge, existing LLM-based approaches face challenges in addressing the complexity and contextual sensitivity required by real-world RE tasks.

To address this challenge, we propose KGMAF, a knowledge-guided multi-agent framework for intelligent requirements development. KGMAF is designed to operate under a human-in-the-loop (HITL) paradigm and features six role-specialized agents—*Interviewer, End-User, Analyst, Archivist, Reviewer*, and *Deployer*—each equipped with predefined actions and injected task-relevant knowledge. Drawing inspiration from the blackboard architecture [11], KGMAF employs a centralized artifact pool that stores intermediate and final artifacts and serves as a coordination medium for agents to observe changes and decide subsequent actions automatically. The HITL mechanism facilitates quality assurance by involving requirements engineers and stakeholders in the validation and refinement of artifacts produced by agents.

We demonstrate the feasibility of KGMAF through a proof-of-concept implementation applied to a real-world insurance management system. The system autonomously performs RE tasks from input prompts and generates structured SRS drafts and intermediate artifacts, which are further validated by human reviewers. Our case study highlights the promise of knowledge-injected agent collaboration in reducing human effort and improving artifact traceability and quality. We further distill key insights on knowledge-guided agent collaboration, together with challenges and opportunities, to outline directions for advancing intelligent RE with multi-agent systems.

*These authors contributed equally to this work.
†Corresponding author.

2643-1572/25/$31.00 ©2025 IEEE
DOI 10.1109/ASE63991.2025.00334

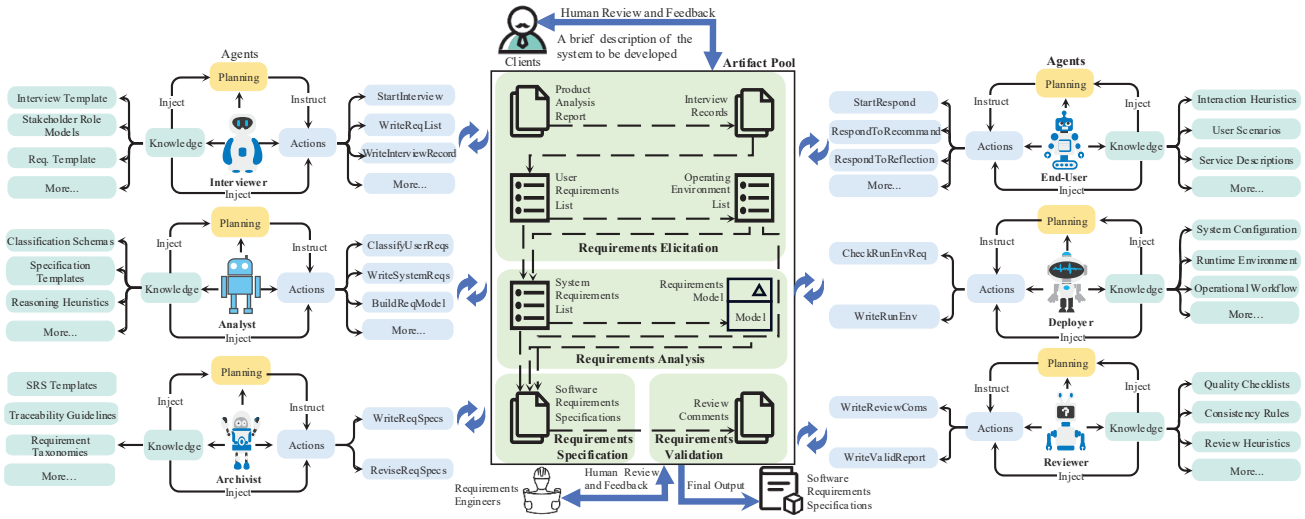

Fig. 1. Overview of the knowledge-guided multi-agent framework for intelligent requirements development. "More..." indicates additional actions or knowledge.

II. Background and Related Work

Automation of RE activities has gained increasing attention for its potential to improve efficiency, consistency, and scalability [12], [13]. These tasks are the core focus of this work. However, automating RE remains challenging due to the need for accurate intent modeling, domain-specific knowledge integration, and structured collaboration. LLMs, such as GPT [14] and LLaMA [15], exhibit strong capabilities in language or code understanding and generation [16]–[20], and have been extended into agent-based systems [21]–[24] that combine planning and execution modules to automate complex tasks. In the RE domain, prior studies have applied LLMs to generate interview scripts [25], stakeholder questions [26], and SRS drafts [27]. While promising, these approaches are largely single-agent and task-specific. In contrast, our proposed KGMAF envisions a collaborative multi-agent framework, where domain-aware agents with specialized roles coordinate to simulate real-world RE workflows and address challenges in automation, consistency, and artifact quality.

III. Conceptual Design of KGMAF

KGMAF transforms a brief project idea into a structured SRS by six agents with a shared artifact pool (Fig. 1).

A. KGMAF *Setting*

In KGMAF, each agent is responsible for specific RE tasks and is designed with three core modules: predefined actions, a planning mechanism, and embedded knowledge. Predefined actions define the agent's capabilities, while the planning mechanism enables context-aware selection of actions based on current artifacts. These behaviors are guided by injected domain knowledge via structured prompts and reasoning strategies [28]–[30]. All agents interact through a shared artifact pool, which serves as the central workspace for storing and updating intermediate and final artifacts. Agents continuously monitor this pool and adapt their actions based on real-time changes, enabling flexible, event-driven collaboration. To ensure quality and address ambiguity, a HITL mechanism allows requirements engineers and clients to review, revise, and validate artifacts. Human feedback is treated as a first-class input that can also trigger further agent actions, supporting iterative human–agent co-evolution of requirements artifacts.

B. *Agent Roles and HITL in* KGMAF

KGMAF consists of six LLM-based agents, each assigned a specialized role in the requirements engineering process. All agents are equipped with structured knowledge and predefined actions, and interact via a shared artifact pool. Their roles and HITL integration are summarized below.

Interviewer. The Interviewer initiates requirements elicitation by conducting structured conversations with the End-User and consulting the Deployer to understand environmental constraints. It produces early artifacts including the Product Analysis Report (PAR), Interview Records (IR), User Requirements List (URL), and Operating Environment List (OEL). Its behavior is guided by elicitation knowledge—e.g., stakeholder models, interview templates, and question-generation strategies—embedded via structured prompts.

End-User. The End-User simulates plausible responses from real users based on role-based prompts and usage scenarios. It contributes raw requirement statements and engages in iterative refinement during the elicitation phase. Its responses reflect both elicitation and contextual knowledge derived from domain manuals, templates, and environmental assumptions.

Deployer. The Deployer collaborates with the Interviewer to define the operational environment of the target system, contributing to the OEL. It encodes technical knowledge on system configuration, runtime constraints, and deployment workflows, drawn from domains such as embedded systems and aerospace. The Deployer ensures deployment assumptions are explicitly captured and aligned with downstream artifacts.

Analyst. The Analyst analyzes elicited requirements, classifies them by type and priority, and synthesizes two key artifacts: the System Requirements List (SRL) and the Requirements Model (RM). It leverages analysis and specification knowledge—such as dependency graphs, modeling

TABLE I
SUMMARY OF KNOWLEDGE CATEGORIES TO SOURCES, EXTRACTION AND INJECTION MECHANISMS

Knowledge	Sources	Extraction Strategy	Injection Mechanism
Elicitation Knowledge	SP, LS, DE	Stakeholder analysis, interview transcript mining, literature pattern recognition	Injected via structured prompt templates, stakeholder role models, and guided question generation rules embedded in dialogue logic to support initial requirement acquisition and clarification.
Analysis Knowledge	SP, LS, DE	Domain modeling, requirement clustering, semantic labeling	Encoded into classification schemas, dependency graphs, conflict detection rules, and prioritization heuristics for interpreting and analyzing requirement artifacts.
Specification Knowledge	SP, LS, DE	Template extraction, taxonomy alignment, model pattern summarization	Integrated into construction workflows using template libraries and requirement taxonomies, including generation rules for structured artifacts, such as SRL, RM, and SRS.
Validation Knowledge	SP, LS, DE	Checklist mining, consistency rule derivation, evaluation heuristic identification	Injected through quality checklists, traceability matrices, and consistency criteria used in artifact-level quality checking and review loop orchestration.
Contextual Knowledge	SP, DE	Configuration rule mining, constraint mapping, environment modeling	Embedded into environment-aware reasoning logic using deployment patterns, constraint libraries, and runtime configuration rules for contextual requirement interpretation.
Cognitive Strategies	DE	Think-aloud protocols, task walkthroughs, abstraction and decision heuristics	Translated into reasoning chains and cognitive rules (e.g., abstraction, prioritization, trade-off analysis) to support abstract modeling, prioritization, and conflict reasoning during requirement processing.

TABLE II
AGENT ROLES, REQUIRED KNOWLEDGE AND ABILITIES IN KNOWLEDGE-DRIVEN RE

Roles	Knowledge	Abilities	Task Definitions within AI/LLM Terms
Interviewer	Elicitation Knowledge	Information Provider	Requirement elicitation via prompt-driven question generation; intent understanding through intent classification.
End-User	Elicitation Knowledge, Contextual Knowledge	Judger, Information Provider	Provide requirement feedback via guided prompts; assess usability using satisfaction classification.
Analyst	Analysis Knowledge, Specification Knowledge, Cognitive Strategies	Content Generator, Translator	Requirement classification for type labeling via text classification; dependency analysis using relation extraction and graph construction; model generation through template-guided text generation (e.g., SRL, RM).
Reviewer	Validation Knowledge, Cognitive Strategies	Judger	Validate artifact consistency using rule-based checks; evaluate requirement quality via checklist-based scoring.
Archivist	Specification Knowledge, Validation Knowledge	Information Provider, Content Generator	Embedded into environment-aware reasoning using deployment patterns, constraint libraries, and runtime configuration rules for contextual requirement interpretation.
Deployer	Contextual Knowledge	Information Provider	Context-aware reasoning under deployment constraints; align requirements with architecture via pattern matching.

templates, and classification schemas—to interpret user intent and formalize requirements for downstream processing.

Archivist. The Archivist consolidates upstream artifacts into a coherent SRS, and continuously maintains it throughout iterative development. Guided by specification and validation knowledge such as SRS templates, requirement taxonomies, and documentation heuristics, the Archivist ensures that the SRS remains structurally consistent, traceable, and aligned with evolving artifacts and stakeholder feedback.

Reviewer. The Reviewer plays a central role in quality assurance, validating the completeness, traceability, and consistency of the SRS. It applies validation knowledge and cognitive strategies, such as review heuristics and quality checklists, to identify potential issues and recommend refinements.

Human-in-the-Loop. After an agent generates an artifact, the user is prompted to review it and provide feedback, after which the agent revises and regenerates the artifact before continuing. This process enables *Requirements Engineers* to check feasibility, clarity, and traceability, and *Clients* to confirm early requirements and business-facing SRS sections. Human input is treated as a first-class update that triggers responsive agent behavior and supports iterative co-evolution.

C. Knowledge Extraction and Injection

To enable knowledge-driven agent collaboration, KGMAF systematically extracts requirements knowledge from three primary sources: existing software projects, RE literature and standards, and domain experts.

From Software Projects (SP). We analyze open-source and industrial repositories to identify common requirement artifacts (e.g., PAR, IR, URL, OEL, etc.), their structural patterns, and inter-artifact dependencies (e.g., IR $\rightarrow$ SRL, RM $\rightarrow$ SRS). These insights provide analysis-oriented knowledge for requirement clustering, traceability, and workflows.

From RE Literature and Standards (LS). Standards such as IEEE Std 830 and ISO/IEC/IEEE 29148:2018 [31], together with textbooks and best practices, offer widely accepted templates, taxonomies, and quality metrics. We extract agent-relevant knowledge (e.g., requirement types, elicitation principles, validation criteria) using natural language processing and semantic parsing to support task planning and evaluation.

From Domain Experts (DE). We adopt cognitive elicitation methods [32]–[35] to capture expert reasoning processes. These include abstraction, prioritization, and conflict resolution strategies, which guide agents in interpreting ambiguity, refining requirements, and maintaining consistency.

Knowledge Categorization and Injection. All extracted knowledge is categorized into six functional types: ❶ *Elicitation*, ❷ *Analysis*, ❸ *Specification*, ❹ *Validation*, ❺ *Contextual*, and ❻ *Cognitive Strategies*. Table I summarizes this mapping. These knowledge types are injected into agents through prompt design, behavioral rules, and memory modules.

Knowledge-Centric Agent Modeling. Each agent is defined by a triplet: *Role, Knowledge, Abilities*. For example, the Interviewer elicits requirements using stakeholder models and guided questioning. Table II shows how different knowledge types align with agent responsibilities, enabling coordinated, knowledge-informed RE collaboration.

IV. CASE STUDY

We conduct a case study on an enterprise-level insurance management system [36] to evaluate the practicality of KGMAF. The system manages client information and employee insurance distribution, with multiple functional modules and

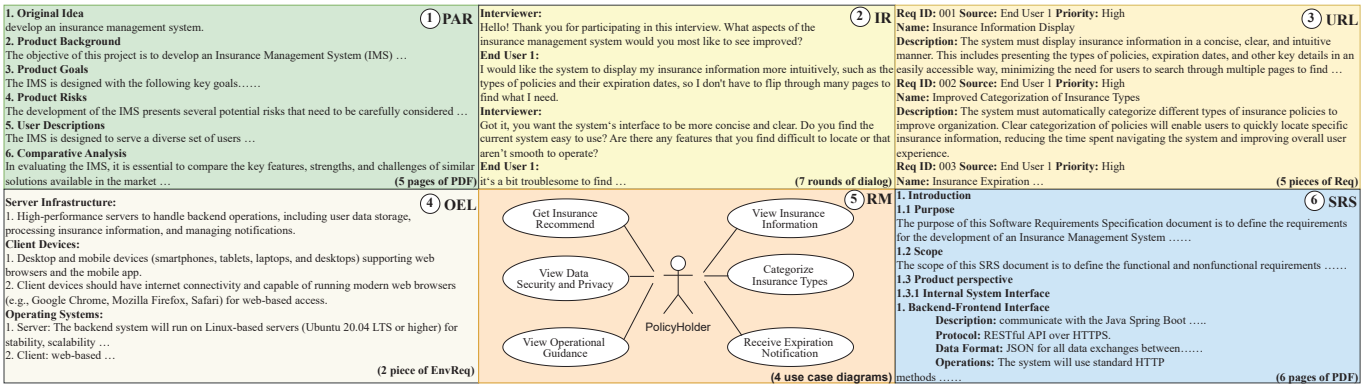

Fig. 2. Artifacts generated by KGMAF on the insurance management system.

diverse requirements. It is selected for its real-world applicability and complexity. Using *GPT-4-turbo-2024-04-09* [37] as the base LLM (*Top-P*=1.0, *Temperature*=0.3, *Max Tokens*=4096, penalties set to 0), we simulate a scenario starting from a brief input—*"I want to develop an insurance management system"*—added to the artifact pool. Agents then operate in observation–action cycles: the **Interviewer** and **End-User** produce the PAR, IR, and URL; the **Deployer** adds the OEL; the **Analyst** generates the SRL and RM; the **Archivist** composes the SRS; and the **Reviewer** conducts iterative validation. Fig. 2 shows representative outputs across seven stages. Results are fully automated to establish a baseline; HITL variants are available at [38]. Most generated artifacts were consistent with the intended scope, though occasional redundancies or conflicts required correction at HITL checkpoints. Both human feedback and injected knowledge (e.g., checklists and templates) improved consistency and structure, while version control mitigates developer effort by emphasizing the final SRS and retaining intermediate artifacts only for traceability. Future work will further address cognitive load through memory mechanisms and summarization, and will establish evaluation metrics such as accuracy, completeness, conflict rate, and review effort to assess practical usefulness.

V. Insights, Challenges, and Opportunities

We distill three key insights from our study, along with corresponding challenges and future opportunities.

(1) *Knowledge is Foundational to Intelligent RE.* **Insight:** Structured, domain-specific knowledge is critical for LLM-based agents to generate relevant and accurate requirements. Without it, outputs may appear fluent yet misaligned with stakeholder intent or system context. **Challenges:** How can diverse knowledge types be systematically extracted, represented, and injected into agents? **Opportunities:** Future work can explore automated pipelines for knowledge acquisition from heterogeneous sources, semantic models, and expert-informed planning strategies for improved agent reasoning.

(2) *HITL Remains Indispensable.* **Insight:** While automation can proceed end-to-end, HITL mode is vital for validating artifacts that involve ambiguity, domain trade-offs, or user intent. **Challenges:** How to reduce human effort while ensuring timely intervention? How to design prompts where users can either proceed automatically or review and give feedback before continuation? **Opportunities:** Future work includes adaptive triggers that let agents request human input when confidence is low, dialog-based artifact reviews, and lightweight feedback loops to balance efficiency with control.

(3) *Standardized Agent Communication is Critical.* **Insight:** Multi-agent collaboration in RE requires consistent input–output formats to ensure semantic alignment and artifact compatibility. **Challenges:** How can agents coordinate actions while adhering to strict schemas and minimizing misinterpretation or artifact drift? **Opportunities:** Future research should investigate schema-constrained messaging, contract-based interfaces, and runtime validation mechanisms to support robust agent interaction and artifact integrity.

VI. Conclusion

This paper introduces KGMAF, a knowledge-guided multi-agent framework that addresses key limitations of existing LLM/agent-based RE approaches via six specialized agents and a shared artifact pool. To simulate expert behavior, agents are equipped with domain and process knowledge obtained through multi-source extraction and injected via prompt engineering and knowledge retrieval. KGMAF also incorporates HITL to enhance artifact quality and reliability. A case study illustrates its real-world applicability. We conclude by identifying future work in advancing knowledge modeling, strengthening human–agent collaboration, and developing standardized multi-agent communication to better support intelligent RE.

Acknowledgment

This work was supported by the Science and Technology Research Program of Chongqing Municipal Education Commission (Grant No. KJQN202500631), the Humanities and Social Sciences Fund of the Ministry of Education (Grant No. 20YJCZH047), by the National Natural Science Foundation of China (Grant No. 62192731, 62192734), and the National Research Foundation, Singapore, and DSO National Laboratories under the AI Singapore Programme (AISG Award No: AISG4-GC-2023-008-1B).

REFERENCES

[1] B. H. C. Cheng and J. M. Atlee, "Research directions in requirements engineering," in *Proceedings of the 29th International Conference on Software Engineering – Workshop on the Future of Software Engineering*, Minneapolis, MN, USA, May 23-25 2007, pp. 285–303.

[2] B. Nuseibeh and S. M. Easterbrook, "Requirements engineering: A roadmap," in *Proceedings of the 22nd International Conference on Software Engineering, Future of Software Engineering Track*. Limerick Ireland: ACM, June 4-11 2000, pp. 35–46.

[3] D. L. Parnas, "Software engineering programs are not computer science programs," *IEEE software*, vol. 16, no. 6, pp. 19–30, 1999.

[4] Z. Jin, L. Liu, X. Chen, and T. Li, *Software Requirements Engineering Methods and Practices*. Tsinghua University Press, 2023.

[5] C. Wang, J. Zhang, Y. Feng, T. Li, W. Sun, Y. Liu, and X. Peng, "Teaching code llms to use autocompletion tools in repository-level code generation," *CoRR*, vol. abs/2401.06391, no. 1, pp. 1–13, 2024.

[6] D. Jin, S. Zhao, Z. Jin, X. Chen, C. Wang, Z. Fang, and H. Xiao, "An evaluation of requirements modeling for cyber-physical systems via llms," *arXiv preprint arXiv:2408.02450*, 2024.

[7] C. Qian, W. Liu, H. Liu, N. Chen, Y. Dang, J. Li, C. Yang, W. Chen, Y. Su, X. Cong *et al.*, "Chatdev: Communicative agents for software development," in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2024, pp. 15 174–15 186.

[8] S. Hong, X. Zheng, J. Chen, Y. Cheng, J. Wang, C. Zhang, Z. Wang, S. K. S. Yau, Z. Lin, L. Zhou *et al.*, "Metagpt: Meta programming for multi-agent collaborative framework," *arXiv preprint arXiv:2308.00352*, 2023.

[9] M. Ataei, H. Cheong, D. Grandi, Y. Wang, N. Morris, and A. Tessier, "Elicitron: An llm agent-based simulation framework for design requirements elicitation," 2024. [Online]. Available: https://arxiv.org/abs/2404.16045

[10] D. Jin, Z. Jin, X. Chen, and C. Wang, "Mare: Multi-agents collaboration framework for requirements engineering," *arXiv preprint arXiv:2405.03256*, 2024.

[11] I. D. Craig, "Blackboard systems," *Artificial Intelligence Review*, vol. 2, no. 2, pp. 103–118, 1988.

[12] X. Hou, Y. Zhao, Y. Liu, Z. Yang, K. Wang, L. Li, X. Luo, D. Lo, J. Grundy, and H. Wang, "Large language models for software engineering: A systematic literature review," *ACM Trans. Softw. Eng. Methodol.*, vol. 33, no. 8, Dec. 2024.

[13] J. Mucha, A. Kaufmann, and D. Riehle, "A systematic literature review of pre-requirements specification traceability," *Requirements Engineering*, vol. 29, no. 2, pp. 119–141, June 2024. [Online]. Available: https://doi.org/10.1007/s00766-023-00412-z

[14] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901.

[15] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample, "Llama: Open and efficient foundation language models," 2023.

[16] W. Sun, Y. Miao, Y. Li, H. Zhang, C. Fang, Y. Liu, G. Deng, Y. Liu, and Z. Chen, "Source code summarization in the era of large language models," in *2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE)*, 2025, pp. 1882–1894.

[17] W. Sun, Y. Chen, M. Yuan, C. Fang, Z. Chen, C. Wang, Y. Liu, B. Xu, and Z. Chen, "Show me your code! kill code poisoning: A lightweight method based on code naturalness," in *2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE)*, 2025, pp. 2663–2675.

[18] W. Sun, Y. Chen, C. Fang, Y. Feng, Y. Xiao, A. Guo, Q. Zhang, Z. Chen, B. Xu, and Y. Liu, "Eliminating backdoors in neural code models for secure code understanding," *Proc. ACM Softw. Eng.*, vol. 2, no. FSE, Jun. 2025. [Online]. Available: https://doi.org/10.1145/3715782

[19] W. Sun, Y. Zhang, J. Zhu, Z. Wang, C. Fang, Y. Zhang, Y. Feng, J. Huang, X. Wang, Z. Jin, and Y. Liu, "Commenting higher-level code unit: Full code, reduced code, or hierarchical code summarization," 2025. [Online]. Available: https://arxiv.org/abs/2503.10737

[20] Y. Chen, W. Sun, C. Fang, Z. Chen, Y. Ge, T. Han, Q. Zhang, Y. Liu, Z. Chen, and B. Xu, "Security of language models for code: A systematic literature review," *ACM Trans. Softw. Eng. Methodol.*, Aug. 2025, just Accepted. [Online]. Available: https://doi.org/10.1145/3735554

[21] Z. Wang, S. Cai, G. Chen, A. Liu, X. Ma, and Y. Liang, "Describe, explain, plan and select: Interactive planning with LLMs enables open-world multi-task agents," in *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. [Online]. Available: https://openreview.net/forum?id=KtvPdGb31Z

[22] Y. Zhang, R. Li, P. Liang, W. Sun, and Y. Liu, "Knowledge-based multi-agent framework for automated software architecture design," in *Proceedings of the 33rd ACM International Conference on the Foundations of Software Engineering*, ser. FSE Companion '25. New York, NY, USA: Association for Computing Machinery, 2025, p. 530–534. [Online]. Available: https://doi.org/10.1145/3696630.3728493

[23] J. Huang, D. Jin, W. Sun, Y. Liu, and Z. Jin, "Knowledge-guided multi-agent framework for automated requirements development: A vision," 2025. [Online]. Available: https://arxiv.org/abs/2506.22656

[24] D. Jin, W. Sun, J. Huang, P. Liang, J. Xuan, Y. Liu, and Z. Jin, "iredev: A knowledge-driven multi-agent framework for intelligent requirements development," 2025. [Online]. Available: https://arxiv.org/abs/2507.13081

[25] B. Görer and F. B. Aydemir, "Generating requirements elicitation interview scripts with large language models," in *Proceedings of the 31st International Requirements Engineering Conference Workshops*. Hannover, Germany: IEEE, September 4-5 2023, pp. 44–51.

[26] H. Hasso, B. Fischer-Starcke, and H. Geppert, "Quest-re question generation and exploration strategy for requirements engineering," in *Proceedings of the 32nd International Requirements Engineering Conference - Workshops*. Reykjavik, Iceland: IEEE, June 24-25 2024, pp. 1–9.

[27] M. Krishna, B. Gaur, A. Verma, and P. Jalote, "Using llms in software requirements specifications: An empirical evaluation," in *Proceedings of the 32nd International Requirements Engineering Conference*. Reykjavik, Iceland: IEEE, June 24-28 2024, pp. 475–483.

[28] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *Advances in neural information processing systems*, vol. 35, pp. 24 824–24 837, 2022.

[29] S. Yao, D. Yu, J. Zhao, I. Shafran, T. Griffiths, Y. Cao, and K. Narasimhan, "Tree of thoughts: Deliberate problem solving with large language models," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[30] P. S. H. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive NLP tasks," in *Proceedings of the 34th Annual Conference on Neural Information Processing Systems*. Virtual Event: Curran Associates, Inc., December 6-12 2020, pp. 1–16.

[31] *ISO/IEC/IEEE 29148:2018 Systems and software engineering — Life cycle processes — Requirements engineering*, International Organization for Standardization, International Electrotechnical Commission, and Institute of Electrical and Electronics Engineers Standard ISO/IEC/IEEE 29 148:2018, 2018, this publication was last reviewed and confirmed in 2024. Therefore this version remains current.

[32] H. A. Linstone and M. Turoff, *The Delphi Method: Techniques and Applications*. Reading, MA: Addison-Wesley, 1975.

[33] R. C. Clark and F. Estes, *Cognitive Task Analysis*. Washington, DC: International Society for Performance Improvement, 2008.

[34] C. H. Lewis, "Using the "thinking aloud" method in cognitive interface design," IBM TJ Watson Research Center, Tech. Rep. RC-9265, 1982.

[35] C. Lewis, P. Polson, C. Wharton, and J. Reiman, "Testing a walkthrough methodology for theory-based design of walk-up-and-use interfaces," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '90)*. Seattle, WA, USA: ACM, 1990, pp. 235–242. [Online]. Available: https://dl.acm.org/doi/10.1145/97243.97279

[36] "The source for the selected case study," https://github.com/lenve/javadoc.

[37] R. OpenAI *et al.*, "Gpt-4 technical report," *ArXiv*, vol. 2303, p. 08774, 2023.

[38] "The generated requirements artifacts in case study," https://anonymous.4open.science/r/KARE-ASE-Vision.