Autonomous Agents in Software Development: A Vision Paper

Zeeshan Rasheed[1], Muhammad Waseem[2](✉), Malik Abdul Sami[3], Kai-Kristian Kemell[4], Aakash Ahmad[5], Anh Nguyen Duc[1,2,3,4,5], Kari Systä[1,2,3,4,5], and Pekka Abrahamsson[1,2,3,4,5]

[1] Tampere University, Tampere, Finland
{zeeshan.rasheed,kari.systa,pekka.abrahamsson}@tuni.fi,
Anh.Nguyen.duc@usn.no

[2] University of Jyväskylä, Jyväskylä, Finland
muhammad.m.waseem@jyu.fi

[3] University of Helsinki, Helsinki, Finland

[4] Lancaster University Leipzig, Leipzig, Germany
kai-kristian.kemell@helsinki.fi

[5] University of South Eastern Norway, Notodden, Norway

Abstract. Large Language Models (LLM) are reshaping the field of Software Engineering (SE). They enable innovative methods for executing many SE tasks, including automation of entire process of Software Development Life Cycle (SDLC). However, only a limited number of existing works have thoroughly explored the potential of LLM based AI agents to automate the entire lifecycle in SE. In this paper, we demonstrate the success of our initial efforts in automating the entire lifecycle autonomously based on given software specification as input, which has shown remarkable efficiency and significantly reduced development time. Our preliminary results suggest that the careful implementation of AI agents can enhance the development lifecycle. We aim to streamline the SDLC by integrating all phases into an AI-driven chat interface, enhancing efficiency and transparency. Furthermore, we seek to enhance collaboration, creating an environment where stakeholders from various backgrounds can contribute, review, and refine ideas and requirements in real-time. This forward-looking direction guarantees to redefine the paradigms of SE and also make software creation more inclusive, collaborative, and efficient.

Keywords: OpenAI · AutoGPT · Artificial Intelligence · Natural Language Processing · Generative AI · Software Engineering · Large Language Model

M. Waseem, K.-K. Kemell, A. Ahmad, A. N. Duc, K. Systä and P. Abrahamsson—Contributing authors.

L. Marchesi et al. (Eds.): XP 2024 Workshops, LNBIP 524, pp. 15–23, 2025.
https://doi.org/10.1007/978-3-031-72781-8_2

1 Introduction

AI has reshaped our interaction with machines and impacted many industries. Its most promising use is in Natural Language Processing (NLP), which helps computers understand and interact with human language [1]. Recent advancements in NLP have led to the development of Large Language Models (LLMs) such as the GPT series [2,3]. The interaction between LLMs and the domain of Software Engineering (SE) has led to significant advancements and intriguing possibilities [4]. LLMs have shown their potential to streamline multiple facets of the Software Development Life Cycle (SDLC) [5]. These advanced language models have emerged as a formidable force, utilizing the power of AI to autonomously generate code, offer insights, and assist developers across various stages of the SDLC. However, challenges such as ensuring code correctness, maintenance, and bridging the gap between NLP and programming semantics remain critical considerations in this symbiotic relationship.

In this paper, we demonstrate the success of our initial work in automating the entire SDLC by utilizing AI agents based on LLMs. We utilized 12 AI agents collaboratively functioning to autonomously execute all stages of the SDLC, including project planning, requirements engineering, system design, development, testing, deployment, among others. To assess the capabilities of our proposed model, we implemented software specification of varying sizes as inputs. Initial results suggest that our model reduces the development time, allowing for project completion within five minutes, and also enhances accuracy by delivering precise outcomes. We aim to redefine the frameworks of SE, making software creation a more inclusive, collaborative, and efficient process. With this goal in mind, we plan to extend this work to revolutionize software development through AI-driven chat interfaces, transforming the entire creation lifecycle by integrating all phases of software development into an AI-driven chat interface. Our proposed model will also facilitate a collaborative environment where multiple stakeholders can contribute, review, and refine ideas and requirements in real-time. The AI chat-based environment will include a feedback loop where the system learns from each interaction, refining its support and providing code recommendations over time. Our contribution can be summarized as follow:

- Demonstrate successful automation of the whole process of SDLC with LLM-based AI agents, enhancing efficiency and reducing time.
- Our future goal is to proposed an AI-driven chat interface integrating all development phases to promote real-time stakeholder collaboration and continuous AI learning for improved code recommendations.
- We also aimed to redefine the frameworks of SE to make the process of software creation more inclusive, collaborative, and efficient.

2 Background

2.1 Generative AI

Generative AI refers to a category of AI models and algorithms that are designed to generate new content that is often similar to content created by humans [6].

This type of AI has experienced notable progress in recent times [7]. Nowadays, generative AI has been utilized in various fields, such as NLP, computer vision, and image and video generation [8]. In NLP, generative AI techniques are commonly used for various tasks, including text generation, machine translation, dialog systems, and code generation. LLM is a specific type of generative AI model that excels at generating human-like text due to its architecture, pre-training, and fine-tuning processes [2]. The foundation of the LLM can be traced back to the introduction of the transformer architecture proposed by Vaswani *et al.*. [9]. This architectural innovation transformed the field of NLP by introducing the self-attention mechanism, enabling the model to capture contextual connections among words, irrespective of their position within a sequence. In 2018, OpenAI introduced the GPT-1 model to demonstrate the potential of LLMs for text generation tasks [2]. Several researchers and OpenAI have made significant contributions to improving the performance of GPT models by using a variety of techniques and approaches [2,3].

2.2 Large Language Models in SE

LLMs have shown promise in various SE applications [10,11]. LLMs generation capabilities offer valuable assistance and enhancements to SE processes [12–15]. By utilizing the remarkable natural language generation capabilities of GPT models, various SE tasks can now be automated and streamlined, including code generation, error detection, documentation creation, and beyond. Through GPT-powered code completion and generation, developers can swiftly produce high-quality code and even entire programs, significantly expediting the SDLC [16–19]. Several studies (e.g., [20–23]) has explored LLMs in SE. However, much of this research has been limited to case studies, with a focus on user perceptions in coding and writing. To fill this gap, our goal is to leverage multiple LLM based AI agents for autonomy generate the whole process of SDLC based on developer-provided software specification.

3 Research Methodology

In this study, we present an innovative approach to automating the entire SDLC by integrating multiple AI agents inspired by LLMs. This section outlines the methodology employed to design and implement our model. We have formulated the research question (RQ):

RQ. *How do multi-AI agents collaborate with each other to automate the entire lifecycle of software development?*

The main objective of **RQ** is to determine how multiple AI agents work together to automate the entire lifecycle of software development. **RQ** primarily aims to determine the dynamics of how these agents interact, coordinate, and function collectively to automate tasks from requirement engineering to deployment.

3.1 AI Agent Approach for Automating the SDLC (RQ)

As shown in Fig. 1, we utilized 12 LLM-based AI agents that collaborate to achieve the final result. Each agent in the system is a specialized instance of an LLM, trained to handle different aspects of SDLC. Each agent specializing in a distinct aspect of the software development process, collectively contribute to improving efficiency and enhancing the SDLC. Below, we discuss in detail that how AI agents collaborate to autonomously carry out the SDLC.

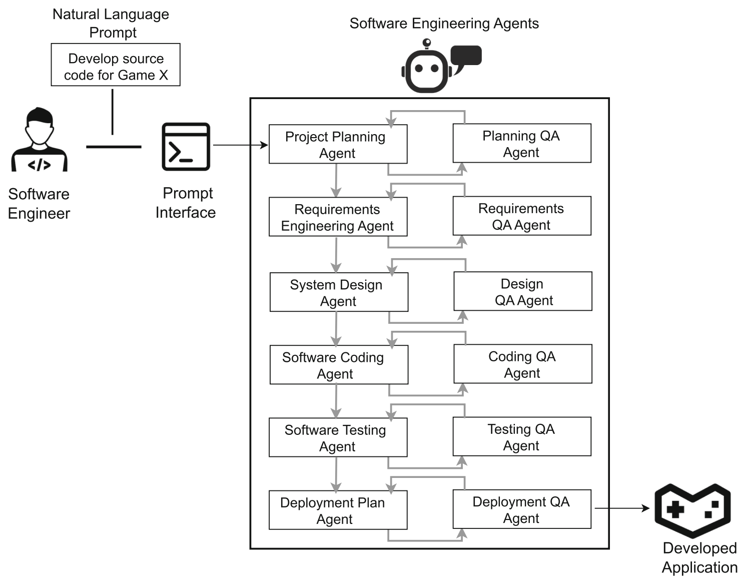

Fig. 1. A schematic representation of an automated software development process using autonomous agents

Agent-01 is crucial for project planning, starting with gathering software specifications and defining the project's scope, goals, and execution plan. Its main role includes outlining steps to achieve these objectives. Agent-2 complements this by evaluating the quality and feasibility of Agent-01's plans against the organization's goals, ensuring the project's plan is comprehensive, feasible, and aligned with overarching objectives, serving as a critical phase in the project lifecycle. Agent 3 moves to requirement engineering as the next phase, showing skill in identifying and understanding high-level requirements. This phase guarantees that all future stages of development are solidly based on a detailed understanding of the project's goals and constraints. Through this process, Agent 3 establishes a clear plan, which acts as the foundation for all upcoming project development efforts, closely matching the project's specified goals and the limits within which it must work. Collaborating with Agent-3, Agent-4's primary role is to assess the quality and precision of the project requirements. Early results show significant improvements in the quality and consistency of requirements documentation, highlighting Agent-4's essential contribution to defining clear requirements. This leads to smoother development phases and enhances the project's overall success and efficiency. Agent-5 is pivotal in converting high-level requirements into detailed system designs, accelerating the design phase and ensuring compliance with project specifications, laying a foundation for development with clear technical directions. Agent-6 enhances this by analyzing design quality and identifying system architecture flaws early, boosting efficiency by saving time and resources. Agent-07 streamlines software testing and defect detection through automation, facilitating continuous testing, speeding up development, and improving quality with less manual effort. Working in tandem, Agent-8 assesses the effectiveness of these automated tests, leading to better bug detection and resolution, underlining the importance of comprehensive testing in achieving high-quality software and efficient resource utilization. Agent-9 plays a important role in translating high-level design specifications into executable code. Early analysis of performance metrics suggests a significant reduction in development time. This efficiency accelerates the software development process and also ensures that the final product meets the expected performance and reliability criteria, showcasing the agent's pivotal role in streamlining software creation. Agent-10 is tasked with code quality analysis, focusing on ensuring that the generated code follows established coding standards and best practices. Initial findings highlight its success in promoting the development of clean, maintainable code. This reinforces the agent's role in enhancing software maintainability and reliability through rigorous quality assessments. Agent-11 is instrumental in managing the deployment phase of software projects, focusing on the systematic coordination and execution of steps essential for launching the software in a production environment or to its end-users. This phase is crucial for the seamless transition of software from development to actual use, ensuring that all components are correctly configured, integrated, and ready for operation in their intended settings. Agent-12's responsibility is to carefully review, assess, and refine the deployment plan formulated by Agent-11, aiming to guarantee a seamless and robust deployment process. This involves ensuring that the plan follows high standards of quality, anticipating potential challenges, and incorporating measures for a smooth transition to the production environment or end-user delivery.

4 Preliminary Empirical Results

In this section, we present preliminary results from our developed novel LLM based model. To evaluate capability of our model, we prompted 10 projects and as the outcome of the autonomous development process, the agents produced a requirements engineering specification, a software design plan, commented software code, a test and deployment plans. To offer a clear understanding of the model's outcome, we have provided Table 1 to show the results generated by our proposed model. However, below we only explain the outcomes of three projects due to page limitations.

Initially, the agents were prompted to "develop a snake game with GUI". As output the agents perform the whole SDLC process. The amount of documentation including review rounds is 6216 words. This is about 13 pages of single-spaced text. Requirements specification agent produced 7 Functional Requirements (FR), 7 Non-Functional Requirements (NFR) and 4 constraints. 11 were fully met, 2 partially, 4 not verified and 4 were not met. There was no restart-option, the code was not commented according to the standard PEP 257 and object-oriented programming principles were not implemented. The code did not include test-cases. The actual generated software code was 115 Lines of Code (LOC) and the whole project took less than 4 min to complete. Game required human debugging to make it run.

The second experiment was "Build a chat based application". The amount of documentation including review rounds is 7685 words. Requirements specification agent produced 9 FR and 9 NFR and 0 constraints. 14 were fully met and 4 were not met. The actual generated software code was 175 lines and the whole project took less than 7 min to complete.

Preliminary results highlight our multi-agent framework's potential to automate and optimize the software development process. Key focuses include scaling autonomous capabilities, defining limits, and extending this work to integrate all phases into an AI-driven chat interface for improved efficiency and transparency. Additionally, we aim to enhance collaboration by enabling stakeholders from diverse backgrounds to contribute and refine ideas in real time.

5 Future Work

Our future work aims to innovate the software development through AI-driven chat interfaces. This initiative aims to transform the SDLC, making it more inclusive, efficient, and collaborative through the innovative application. Our future goal is to utilize advanced AI functionalities to support deeper multi-user collaboration, enabling the AI to understand and adapt to the context of discussions. The result of this effort will be continuous integration of these AI capabilities with project management tools, transforming discussions into actionable items without manual intervention. For instance, when a team discusses a new feature, the AI would automatically prioritize this in the project's backlog, generate relevant user stories, and even suggest test cases based on the discussion's content. Moreover, to ensure that the AI remains effective over time and adapts to the evolving landscape of software development, we plan to incorporate a continuous learning mechanism. This system will refine the AI's recommendations and decision-making processes based on feedback and enable it to learn from the collective intelligence of its user base.

Accessibility will also be a key focus, with efforts aimed at making the collaborative workspace inclusive for global teams through real-time transcription and translation capabilities. This will break down language barriers and faster a more diverse and inclusive environment for software development. Finally, the implementation phase and iterative refinement of the AI system based on user

Table 1. Result produced by LLM based AI agent model

S.No	Input	Words	FR	NFR	Const.	LOC	Mins	Output
01	Develop a snake game with GUI	6216	7	7	4	115	4	Snake game with restart button
02	Build a chat based application	7685	9	9	0	175	7	Messaging chat based app
03	Create a tic-tac-toe game	5366	9	6	3	94	4	Tic tac toe game with GUI
04	Build a real-time weather app	6189	8	6	2	119	5	Real-time weater forecasting app
05	Create a fitness tracker app	4890	6	5	2	107	4	Real-time tracker app
06	Develop a virtual reality tour app	7112	7	9	5	189	9	Developed virtual app
07	Create a blog platform with user authentication	6788	7	9	3	149	6	Blog platform
08	Develop a personal finance tracker app	5312	6	3	2	134	5	Finanace app
09	Develop a short e-commerce website	5982	8	9	4	151	8	Small E-commerce Website
10	Create a recipe finder app	4112	5	4	2	133	4	Recipe search app

feedback and performance metrics will be critical. This approach will ensure that the system meets the initial goals and evolves in response to real-world use and feedback, Therefore Continuously enhancing the efficiency and innovation within software development processes.

6 Conclusions

This paper proposes to explore how LLM based AI agents can autonomously perform various SE tasks. Through initial experiments, we have shown some interesting results which may have significant consequences. We demonstrate that our method significantly reduces development time and advances code generation methodologies, reinforcing the potential of AI-driven practices in the SE domain. With this goal in mind, we aim to extend our work to study how far we can scale these methods and where do really need human developers to be involved in. Experimenting using our proposed framework, we can gain many insights on how AI technology can fundamentally change how software is developed. The various SE roles and activities are designed based on human nature, capabilities and limitations in knowledge, skills and communication. AI agents may not have these limits. Then a follow up question to ask is, do we still need the same activities and life cycles as we did in the past 50 to 60 years of software engineering?

Acknowledgment. We express our sincere gratitude to Business Finland for their generous support and funding of our project. Their commitment to fostering innovation and supporting research initiatives has been instrumental in the success of our work.

References

1. Chowdhary, K., Chowdhary, K.: Natural language processing. Fundam. Artif. Intell. 603–649 (2020)
2. Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al.: Improving language understanding by generative pre-training (2018)
3. Radford, A., et al.: Language models are unsupervised multitask learners. OpenAI blog **1**(8), 9 (2019)
4. Rasheed, Z., Waseem, M., Systä, K., Abrahamsson, P.: Large language model evaluation via multi AI agents: preliminary results. In: ICLR 2024 Workshop on Large Language Model (LLM) Agents (2024)
5. Khan, J.Y., Uddin, G.: Automatic code documentation generation using GPT-3. In: Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering, pp. 1–6 (2022)
6. Baidoo-Anu, D., Owusu Ansah, L.: Education in the era of generative artificial intelligence (AI): understanding the potential benefits of chatGPT in promoting teaching and learning. Available at SSRN 4337484 (2023)
7. Cao, Y., et al.: A comprehensive survey of AI-generated content (AIGC): a history of generative AI from GAN to chatGPT. arXiv preprint arXiv:2303.04226 (2023)
8. Hacker, P., Engel, A., Mauer, M.: Regulating chatGPT and other large generative AI models. In: Proceedings of the 2023 ACM Conference on Fairness, Accountability, and Transparency, pp. 1112–1123 (2023)
9. Vaswani, A., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems, vol. 30 (2017)
10. Feng, Y., Vanam, S., Cherukupally, M., Zheng, W., Qiu, M., Chen, H.: Investigating code generation performance of chat-GPT with crowdsourcing social data. In: Proceedings of the 47th IEEE Computer Software and Applications Conference, pp. 1–10 (2023)
11. Waseem, M., Das, T., Ahmad, A., Fehmideh, M., Liang, P., Mikkonen, T.: Using chatGPT throughout the software development life cycle by novice developers. arXiv preprint arXiv:2310.13648 (2023)
12. Thiergart, J., Huber, S., Übellacker, T.: Understanding emails and drafting responses–an approach using GPT-3. arXiv preprint arXiv:2102.03062 (2021)
13. Sami, A.M., et al.: System for systematic literature review using multiple AI agents: concept and an empirical evaluation. arXiv preprint arXiv:2403.08399 (2024)
14. Rasheed, Z., et al.: Can large language models serve as data analysts? A multi-agent assisted approach for qualitative data analysis. arXiv preprint arXiv:2402.01386 (2024)

15. Waseem, M., et al.: Artificial intelligence procurement assistant: enhancing bid evaluation. In: International Conference on Software Business, pp. 108–114. Springer, Cham (2023)
16. Dong, Y., Jiang, X., Jin, Z., Li, G.: Self-collaboration code generation via chatGPT. arXiv preprint arXiv:2304.07590 (2023)
17. Rasheed, Z., et al.: Autonomous agents in software development: a vision paper. arXiv preprint arXiv:2311.18440 (2023)
18. Waseem, M., Ahmad, A., Liang, P., Fehmideh, M., Abrahamsson, P., Mikkonen, T.: Conducting systematic literature reviews with chatGPT (2023)
19. Rasheed, Z., Waseem, M., Saari, M., Systä, K., Abrahamsson, P.: CodePori: large scale model for autonomous software development by using multi-agents. arXiv preprint arXiv:2402.01411 (2024)
20. Ahmad, A., Waseem, M., Liang, P., Fahmideh, M., Aktar, M.S., Mikkonen, T.: Towards human-bot collaborative software architecting with chatGPT. In: Proceedings of the 27th International Conference on Evaluation and Assessment in Software Engineering, pp. 279–285 (2023)
21. Barke, S., James, M.B., Polikarpova, N.: Grounded copilot: How programmers interact with code-generating models. In: Proceedings of the ACM on Programming Languages, vol. 7, no. OOPSLA1, pp. 85–111 (2023)
22. Vaithilingam, P., Zhang, T., Glassman, E.L.: Expectation vs. experience: evaluating the usability of code generation tools powered by large language models. In: CHI Conference on Human Factors in Computing Systems Extended Abstracts, pp. 1–7 (2022)
23. Ma, W., et al.: The scope of chatGPT in software engineering: a thorough investigation. arXiv preprint arXiv:2305.12138 (2023)