

Agile Software Management with Cognitive Multi-Agent Systems

Konrad Cinkusz^a and Jarosław A. Chudziak^b

Faculty of Electronics and Information Technology, Warsaw University of Technology, Warsaw, Poland
{konrad.cinkusz.stud, jaroslaw.chudziak}@pw.edu.pl

Keywords: Agile Methodologies, Multi-Agent Systems, Large Language Models, Software Project Management, MAS-LLM Integration.

Abstract: This paper explores the integration of cognitive agents powered by Large Language Models (LLMs) into software project management within the Scaled Agile Framework (SAFe). We introduce the CogniSim framework, an ecosystem where virtual agents operate in a simulated software environment to fulfill key roles in IT project development. Emphasis is placed on the adaptability of these agents to the Scrum methodology, particularly in decision-making and problem-solving. By combining LLMs with Multi-Agent Systems (MAS), we focus on improvements in project management, development processes, and Agile methodologies. Through simulations and case studies, we demonstrate advancements in task delegation, communication, and project lifecycle management, highlighting the potential of LLM-augmented MAS to manage software projects with increased precision and intelligence. Our findings provide insights into essential components for an effective cognitive multi-agent ecosystem, including Dynamic Context techniques and Theory of Mind for enhanced agent collaboration, laying the groundwork for future research in this field.

1 INTRODUCTION

The complexity and scale of modern software systems necessitate advanced approaches to software engineering. Agile methodologies like SAFe have become standard practices, emphasizing iterative development, customer collaboration, and flexibility (Dingsøyr et al., 2012). However, effectively managing large-scale, complex projects within these frameworks remains challenging (Perkusich et al., 2020).


Multi-Agent Systems offer a promising solution to these challenges (Talebirad and Nadiri, 2023; Chudziak and Wawer, 2024). MAS consist of networks of autonomous agents that collaborate to achieve defined objectives within their environment (Cruz, 2024). In software engineering, each agent can manage specific aspects of the development lifecycle, such as requirements gathering, code generation, testing, or deployment. The characteristics of MAS—reactivity, proactiveness, and social abilities—make them well-suited for managing complex and dynamic software engineering environments (Li et al., 2023).


Simultaneously, Large Language Models, such as GPT-4, have transformed natural language process-

ing by generating human-like content across various formats (Guo et al., 2023). In software engineering, LLMs can automate routine tasks like code completion, documentation, and debugging, reducing errors and boosting productivity (Barua, 2024). Integrating LLMs with MAS creates cognitive multi-agent ecosystems that leverage the strengths of both technologies (Singhal et al., 2023). For instance, agents responsible for code generation can utilize LLMs to remain aligned with the latest programming paradigms and libraries, ensuring that the software remains current and resilient (Guo et al., 2023).

Effective collaboration among agents is crucial in these ecosystems. Orchestration and choreography paradigms offer strategies for coordinating agent interactions (Cruz, 2024), while frameworks like FIPA (IEEE Foundation for Intelligent Physical Agents (FIPA), 2012) provide specifications for agent communication and interoperability. Techniques from symbolic knowledge management, such as nonmonotonic logic and belief logics, enable agents to handle incomplete and evolving information, enhancing their decision-making capabilities.

PASSI (Process for Agent Societies Specification and Implementation), shown in Figure 1, is a comprehensive methodology designed to guide the development of FIPA-compliant multi-agent systems from

^a  <https://orcid.org/0009-0001-5709-1172>

^b  <https://orcid.org/0000-0003-4534-8652>

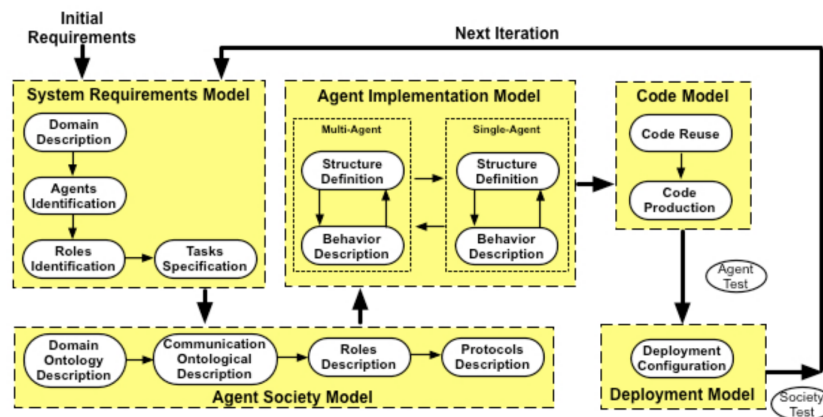


Figure 1: The PASSI design process (IEEE Foundation for Intelligent Physical Agents (FIPA), 2012).

initial requirements to code implementation. It integrates object-oriented software engineering principles with agent-based system design, offering a structured approach for building systems that involve peer-to-peer agent interactions. By referencing PASSI, we situate our framework within a lineage of methodologies that emphasize standardized communication protocols and semantic interactions—ensuring interoperability, scalability, and adherence to established best practices. Our approach extends these fundamentals by leveraging Large Language Models, enabling cognitive agents to augment traditional PASSI-based design principles with enhanced adaptability, richer context-awareness, and more flexible decision-making capabilities within complex, evolving software environments.

Reinforcement learning further improves agent collaboration by allowing agents to learn optimal behaviors through trial and error (Sutton and Barto, 1998), helping them adapt to new challenges and optimize interactions in dynamic, unpredictable environments (Du et al., 2023). Dynamic Context techniques enhance the capabilities of LLM-augmented MAS by allowing agents to adapt their behavior based on real-time data (Du et al., 2024), while incorporating the Theory of Mind (Li et al., 2023; Kosinski, 2024) enables agents to predict and understand the actions and intentions of others, improving overall cooperation (Kim et al., 2024).

In this paper, we introduce the CogniSim framework, where cognitive agents powered by LLMs collaborate within a simulated software environment, assuming key Agile roles in product management, architecture, development, and testing. The framework aligns with Agile methodologies, focusing on the Scaled Agile Framework (Scaled Agile, Inc., 2024) and emphasizing the adaptability of agents to Scrum. By simulating complex workflows and environments, including potential integration with LeSS

(LeSS Company B.V., 2024), DaD (Project Management Institute, 2024), and Nexus (Scrum.org, 2024), CogniSim enables efficient decision-making and context-aware actions. Through simulations and case studies, we highlight how the integration of LLMs and MAS could improve task delegation, communication, and lifecycle management, illustrating the potential to manage software projects with increased precision and intelligence.

2 FRAMEWORK OVERVIEW

Managing modern software projects requires innovative solutions that address the growing demands for adaptability, coordination, and efficiency. The CogniSim framework provides such a solution, integrating cognitive agents powered by Large Language Models within a Multi-Agent System to optimize project management processes in Agile methodologies like Scrum and SAFe.

2.1 Concept of CogniSim

CogniSim could operate within a multi-layered system architecture that integrates both internal and external software ecosystems, forming a cohesive bridge between them. As shown in Figure 2, CogniSim interacts with external systems through various APIs, ensuring the cognitive agents can access real-time data and make informed decisions.

Within the internal environment, agents are structured into distinct roles, each reflecting critical responsibilities commonly found in Agile software management. Product Owners manage backlog prioritization, DevOps Engineers oversee CI/CD pipelines, and the Development Team implements features based on user stories. By augmenting these roles with

LLM-driven capabilities, CogniSim automates routine tasks and optimizes efficiency.

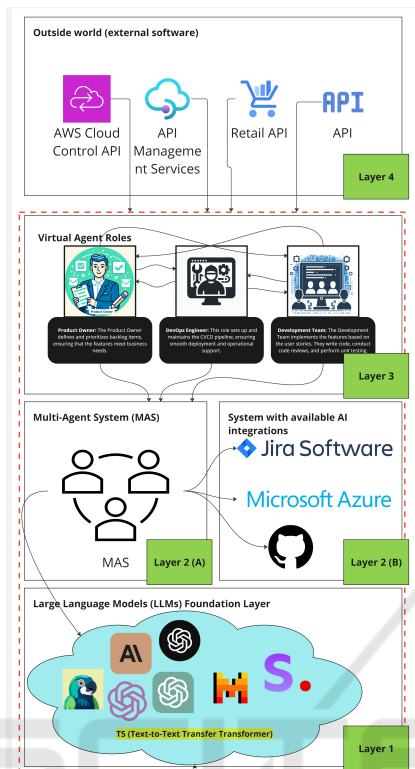


Figure 2: Multilayer concept showing how system is working with external environment.

2.2 Design of CogniSim

CogniSim integrates cognitive agents, driven by LLMs, to emulate human-like decision-making, automate tasks, and streamline workflows through intelligent collaboration. Each agent learns and adapts based on environmental interactions, improving over time via machine learning techniques. By simulating human team dynamics, these agents enhance communication, coordination, and overall project efficiency (Li et al., 2023).

2.3 Components of the CogniSim Framework

Key components include:

- **Cognitive Agents.** AI-driven entities capable of processing natural language, learning from data, and interacting with other agents and stakeholders.
- **Communication Protocols.** Standardized protocols for message exchange and coordination.

- **Decision-Making.** Agents select optimal actions using rule-based, data-driven, or hybrid approaches.
- **Collaboration Tools.** Interfaces that facilitate interaction among agents and between agents and human team members.

2.4 Integration with Agile Methodologies

CogniSim enhances Agile practices by automating routine tasks and offering data-driven insights. In a SAFe context, CogniSim coordinates multiple teams, manages dependencies, and aligns work with the project vision. Figures 3 and 4 show how agents collaborate with human stakeholders during Program Increment (PI) preparation and Scrum iterations. Cognitive agents, representing roles traditionally held by humans, assist in backlog refinement, sprint planning, code implementation, and quality assurance. Over time, agents improve their performance by actively learning from feedback loops provided during the development process.

2.5 Roles and Responsibilities of Agents

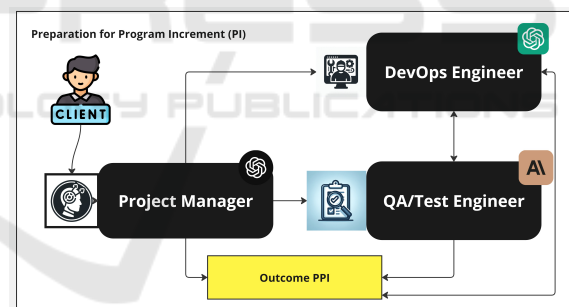


Figure 3: Role of cognitive agents in the preparation phase for Program Increment (PI).

In CogniSim, cognitive agents simulate roles such as:

- **Project Manager:** Manages communication with clients, ensuring requirements are captured and prioritized.
- **DevOps Engineer:** Maintains CI/CD pipelines, deployment processes, and collaborates with QA.
- **QA/Test Engineer:** Defines test cases, automates tests, and ensures product quality.
- **Development Team:** Implements features, coordinates with UX and system teams, and integrates feedback.
- **UX Designer:** Designs user interfaces, working with development to ensure usability.

- **System Team:** Maintains build/testing environments, ensuring integration efforts run smoothly.
- **Customer Representatives:** Provide continuous feedback throughout development.

This automation frees human teams to focus on strategic activities, while agents handle routine tasks and offer real-time insights.

2.6 Benefits of CogniSim

Integrating cognitive agents yields several benefits:

- **Enhanced Decision-Making:** Data-driven insights improve choices throughout the project life-cycle.
- **Increased Efficiency:** Automating routine tasks frees humans for more strategic work.
- **Improved Collaboration:** Natural language processing and role simulation improve communication.
- **Scalability:** Coordinating multiple teams and managing dependencies supports large-scale Agile implementations.

2.7 Use Case Illustration

In a virtual environment, cognitive agents manage project management, DevOps, QA, and development roles. These agents automate code generation, testing, and deployment while adhering to Agile practices like Scrum and SAFe, optimizing workflows and reducing time to market. Figure 4 shows cognitive agents' roles during a Scrum iteration, ensuring continuous integration, feedback, and quality deliverables.

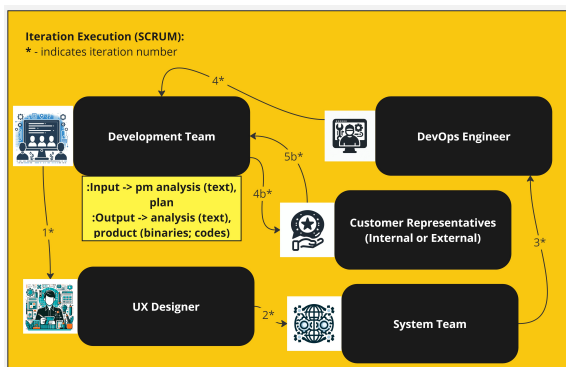


Figure 4: Cognitive agents' roles during Scrum iteration execution.

3 METHODOLOGY

This section outlines the methodology employed to integrate cognitive agents powered by Large Language Models into Agile software project management using CogniSim. We describe the agent-based system design, the development environment, and how virtual agents simulate key roles within SAFe.

3.1 Structure of a Single Virtual Agent

Each virtual agent in CogniSim represents a self-contained unit powered by an AI-driven LLM core. Figure 5 illustrates the structure of a typical agent.

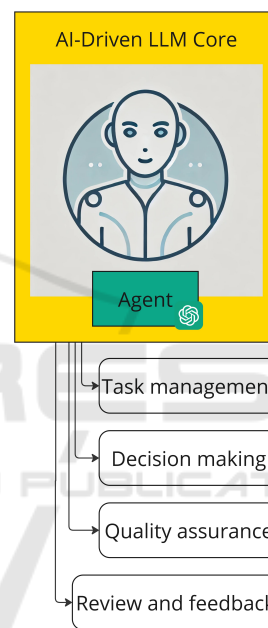


Figure 5: Structure and functionalities of a single virtual agent in CogniSim.

Agents perform:

- **Task Management:** Organizing and prioritizing tasks based on requirements.
- **Decision Making:** Using LLM capabilities to evaluate solutions and provide data-driven decisions.
- **Quality Assurance:** Automated checks ensuring outputs meet quality standards.
- **Review and Feedback:** Offering insights on performance and outcomes for continuous improvement.

This modular approach supports scalability, flexibility, and improved efficiency throughout the project lifecycle.

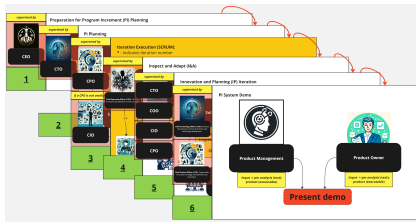


Figure 6: CogniSim MAS powered by LLMs (Cinkusz and Chudziak, 2024).

3.2 Agent-Based System Design

The CogniSim framework, as shown in Figure 6, employs agents driven by advanced LLMs to simulate human-like interactions and decisions, automating tasks traditionally handled by humans (Huang et al., 2024). These agents operate autonomously, collaborating within a MAS to address all phases of the software lifecycle, from backlog refinement to risk assessment. Figure 7 illustrates how the MAS powered by LLMs processes tasks through agent interactions.

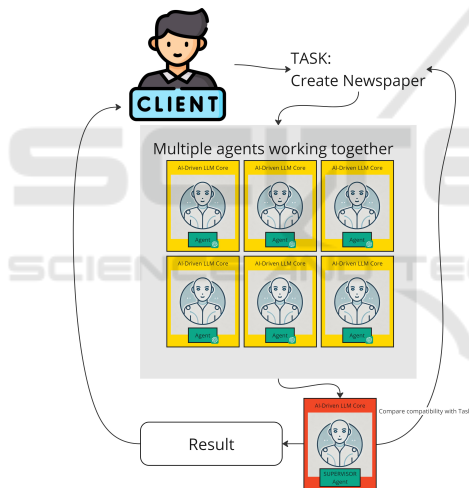


Figure 7: Multi-Agent System powered by LLMs containing simple tasks from a client and a supervisor agent outside the MAS for checking compatibility.

3.3 Development Platform and Tools

CogniSim is implemented in Python, using OpenAI’s GPT-4 and GPT-3.5 models, and LangChain (LangChain, 2023) for LLM integration. Agents are defined using JSON, ensuring flexibility. Tools like Visual Studio Code and libraries such as `tqdm`, `pandas`, and `openai` support development and testing.

Each agent’s behavior within CogniSim is modeled through a layered architecture combining LLM-driven reasoning and structured interaction con-

tracts. Agents are instantiated from JSON configurations that define roles, instructions, and constraints; once initialized, they utilize GPT-3.5 or GPT-4 via LangChain to parse input contexts, generate responses, and reason about tasks. For instance, a Product Management agent might process backlog refinement instructions and produce prioritized user stories, while a System Architect agent could leverage LLM outputs for architectural decisions, referencing code snippets or design patterns.

3.4 Utilizing Virtual Agents in Software Management

Virtual agents simulate key roles—analysts, designers, programmers, testers, project managers—each equipped with prompts that initiate interactions. For instance, Figure 8 shows a prompt for a Business Analyst during PI Planning, guiding their tasks in alignment with Agile processes.

4 EXPERIMENTS AND RESULTS

A series of simulations were conducted to assess the effectiveness of the CogniSim framework in replicating Agile processes within SAFE. The setup included key phases such as Program Increment (PI) Planning and Iteration Execution, with cognitive agents representing roles like Release Train Engineer, Product Management, and System Architect. Their interactions were facilitated through a dialogue system, aligning objectives, prioritizing features, and refining backlogs.

The research questions centered on evaluating agent capabilities in communication, coordination, and decision-making. The multi-agent simulations emphasized communication between agents, reflecting the collaboration required in Agile development. During PI Planning, agents aligned objectives, assessed technical feasibility, and identified dependencies, closely mirroring real-world Agile practices.

Results showed that the cognitive agents successfully replicated communication and decision-making processes inherent in Agile methodologies. They coordinated effectively, shared information efficiently, and aligned their objectives for better planning and execution. The agents provided data-driven insights for feature prioritization and risk mitigation, improving decision-making quality.

Comparisons between agent outputs and human analysts indicated that agents performed at a high level, often matching or exceeding human-like analysis in feature evaluation and risk assessment. These

As a Business Analyst, your role during the PI Planning event is to collaborate with Product Owners to ensure that features are broken down into clear user stories with well-defined acceptance criteria. You work closely with Development Teams to clarify details and answer questions as they estimate and commit to user stories for the Program Increment. Your focus is on ensuring that requirements are understood, dependencies are identified, and that the stories are ready for implementation. Please follow these steps:

1. Analyze the Program Increment goals and break down features into user stories.
2. Define acceptance criteria for each user story.
3. Identify dependencies and potential risks.
4. Communicate with Development Teams to clarify any uncertainties.
5. Update documentation and ensure everything is ready for implementation.

Figure 8: Natural language prompt demonstrating the role of a Business Analyst during PI Planning.

findings suggest that LLM-augmented agents can enhance software project management, enabling more efficient data-driven decisions.

The simulations confirmed that agents adhered to their respective Agile roles, ensuring consistent goal alignment. This supports the system's potential for improving productivity and quality in complex software projects, meeting the study's success criteria.

4.1 Output Quality Analysis

A comprehensive evaluation of the quality and effectiveness of agent-generated content can be achieved through the application of diverse measurement algorithms. Techniques such as cosine similarity, code illustrated in Figure 9, are instrumental in detecting redundant or overlapping information, thereby informing strategies for generating outputs that are more varied and contextually appropriate. Metrics examining lexical and syntactic diversity further ensure that the system avoids producing uniform or overly predictable outputs. Goal achievement metrics, which incorporate keyword matching and semantic similarity, provide a mechanism for determining the alignment of content with predefined objectives.

5 DISCUSSION AND FUTURE WORK

This study explored integrating cognitive agents powered by LLMs into Multi-Agent Systems to enhance Agile software project management frameworks like Scrum and SAFe. The CogniSim framework demonstrated how such technologies can automate complex tasks, improve decision-making, and boost overall productivity by simulating key software development

roles.

Future research should focus on scaling CogniSim for larger teams and more intricate projects, evaluating performance using metrics like sprint completion times, defect rates, and adherence to deadlines. Ensuring interoperability with existing CI/CD pipelines and issue trackers will enhance practical applicability (Horling and Lesser, 2004).

CogniSim's agent architecture employs a modular design, where cognitive agents integrate LLM-driven reasoning layers with domain-specific knowledge modules and standardized interfaces. These agents communicate using established protocols (like FIPA ACL) and optional extensions tailored to Agile processes, ensuring smooth collaboration and interoperability. Although certain workflows (e.g., continuous integration and testing) operate through largely stable pipelines, CogniSim supports dynamic adjustments, allowing agents to reconfigure their pipelines based on feedback, evolving requirements, or performance metrics, thereby enhancing resilience and adaptability throughout the software lifecycle.

Improving human-agent collaboration through intuitive interfaces and adaptive MAS architectures will further align the system with various Agile practices (Guo et al., 2024). Incorporating adaptive learning capabilities may help agents respond more effectively to individual team member styles, enhancing overall team efficiency. Recent advances explore the synergy of logical reasoning, long-term memory, and collaborative intelligence within multi-agent LLM ecosystems (Kostka and Chudziak, 2024).

Unlike existing multi-agent Agile management frameworks such as PASSI (IEEE Foundation for Intelligent Physical Agents (FIPA), 2012), which rely on predefined coordination rules and heuristic decision-making, CogniSim integrates Large Language Model-driven cognitive capabilities. This inte-

```

from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

def compute_cosine_similarity(messages):
    corpus = [msg['message'] for msg in messages]
    vectorizer = TfidfVectorizer(stop_words='english')
    tfidf_matrix = vectorizer.fit_transform(corpus)
    similarities = []
    for i in range(1, tfidf_matrix.shape[0]):
        sim = cosine_similarity(tfidf_matrix[i-1], tfidf_matrix[i])[0][0]
        similarities.append(sim)
    return similarities

# Example call
similarities = compute_cosine_similarity([{'message': 'Hello world'}, {'message':
↪ 'Hello there!'}])

```

Figure 9: Code snippet for computing content similarity between messages using cosine similarity.

gration enhances adaptability, linguistic understanding, and decision quality while aligning more closely with widely adopted Agile practices. Maintaining transparency in decision-making and trust remains a priority, along with ensuring ongoing updates to adhere to ethical standards (Tariverdi, 2024).

Implementing CogniSim in real-world case studies will provide insights into its impact on productivity, quality, and team dynamics, while testing it across diverse projects will assess scalability and effectiveness (Chudziak and Cinkusz, 2024). Embedding predictive analytics within MAS could anticipate delays or issues, enabling proactive corrective actions in dynamic environments (Lin et al., 2024).

6 CONCLUSION

Integrating cognitive agents powered by Large Language Models within Multi-Agent Systems presents a significant step forward in Agile software project management. The CogniSim framework demonstrated that these technologies automate routine tasks, enhance decision-making, and improve collaboration among virtual team members, increasing efficiency and adaptability (Dignum, 2009).

By simulating roles such as project managers, developers, and QA engineers, cognitive agents manage complex tasks traditionally handled by humans, allowing human teams to focus on strategic activities requiring creativity. Experiments showed that cognitive agents effectively replicated communication and decision-making processes integral to Agile methodologies, often matching or exceeding human performance in certain tasks (Konar, 2000).

Challenges remain in scaling the framework for larger projects, improving adaptability to unexpected changes, and ensuring seamless human-agent collaboration. Addressing ethical considerations—data privacy, security, and transparency—is also critical as these technologies become more integrated into development processes (Müller, 2020).

Future work will refine the framework, validate it in real-world scenarios, and ensure responsible integration, aiming to achieve benefits like improved decision-making and reduced operational costs (Rose, 2020). The advancements in MAS and LLMs within Agile frameworks hold promise for more intelligent, responsive project management practices, enabling teams to adapt swiftly to evolving requirements and deliver higher-quality software products.

By incorporating LLM-powered cognitive agents into multi-agent systems, this research introduces an approach for optimizing Agile software development processes. It combines the adaptability of cognitive computing with coordination strategies inherent to agent-based systems. The study's primary contributions include demonstrating enhanced decision-making capabilities, the automation of repetitive tasks, and improved communication within dynamic and complex project environments.

REFERENCES

- Barua, S. (2024). Exploring autonomous agents through the lens of large language models: A review.
- Chudziak, J. A. and Cinkusz, K. (2024). Towards llm-augmented multiagent systems for agile software engineering. In *Proceedings of the 39th IEEE/ACM In-*

- ternational Conference on Automated Software Engineering (ASE '24), page 2, New York, NY, USA. ACM.
- Chudziak, J. A. and Wawer, M. (2024). Elliottagents: a natural language-driven multi-agent system for stock market analysis and prediction. In *Proceedings of the 38th Pacific Asia Conference on Language, Information and Computation*, Tokyo, Japan.
- Cinkusz, K. and Chudziak, J. A. (2024). Communicative agents for software project management and system development. In *Proceedings of the 21th International Conference on Modeling Decisions for Artificial Intelligence*, Tokyo, Japan.
- Cruz, C. (2024). Transforming competition into collaboration: The revolutionary role of multi-agent systems and language models in modern organizations.
- Dignum, V. (2009). *Handbook of Research on Multi-Agent Systems: Semantics and Dynamics of Organizational Models*. Information Science Reference.
- Dingsøy, T., Nerur, S., Balijepally, V., and Moe, N. B. (2012). A decade of agile methodologies: Towards explaining agile software development. *Journal of Systems and Software*, 85(6):1213–1221.
- Du, H., Li, Z., Niyato, D., Kang, J., Xiong, Z., Huang, H., and Mao, S. (2023). Diffusion-based reinforcement learning for edge-enabled ai-generated content services.
- Du, H., Thudumu, S., Vasa, R., and Mouzakis, K. (2024). A survey on context-aware multi-agent systems: Techniques, challenges and future directions.
- Guo, T., Chen, X., Wang, Y., Chang, R., Pei, S., Chawla, N. V., Wiest, O., and Zhang, X. (2024). Large language model based multi-agents: A survey of progress and challenges.
- Guo, Z., Jin, R., Liu, C., Huang, Y., Shi, D., Supryadi, Yu, L., Liu, Y., Li, J., Xiong, B., and Xiong, D. (2023). Evaluating large language models: A comprehensive survey.
- Horling, B. and Lesser, V. (2004). A survey of multi-agent organizational paradigms. *The Knowledge Engineering Review*, 19(4):281–316.
- Huang, X., Liu, W., Chen, X., Wang, X., Wang, H., Lian, D., Wang, Y., Tang, R., and Chen, E. (2024). Understanding the planning of llm agents: A survey.
- IEEE Foundation for Intelligent Physical Agents (FIPA) (2012). *Design Process Documentation Template*. IEEE FIPA DPDF Working Group.
- Kim, A. G., Muhn, M., and Nikolaev, V. V. (2024). Financial statement analysis with large language models. *Chicago Booth Research Paper, Fama-Miller Working Paper*.
- Konar, A. (2000). *Artificial Intelligence and Soft Computing: Behavioral and Cognitive Modeling of the Human Brain*. CRC Press.
- Kosinski, M. (2024). Evaluating large language models in theory of mind tasks.
- Kostka, A. and Chudziak, J. A. (2024). Synergizing logical reasoning, long-term memory, and collaborative intelligence in multi-agent llm systems. In *Pacific Asia Conference on Language, Information and Computation (PACLIC 38)*, Tokyo, Japan.
- LangChain (2023). Langchain core api reference.
- LeSS Company B.V. (2024). LeSS Framework.
- Li, H., Chong, Y., Stepputtis, S., Campbell, J., Hughes, D., Lewis, C., and Sycara, K. (2023). Theory of mind for multi-agent collaboration via large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.
- Lin, F., Kim, D. J., and Chen, T.-H. P. (2024). When llm-based code generation meets the software development process.
- Müller, V. C. (2020). Ethics of artificial intelligence and robotics. In Zalta, E. N., editor, *Stanford Encyclopedia of Philosophy*, pages 1–70. Metaphysics Research Lab, Stanford University.
- Perkusich, M., Chaves e Silva, L., Costa, A., Ramos, F., Saraiva, R., Freire, A., Dilozenzo, E., Dantas, E., Santos, D., Gorgônio, K., Almeida, H., and Perkusich, A. (2020). Intelligent software engineering in the context of agile software development: A systematic literature review. *Information and Software Technology*, 119:106241.
- Project Management Institute (2024). Introduction to Disciplined Agile (DaD).
- Rose, D. (2020). *Artificial Intelligence for Business*. Pearson FT Press.
- Scaled Agile, Inc. (2024). SAFe 6.0 Framework.
- Scrum.org (2024). Nexus Guide.
- Singhal, K., Azizi, S., Tu, T., and et al. (2023). Publisher correction: Large language models encode clinical knowledge. *Nature*, 620:E19.
- Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. MIT Press.
- Talebirad, Y. and Nadiri, A. (2023). Multi-agent collaboration: Harnessing the power of intelligent llm agents.
- Tariverdi, A. (2024). Trust from ethical point of view: Exploring dynamics through multiagent-driven cognitive modeling.