

Engineering LLM-based Multi-Agent Systems: A Taxonomy of Emerging Frameworks

Davide Di Ruscio, *University of L'Aquila, Italy*

Phuong T. Nguyen, *University of L'Aquila, Italy*

Claudio Di Sipio, *Johannes Kepler University, Austria*

Riccardo Rubei, *Mälardalen University, Sweden*

Juri Di Rocco, *University of L'Aquila, Italy*

Abstract—LLM-based multi-agent systems (LMAS) are emerging as a promising paradigm, enabling specialized agents to collaborate and act autonomously across complex software engineering (SE) tasks. Yet, despite rapid progress, the field still lacks a clear conceptual foundation to guide researchers and practitioners in systematically designing, implementing, and evaluating such systems. This article introduces a taxonomy intended to provide practical guidance for integrating LMAS into software development workflows. Grounded in an analysis of prominent frameworks and scientific contributions for engineering LMAS, the taxonomy organizes the design space along five key dimensions offering a structured lens for understanding and engineering LMAS. Our analysis reveals substantial gaps in current frameworks, particularly regarding monitoring, performance evaluation, and quality assessment. These findings highlight pressing research challenges and identify concrete directions toward more reliable, transparent, and effective agentic systems for SE.

In recent years, the application of large language models (LLMs) has reshaped automated software engineering (SE) by supporting an increasingly large portion of the development lifecycle, from requirements engineering to code generation. Recent surveys provide comprehensive overviews of LLM architectures, from foundational Transformer-based models to systems such as GPT-4 and LLaMA, and structured taxonomies covering scalability, application domains, and ethical dimensions [15].

These advances are boosting developer productivity and enabling a new paradigm known as AIWare [3], [4]. Within this paradigm, development workflows are redefined by embedding AI agents as active collaborators capable of assisting developers in documentation tasks, code reviews, or automated testing [5]. A key evolution within AIWare is the emergence of multi-agent systems (MAS), conceptualized as a novel mechanism for combining the strengths of multiple LLMs [6]. MAS orchestrate, manage, and evaluate sets of AI agents, each designed for specialized tasks, to collaborate and compensate for the limitations of individual models. This approach underpins the AgentWare paradigm, where the synergy among multiple agents is expected to

enhance overall performance [7].

Despite the growing adoption of MAS in SE, significant gaps remain. The field lacks a comprehensive understanding of the foundational concepts and features that constitute an effective MAS. This makes it difficult for researchers and practitioners to design systems that are robust, scalable, and adaptable to the diverse challenges. Key questions, such as how to define agent roles, coordinate inter-agent communication [8], [9], or evaluate collective performance remain largely unanswered [10]. It is still unclear how MAS can be integrated into existing development environments without introducing unnecessary complexity or overhead. Without a clear framework or a set of engineering principles, the potential of MAS to transform software development under the AgentWare paradigm cannot be fully realized.

This paper contributes to the understanding of LMAS by proposing a taxonomy to capture both foundational and practical features for designing and developing MAS in the scope of the MOSAICO (Management, Orchestration and Supervision of AI-agent Communities for reliable AI in Software Engineering) EU project,¹ which focuses on supporting the reliable application of generative AI to SE tasks and building a Community of AIs (or AI-agent

community). We begin by analyzing existing systematic studies to derive a set of conceptual dimensions. Then, we examine prominent agentic frameworks to identify concrete mechanisms that map to these dimensions. The final taxonomy consists of 5 critical dimensions (*Core Architecture, Governance Rules, Agent Monitoring, Behavioral Models, and Reusability*), each supported by a set of subconcepts that collectively provide a structured lens for understanding and engineering LMAS. Our taxonomy is technology-independent and covers both existing frameworks and relevant scientific contributions, with the goal of providing guidance for researchers and practitioners in the systematic design, implementation, and evaluation of MAS for SE.

Paradigm Shift

The software development landscape has recently undergone 3 major paradigm shifts. *CodeWare* relied on fully manual programming, with developers writing and maintaining code directly. *NeuralWare* introduced AI-assisted development, but primarily through isolated models trained for specific tasks. Today, the emergence of *AgentWare* marks the next evolution: systems composed of multiple AI agents collaborating autonomously to address complex SE problems.

Latest News

- [2025/12] [DeepSpeed Core API updates: PyTorch-style backward and low-precision master states](#)
- [2025/11] [DeepSpeed ZeRO++ powers large-scale distillation training of LLMs for Recommendation Systems at LinkedIn](#)
- [2025/10] We hosted the [Ray x DeepSpeed Meetup](#) at Anyscale. We shared our most recent work on SuperOffload, ZenFlow, Muon Optimizer Support, Arctic Long Sequence Training and DeepCompile. Please find the meetup slides [here](#).
- [2025/10] [SuperOffload: Unleashing the Power of Large-Scale LLM Training on Superchips](#)
- [2025/10] [Study of ZenFlow and ZeRO offload performance with DeepSpeed CPU core binding](#)
- [2025/08] [ZenFlow: Stall-Free Offloading Engine for LLM Training](#)
- [2025/06] [Arctic Long Sequence Training \(ALST\) with DeepSpeed: Scalable And Efficient Training For Multi-Million Token Sequences](#)
- [2025/06] [DeepNVMe: Affordable I/O scaling for Deep Learning Applications](#)

► More news

Extreme Speed and Scale for DL Training

DeepSpeed enabled the world's most powerful language models (at the time of this writing) such as *MT-530B* and *BLOOM*. DeepSpeed offers a confluence of *system innovations*, that has made large scale DL training effective, and efficient, greatly improved ease of use, and redefined the DL training landscape in terms of scale that is possible. These innovations include ZeRO, ZeRO-infinity, 3D-Parallelism, Ulysses Sequence Parallelism, DeepSpeed-MoE, etc.

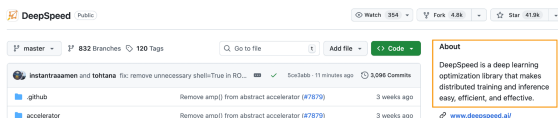


FIGURE 1: A well-documented repository with “About” description.

The README Summarization Problem

To illustrate the paradigm shift from single-model to multi-agent approaches, consider the task of automatically gen-

erating descriptions for GitHub repositories from their README files. Fig. 1 shows a typical scenario: while the DeepSpeed repository² provides a concise “About” description to immediately convey its purpose, many repositories leave this field blank [13], forcing visitors to read long README files to understand the project’s scope.

This task presents challenges that make single-model approaches struggle: README files exhibit *heterogeneous content structure*, mixing prose with code snippets and Markdown formatting, and *extreme length variability*, ranging from a few sentences to thousands of lines requiring selective extraction.

A Multi-Agent Solution

These challenges motivate a multi-agent decomposition. Nguyen et al. [7] demonstrated how specialized agents can address each challenge through collaboration. Four agents are grouped into 2 pipelines (Fig. 2) implemented using LangChain:³ the upper part corresponding to *Optimization Pipeline* handles prompt refinement, while the lower part corresponding to *Evaluation Pipeline* generates and assesses summaries.

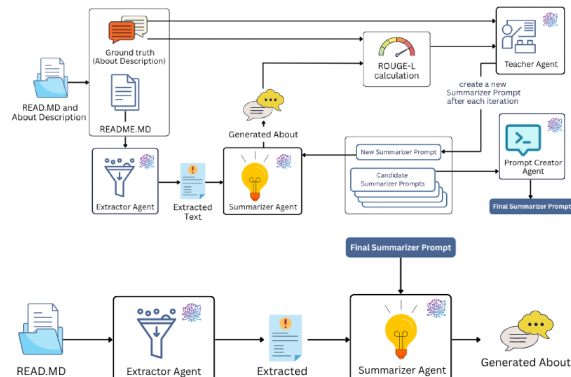


FIGURE 2: MAS pipelines for README summarization: agents collaborate through specialized roles and feedback loops.

Extractor Agent addresses content heterogeneity by filtering out non-descriptive sections (installation, contributing guidelines, licenses). Summarizer Agent generates candidate “About” descriptions, guided by prompts that evolve through interaction with Teacher Agent, which evaluates summaries using ROUGE metrics and proposes improvements. Finally, Prompt Creator Agent synthesizes successful prompt patterns into a reusable template, enabling generalization across diverse repositories.

²<https://github.com/microsoft/DeepSpeed>

³<https://www.langchain.com/>

However, this is only one possible multi-agent configuration. Alternative designs might use different roles, coordination patterns, or communication protocols. Such design choices introduce trade-offs in performance, scalability, and robustness, motivating the need for a systematic taxonomy to guide the construction of MAS for SE tasks.

A Practical Lens for Engineering MAS

To develop the taxonomy, we started from a focused corpus of primary studies and technical artifacts addressing LLM-based agentic and multi-agent systems for SE. Studies were selected based on explicit inclusion criteria: they describe complete agentic pipelines (rather than isolated techniques), make architectural and coordination choices explicit, and report concrete usage scenarios or empirical evaluations. This selection enabled the identification of recurring design concerns that are independent of specific tasks or implementations. We included English-language surveys on LLM-based MAS published in well-ranked venues (SCIMAGO/CORE) or as relevant preprints, from Jan. 2023 to Jan. 2025, excluding non-LLM MAS, non-systematic studies, workshops, and posters. Using Scopus, we combined MAS-related keywords (multi-agent system, LLM, large language models) with systematic-study keywords (survey, systematic review, literature review, meta-analysis), obtaining 47 initial papers and 18 after applying the criteria. To foster reproducibility, we provide a detailed replication package on Zenodo⁴ that includes the full search protocol, inclusion/exclusion criteria, intermediate filtering steps with exclusion rationale, and the iterative taxonomy derivation process from the initial 57 concepts to the final 21 sub-concepts.

We analyzed a representative set of 14 frameworks to operationalize these concerns. Frameworks covering different design paradigms, e.g., workflow-oriented, graph-based, and tool-augmented were examined by mapping their architectural structures, governance mechanisms, prompt and role specifications, monitoring facilities, and reuse strategies to higher-level abstractions. This synthesis produced a taxonomy organized along five dimensions, each refined into operational sub-concepts. For each sub-concept, Table 1 lists the papers (P) and MAS frameworks (MF) in which it appears, indicating its maturity and adoption.

Core Architecture

The majority of the analyzed studies agree on the core components of MAS, i.e., agents, tools, and memory. The

application domain (FC1.1) defines the high-level goal of the MAS and can be seen as a composition of agents' objectives (FC4.1), specified in the *Behavioral model*. Although frameworks do not enable direct specification of the application domain with dedicated capabilities, they indirectly support that by defining agents' objectives (FC4.1), external tools (FC1.5), and roles (FC4.4) to accomplish the task.

Agents (FC1.4) are the core elements of a MAS and can be homogeneous or heterogeneous. *Homogeneous vs. Heterogeneous (FC1.2)* captures whether agents share the same architecture. Only P3, P7, and P13 explicitly discuss this distinction, while all frameworks support heterogeneous agents, i.e., different providers and architectures such as GPT, Llama, or Claude. *Tools (FC1.5)* provide external capabilities to agents, and *Memory (FC1.3)* stores information about agent state and interactions with the environment.

Governance Rules

The governance rules are the core of any MAS. *Coordination Patterns (FC2.1)* defines how agents interact with each other and with the environment. This is well covered in the literature and supported by several frameworks. For instance, LlamaIndex⁵ allows developers to specify coordination rules that govern when agents invoke tools, stop, or hand off control to other agents. While this enables flexible and modular behaviors, it also increases dependence on the LLM's ability to follow instructions and maintain internal consistency.

Communication Mechanisms (FC2.2) specifies how agents exchange information, e.g., through centralized, decentralized, or shared message pools. These features are tightly coupled, since coordination requires appropriate communication support. *Agent-Environment Interface (FC2.3)* defines the operational context shaping agents' sensory inputs, action space, and interaction possibilities. Three types can be identified: *text-based* (natural language or structured formats such as JSON), *virtual sandbox* (2D/3D visual settings for gaming or robotics), and *physical* (real-world interaction with rich sensory data). All surveyed frameworks currently support only text-based environments, though they could in principle be extended to other settings.

Competency Mapping (FC2.4) defines which agents can employ which external tools. While not fully covered in the literature, frameworks implement this in practice, enabling modular agent design.

Agent Monitoring

The monitoring of agents is a crucial aspect of MAS systems. *Value-addition modeling (FC3.1)* represents a

⁴<https://zenodo.org/records/19919086>

⁵https://github.com/run-llama/llama_index

generic KPI-oriented assessment of agents, spanning from the accuracy metrics to advanced qualitative aspects, e.g., energy consumption, cost, and execution time. However, most of the considered frameworks neglect this aspect. The support for human-in-the-loop functionalities (FC3.2) is also limited. Interestingly, we report that none of the tools has a specific component for *Continual evolution* (FC3.3) of the agents, apart from MF14, even though this concept is mentioned in the literature. Our intuition is that those qualitative aspects are not yet fully implemented in the frameworks, triggering the need for further improvements.

Behavioral Models

The Behavioral model (FC4.1–FC4.5) is well covered by both literature and frameworks, as these features originate from single-agent systems. The agent's *Objective* (FC4.1) represents the minimal concept of any MAS, whereas *Perception* (FC4.2) is bounded by the operational context. *Action* (FC4.3) is also well-defined in the literature, providing a clear understanding of how agents interact with each other and how they communicate. A pivotal feature that impacts the agents' behavior is the role (FC4.4), which specifies all relevant information needed to perform the objective. *Message Content* (FC4.5) refers to the content of the messages exchanged between agents and the environment. This is not explicitly covered by the literature, but a crucial aspect of any MAS, since it defines how agents communicate.

Reusability

It is a crucial aspect of MAS frameworks, enabling developers to leverage existing components and functionalities. The analyzed research papers highlight the importance of reusability features, but there is a lack of systematic studies addressing this issue. Instead, most frameworks provide capabilities that can be used to enhance agent reusability.

Remote access (FC5.1) and *Agent comparison* (FC5.2) are covered by the majority of the frameworks. In contrast, *Benchmarking* (FC5.3) and *Discovery* (FC5.4) are still neglected by most of the frameworks.

Taxonomy in Practice

To demonstrate how the taxonomy guides concrete engineering decisions, we revisit the aforementioned README summarization example and map its architecture to our 5 dimensions.

Core Architecture (FC1). The system implements a heterogeneous design (FC1.2), deploying GPT-4o-mini for high-volume tasks (Extractor Agent, Summarizer Agent) and GPT-4o for reasoning-intensive roles (Teacher Agent, Prompt Creator Agent).

Memory (FC1.3) operates at 2 levels: Teacher Agent tracks iteration-level ROUGE scores, while Prompt Creator Agent aggregates successful patterns for cross-instance generalization. The specialized agents (FC1.4) address distinct challenges: content filtering, summary generation, evaluation, and meta-optimization.

Governance Rules (FC2). The coordination pattern (FC2.1) balances *flexibility vs. controllability*: Teacher-Summarizer feedback loops enable adaptive refinement, while ROUGE gates ensure reproducibility. Communication (FC2.2) uses structured JSON/YAML via LangChain's centralized orchestration. Competency mapping (FC2.4) explicitly assigns filtering to Extractor Agent, generation to Summarizer Agent, evaluation to Teacher Agent, and pattern synthesis to Prompt Creator Agent.

Agent Monitoring (FC3). Value-addition modeling (FC3.1) implements dual constraints: quality and cost. Teacher Agent provides explicit, interpretable feedback (FC3.2) through gap analysis and actionable prompt refinements.

TABLE 1: Elicited taxonomy from literature and MAS frameworks analysis.

Dim.	Description	Covered by the analyzed Papers (P) and MAS Frameworks (MF)
FC1	FC1.1 - Application domain	P1, P2, P3, P4, P5, P6, P7, P9, P11, P13, P14, P15, P18
	FC1.2 Type	P1, P7, P13, MF1, MF2, MF3, MF4, MF5, MF6, MF7, MF8, MF9, MF10, MF11, MF12, MF13, MF14
	FC1.3 Memory	P1, P2, P3, P4, P6, P7, P8, P9, P11, P12, P15, P17, P18, MF1, MF2, MF3, MF4, MF5, MF6, MF7, MF8, MF9, MF10, MF11, MF13, MF14
	FC1.4 Agent	P1, P2, P3, P4, P5, P6, P7, P8, P9, P11, P12, P13, P14, P15, P16, P17, P18, MF1, MF2, MF3, MF4, MF5, MF6, MF7, MF8, MF9, MF10, MF11, MF12, MF13, MF14
	FC1.5 Tool	P1, P2, P3, P4, P7, P8, P11, P15, P18, MF1, MF2, MF3, MF4, MF5, MF6, MF7, MF8, MF9, MF10, MF11, MF12, MF13, MF14
FC2	FC2.1 - Coordination Patterns	P1, P2, P4, P6, P7, P9, P11, P12, P13, P15, P14, P16, P17, P18, MF1, MF2, MF3, MF5, MF6, MF7, MF8, MF9, MF10, MF11, MF12, MF13
	FC2.2 - Communication Mechanism	P1, P2, P4, P6, P7, P9, P11, P12, P13, P14, P15, P16, P17, P18, MF1, MF2, MF3, MF5, MF6, MF7, MF8, MF9, MF10, MF11, MF12, MF13, MF14
	FC2.3 - Agents-Environment Interface	P2, P4, P7, P8, P11, P12, P14, P18, MF1, MF2, MF3, MF5, MF6, MF7, MF9, MF10, MF11, MF12, MF13
	FC2.4 - Competency mapping	P1, P2, P3, P4, P7, P8, P11, P15, P18, MF1, MF2, MF3, MF5, MF6, MF7, MF8, MF9, MF10, MF11, MF12, MF13, MF14
FC3	FC3.1 - Value-addition modeling	P1, P2, P4, P5, P6, P15, P18, MF2, MF6, MF7, MF13
	FC3.2 - Feedback	P1, P2, P4, P5, P6, P7, P11, P12, P18, MF2, MF3, MF4, MF5, MF6, MF8, MF9
	FC3.3 - Continual evolution	P2, P4, P6, P7, P8, P15, MF14
FC4	FC4.1 - Objective	P1, P2, P3, P4, P5, P6, P7, P8, P9, P11, P12, P13, P14, P15, P16, P17, P18, MF1, MF2, MF3, MF4, MF5, MF6, MF7, MF8, MF9, MF10, MF11, MF12, MF13, MF14
	FC4.2 - Perception	P1, P2, P3, P4, P7, P9, P11, P12, P15, P16, P18, MF1, MF3, MF4, MF5, MF6, MF7, MF8, MF9, MF10, MF11, MF12, MF13, MF14
	FC4.3 - Action	P1, P2, P3, P4, P5, P6, P7, P8, P9, P11, P12, P13, P14, P15, P16, P17, P18, MF1, MF2, MF3, MF4, MF5, MF6, MF7, MF8, MF9, MF10, MF11, MF12, MF13, MF14
	FC4.4 - Role	P1, P2, P3, P4, P5, P6, P7, P8, P9, P11, P12, P13, P14, P15, P16, P17, P18, MF1, MF2, MF3, MF4, MF5, MF6, MF7, MF8, MF9, MF10, MF11, MF12, MF13, MF14
	FC4.5 - Message Content	P2, P3, P6, P7, P12, P14, P15, MF1, MF2, MF3, MF4, MF5, MF6, MF7, MF8, MF9, MF10, MF11, MF12, MF13, MF14
FC5	FC5.1 - Remote access	MF1, MF2, MF3, MF6, MF7, MF8, MF9, MF10, MF11, MF12, MF13, MF14
	FC5.2 - Agent comparison	MF2, MF3, MF6, MF7, MF12, MF13
	FC5.3 - Benchmarking	MF2, MF3, MF5, MF11, MF13
	FC5.4 - Discovery	MF2, MF3, MF7

Behavioral Models (FC4). Each agent has clear objectives (FC4.1): minimize noise (*Teacher Agent*), maximize semantic fidelity (*Summarizer Agent*), minimize summary-reference gap (*Teacher Agent*), maximize pattern reusability (*Prompt Creator Agent*). Roles (FC4.4) are encoded as task-specific prompts with behavioral constraints and output schemas.

Reusability (FC5). Remote access (FC5.1) via OpenAI API enables cloud execution. Agent comparison (FC5.2) manifests as iteration-to-iteration ROUGE tracking. Benchmarking (FC5.3) uses curated datasets with ground-truth descriptions.

Beyond its analytical purpose, the taxonomy can act as a structured knowledge base for automated agent management. As an illustrative scenario, consider assembling a multi-agent code review pipeline. An orchestrator agent could query the taxonomy to discover candidates by *Role* (FC4.4), *Objective* (FC4.1), memory capabilities (FC1.3), and finally rank agents via *Agent Comparison* (FC5.2) and *KPI assessment* (FC3.1) metadata to select the most suitable combination. The taxonomy enables suggesting both agents and frameworks matching a task's *Coordination Patterns* (FC2.1), *Communication Mechanisms* (FC2.2), and governance needs such as human-in-the-loop validation (FC3.2). While a technical report is beyond this paper's scope, the taxonomy provides the conceptual foundation that makes automated discovery, comparison, and selection feasible, and constitutes one of the envisioned targets of the MOSAICO platform.⁶

Actionable Insights

Our analysis suggests several practical implications for the engineering of LLM-based multi-agent systems.

▷ MAS should be treated as engineered software systems, requiring explicit design decisions and systematic evaluation. Key choices such as agent heterogeneity, coordination strategies, and memory management are often implicit; the taxonomy makes them explicit, comparable, and reusable across applications.

▷ Monitoring and evaluation remain critical yet under-supported. Capabilities such as feedback integration and continual evolution are rarely implemented, making the robustness of agentic solutions difficult to justify and reproduce.

▷ Governance mechanisms introduce a trade-off between flexibility and controllability. Dynamic coordination increases expressiveness but amplifies dependence on LLM

behavior, calling for hybrid approaches that combine flexible orchestration with explicit supervision.

▷ The limited support for benchmarking, discovery, and agent comparison hinders scalability and reuse. Advancing AgentWare requires shared benchmarks, standardized evaluation pipelines, and richer agent metadata to move beyond isolated, task-specific implementations.

Conclusion and Future Work

The taxonomy aims at providing an engineering-oriented view of the emerging AgentWare paradigm. The analysis confirms that while architectural primitives such as agents, tools, and memory are now well established, advanced capabilities related to monitoring, evaluation, and evolution remain insufficiently supported. Within this context, the MOSAICO initiative serves as a concrete reference for how supervision, governance, and orchestration can be elevated to first-class engineering concerns. This work raises several open research questions critical to the maturation of LMAS:

▷ *How can LMAS be designed to adapt to evolving SE practices and technologies over time?*

▷ *How can monitoring be systematically integrated into agentic frameworks to enable reproducible and comparable evaluations?*

▷ *Which governance and coordination mechanisms best balance flexibility and controllability in multi-agent systems for SE tasks?*

▷ *How can benchmarking, discovery, and reuse of agents be standardized to support scalable and long-lived agentic ecosystems?*

Addressing these questions is essential to move from experimental demonstrations toward robust, transparent, and reusable multi-agent systems that can be reliably adopted in real-world software engineering workflows. Due to space limit, our taxonomy does not cover specific qualitative aspects such as security, cost management, and ethics, which are considered as future work.

Acknowledgments

This paper has been supported by the MOSAICO project that has received funding from EU under the Horizon RIA (Grant Agreement No. 101189664).

⁶<https://gitlab.eclipse.org/eclipse-research-labs/mosaico-project/mosaico-repository>

About the Authors



Davide Di Ruscio is a Full Professor of Computer Science at the University of L'Aquila. His research spans Software Engineering and Model-Driven Engineering, with contributions to domain-specific languages, model evolution, low-code development, and ML-based

recommender systems. For more information, please visit <https://www.disim.univaq.it/DavideDiRuscio>.



Phuong T. Nguyen is Associate Professor at the University of L'Aquila, Italy. He obtained a PhD in Computer Science from the University of Jena, Germany. His research interests include Recommender Systems, Machine Learning, and Mining Software Repositories. Phuong is

Editor-in-Chief of Software Quality Journal. Website: <https://www.disim.univaq.it/ThanhPhuong>.



Claudio Di Sipio is a Postdoctoral researcher at the Johannes Kepler University, Linz. His research interests include recommendation systems for SE, mining OSS repositories, MDE, and the application of ML/AI techniques for software

engineering. More info: <https://claudiodsi.github.io/>. Email: claudio.di_sipio@jku.at.



Riccardo Rubei is a Postdoctoral researcher at Mälardalen University, Sweden. He earned his Ph.D. in 2022 from the University of L'Aquila, Italy. His research interest is related to SE, recommender systems, and several aspects of MDE. Furthermore, he is active in the field of

sustainability and green SE. Email: riccardo.rubei@mdu.se.



Juri Di Rocco is a tenure-track assistant professor at the University of L'Aquila. His research

interests encompass all aspects of software language engineering. His primary focus lies in MDE, specifically in domain-specific modeling languages, recommender systems for MDE, modeling repositories, and mining techniques.

REFERENCES

1. Y. Hou et al., "Large language models for software engineering: A systematic literature review," *J. Syst. Softw.*, vol. 200, p. 111637, 2023.
2. F. Thung et al., "A survey on large language models for code generation and code intelligence," *IEEE Trans. Softw. Eng.*, vol. 49, no. 12, pp. 5100–5117, 2023.
3. D. Monett and C. Lemke, "AI-ware: Bridging AI and Software Engineering for responsible and sustainable intelligent artefacts," in *Proc. Workshop AI-aware Softw. Eng.*, 2021.
4. N. Nijkamp et al., "Green AI in Action: Strategic Model Selection for Ensembles in Production," in *Proc. 1st ACM Int. Conf. AI-Powered Softw.*, 2024, pp. 50–58.
5. S. Ahmad et al., "AI Agents in Software Development: A Survey on LLM-Powered Workflows," in *Proc. ACM/IEEE 46th Int. Conf. Softw. Eng.: Softw. Eng. Practice.*, 2024, pp. 234–245.
6. X. Wang et al., "Multi-Agent Systems with Large Language Models: A Survey," *ACM Comput. Surv.*, vol. 57, no. 1, Article 15, pp. 1–39, 2025.
7. D. S. H. Nguyen et al., "Teamwork makes the dream work: LLMs-Based Agents for GitHub README.MD Summarization, Proceedings of the 33rd ACM International Conference on the Foundations of Software Engineering, 2025, pp. 621-625
8. L. Sun et al., "Multi-Agent Coordination across Diverse Applications: A Survey," arXiv preprint arXiv:2502.14743, 2025.
9. B. Yan et al., "Beyond Self-Talk: A Communication-Centric Survey of LLM-Based Multi-Agent Systems," arXiv preprint arXiv:2502.14321, 2025.
10. L. Wang et al., "A Comprehensive Survey on Multi-Agent Systems with Large Language Models," arXiv preprint arXiv:2402.05718, 2024.
11. A. Hepworth et al., "Onto4MAT: Comprehensive Ontological Framework for Multi-Agent Technologies," *IEEE Intell. Syst.*, vol. 37, no. 4, pp. 45–58, 2022.
12. A. Handler et al., "A taxonomy of multi-agent coordination mechanisms for large language models," in *Proc. Int. Conf. Autonomous Agents Multiagent Syst.*, 2023, pp. 789–798.
13. T. T. H. Doan, P. T. Nguyen, J. Di Rocco, and D. Di Ruscio, "Too long; didn't read: Automatic summariza-

- tion of GitHub README.MD with Transformers,” in *Proc. 27th Int. Conf. Evaluation Assessment Softw. Eng. (EASE '23)*, ACM, New York, NY, USA, pp. 267–272, Jun. 2023, doi: 10.1145/3593434.3593448.
14. P. Pathak and P. S. Rana, “Comparative Analysis of Pretrained Models for Text Classification, Generation and Summarization: A Detailed Analysis,” in *Pattern Recognition*, A. Antonacopoulos, S. Chaudhuri, R. Chellappa, C.-L. Liu, S. Bhattacharya, and U. Pal, Eds. Cham: Springer Nature Switzerland, 2025, pp. 151–166.
 15. I. D. Mienye et al., “Large language models: an overview of foundational architectures, recent trends, and a new taxonomy,” *Discov. Appl. Sci.*, vol. 7, p. 1027, 2025.