# Literature reviews

Roman Reimche

RPTU Kaiserslautern-Landau, Faculty of Business and Economics

**Note:** *This report contains a project documentation and reflection on the portfolio task submitted for the lecture Engineering with Generative AI in WiSe 2024-25. This report is an original work and will be scrutinised for plagiarism and potential LLM use.*

# 1 Reviews on LLM-Based Multi-Language Systems for End-to-End Software Engineering

### 1.0.1 Introduction

Recently, advancements in the field of natural language language processing using large language models has experienced substantial advancements, allowing application not possible before, also in the area of text to text generation – a model receives textual input and produces output that, on multiple benchmarks is close to human performance for the same tasks. This has allowed multi-agent systems to use LLMs as key component for agent design, allowing agentic systems to emulate humanlike information processing, communication and decision taking. One of the known applications of LLMs is software code generation and multiple researchers have recently studied the topic of emulation of software development process using LLM-based multi-agent systems. Such engineering process transcends code generation task by involving complex activities performed by multiple roles of interconnecting actors: requirements engineering, project architecture, project planning and management, generation of multiple heterogeneous software artifacts and quality assurance. Multiple software project paradigms are used in industry, waterfall and agile beign ones of the most common. Moreover, such process in industry can be adapted per case, omitting some and adding other activities, roles and communication rules. In this review we look at 6 recent papers on the topic of LLM-based multi-agent systems for end-to-end software development, that is, full cycle software development with requirements as input and executable system as output, optionally including human interactions in between. We also review critically these contributions in the field and devise current state-of-the-art and future study possibilities.

In Section 1.0.2, we write on our literature selection process. After that, in Section 1.0.3, we summarize the works and in Section 1.0.4 we draw conclusions and present possible future research topics.

### 1.0.2 Literature selection

Since our multi-agent system is not designed to select papers, but only review them, comparing this review with the one generated by the system should exclude comparing literature selection. Besides, the selection process is already described in the project report, we omit it here.

### 1.0.3 Recent works

MetaGPT'24 [1], ChatDev'24 [2], and CodePori [3] implement different variations of the waterfall paradigm of software engineering process. So, ChatDev'24 splits it's straitforward pipeline into design, code and testing phases where in every phase several different agent roles communicate with each other to produce output for the next phase. The authors also introduce a mechanism of communicative dehallucination to reduce the amount hallucination-induced code deffects – agents in every phase discuss the task to make it more detailed. The system is evaluated using the Software Requirement Description Dataset (SRDD) [**?**] and compared with MetaGPT'23[1] and one other framework, showing that ChatDev'24 surpasses the second best competitor (MetaGPT'23) substantially.

MetaGPT'24 [1] employs 6 phases: planning, requirements analysis, architectural design, system design, coding and testing . The authors also introduce an quality enhancement mechanism included in coding role. This mechanism allows the agent to take prior execution and message history into account to enhance produced software artifacts. The results are evaluated using HumanEval [**?**] and MBPP [4] and achieves 85,9% and 87,7% in pass@1. Besides that, MetaGPT'24 ist compared with several other frameworks including ChatDev'2023 [**?**] in terms of capabilities, like requirements and design generation, code and API generation and others, showing that MetaGPT'24 has a more sophisticated process than other frameworks.

CodePori [3] presents a software development process that, to some extent, can be seen as unifying some of the ideas from the previous two. So, this framework includes planning phase as MetaGPT'24 and also in-phase communication between phase-related agents (e.g. 2 developers discussing their work or 2 quality assurance engineers) similar to ChatDev'24. Otherwise, the information goes through the stages of planning, code generation and two stages of quality assurance, one of which involves the outlines inter-agent communication and the other employs only one agent trying to fix flaws missed in the previous stage. This work compares it's results with some other frameworks using HumanEval [**?**] benchmark and pass@1 metric. Among the other frameworks, both ChatDev (almost 70% on pass@1) and MetaGPT (slightly above 80%) are present and the comparison shows that CodePori performed better than both of them hitting almost 90%. However, authors don't specify which versions of these systems were involved in comparison. Furthermore, authors used their own handcrafted set of tasks and manually assessed the quality of software artifacts produced by their framework concluding that 85% of the tasks were finished with acceptable quality (programs were executable after generation or after minor manual adjustments in generated code).

conclusion:

- though the frameworks allow boilerplating, if they can build an almost production ready system using complex real world requirements is still unclear and cen be studied in future

- cost is almostly neglectable compare to humans but unclear how much manual work is required for fixing flaws left by agents

---

[1]The authors use MetaGPT version from 2023 [**?**], whereas we review MetaGPT'24 [**?**]

### 1.0.4 Critical review and conclusion

# References

[1] Sirui Hong, Mingchen Zhuge, Jiaqi Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. Metagpt: Meta programming for a multi-agent collaborative framework, 2024.

[2] Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. Chatdev: Communicative agents for software development, 2024.

[3] Zeeshan Rasheed, Malik Abdul Sami, Kai-Kristian Kemell, Muhammad Waseem, Mika Saari, Kari Systä, and Pekka Abrahamsson. Codepori: Large-scale system for autonomous software development using multi-agent technology, 2024.

[4] Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. A discourse-aware attention model for abstractive summarization of long documents. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, 2018.