Tiako P.F. (ed) *Intelligent and Sustainable Solutions*. Chronicle of Computing. OkIP. CAIF25#8
© 2025 Oklahoma International Publishing https://doi.org/ 10.55432/978-1-6692-0011-6_6

Leveraging AI to Enhance Human-Driven Software Development: A Comparative Study Across Diverse Applications

Aziz Fellah

Northwest Missouri State University, School of Computer Science and Information Systems, Maryville MO 64468, USA
Email: afellah@nwmnissouri.edu

ABSTRACT

This study explores how Generative AI, including AI-Powered tools like ChatGPT, can enhance the Software Development Life Cycle (SDLC). In a software engineering course, students worked in teams on 12 use case studies spanning web development, mobile apps, and game development. These use case studies covered domains such as education, sports, healthcare, and entertainment. Teams adopted dual roles as stakeholders and developers. Each team first defined a use case study and outlined project requirements. Use case studies were then randomly assigned, and teams worked as developers, specifying requirements and architectural designs. Scrum-style meetings facilitated collaboration. The paper compared developer-created and AI-generated user stories, functional and non-functional requirements, and architectural designs, including UML diagrams. Results showed ChatGPT excelled in structured web and app development domains but struggled significantly in game development and faced considerable difficulty in generating UML diagrams across all applications. This research highlights the strengths and limitations of Generative AI in enhancing software development processes.

Keywords: Generative AI, ChatGPT, Software Requirements, Software Development Life Cycle, AI-Powered Tools

I. INTRODUCTION

This research explores how Generative AI tools, like ChatGPT, can support software developers in key phases of the Software Development Life Cycle (SDLC) (de Oliveira Santos, Figueiredo, et al., 2024; Heyn, Knauss, et al., 2021; Marar, 2024). Defining and refining requirements is a fundamental part of the SDLC, ensuring alignment with stakeholder expectations and guiding successful use case study outcomes (Ahmad, Bano, et al., 2021; de Oliveira Santos, Figueiredo, et al., 2024; Yoshioka, Husen, et al., 2021). Software developers strive to capture diverse stakeholder perspectives accurately, as any misalignment can lead to misunderstandings, rework, or even project failure. Well-defined requirements serve as a road map across the various phases of the SDLC, fostering collaboration and consensus while reflecting stakeholder objectives. Reaching consensus on critical decisions, such as those related to architecture and design, can be particularly challenging due to the varied experiences, perspectives, and skill sets of software developers (Bhandari et al., 2023; Todorov, 2022). These differences often lead to conflicting viewpoints, especially in large, complex projects with multiple stakeholders. This research investigates how tools like ChatGPT, powered by Generative AI, can assist developers in eliciting, analyzing, and refining both functional and non-functional requirements (Bhandari et al., 2023; Todorov, 2022). By enhancing collaboration, improving consensus, and addressing uncertainties, ChatGPT enables developers to make informed decisions and identify patterns within complex requirements (Lorenzoni et al., 2021; Terragni et al., 2024). By complementing traditional methods, ChatGPT offers new insights, enhances efficiency, and clarifies requirement management under diverse scenarios within software projects (Barenkamp et al., 2020; Lorenzoni et al., 2021; Smith et al., 2024).

The study progressed through a series of phases designed to mirror real-world software development. It engaged teams of students to work on 12 different use case studies in a software engineering course. Each team was randomly tasked with one use case study in one of three domains: web development, mobile app development, and game development. Domains included education, healthcare, e-commerce, entertainment, and sports. In Phase 1, teams brainstormed and proposed a use case study based on their interests. In Phase 2, projects were reassigned, and teams acted as developers for other teams' proposals, conducting stakeholder-developer meetings to clarify and validate functional and non-functional requirements. Phases 3 and 4 involved creating user stories and developing architectural designs—first by developers and then using AI tools, respectively. Students were unaware of the AI task until after submitting their own user stories, which encouraged deeper reflection during comparisons. In phase 5, teams analyzed AI-generated and developer-generated user stories, identifying similarities, differences, and gaps. In Phase 6, UML component and class diagrams were created, combining the best of AI-generated textual descriptions with the team's own visual diagrams. Finally, in Phase 7, teams presented their findings in scrum-style presentations, compared their work with AI-generated stories, and participated in a Q&A session to promote discussion. The next phases, implementing and coding, are currently underway this academic term, with the same teams continuing their work.

The remainder of this paper is organized as follows: Section II states the objectives of this research, which are to explore the role of AI-powered tools, such as ChatGPT, in comparison to human developers throughout the Software Development Life Cycle (SDLC). Section III outlines our proposed methodology, detailing the multi-phase approach, the participants, and the overall setup for the subsequent sections. Section VI introduces three key projects of the 12 use case studies, covering game design, web services, and mobile app development. Each team categorized their user stories into components, classes, and refined them further into tasks, serving as a road map for facilitating a structured comparison between AI-generated and team-developed outcomes for coding and implementation Section V presents a comparative analysis of the software specifications and design generated by both developers and AI across various domains. In Section VI, we discuss how AI tools can efficiently enhance SDLC research. Several AI-powered tools are highlighted, including *GitHub Copilot, Jira, Slack, Otter.ai, and Tabnine*. Finally, Section VII concludes the study, offering insights and posing key questions for future exploration. In the remainder of this paper, the term ”use case study” is used to describe the process or analysis, while ”project” refers to the tangible artifact or deliverable. When both aspects are relevant, the terms are combined for clarity.

II. Objective: Leveraging AI-Powered Tools

This research aims to explore the potential of AI-powered tools, such as ChatGPT, in comparison with human developers throughout the SDLC. The objective is to examine how these tools can complement or enhance traditional development practices, particularly in tasks like requirements engineering, design, and testing. By comparing AI-generated artifacts with developer-created outputs, this research seeks to evaluate the effectiveness, limitations, and potential applications of AI tools in real-world software development contexts.

A. *Human-AI Comparison for Alignment*

Evaluate and compare human-generated and AI-generated requirements, including UML diagrams, class designs, and task breakdowns, to understand how AI-powered tools such as ChatGPT assist teams in aligning.

B. *Promoting AI Integration and Skill Building*

Encourage developers to explore and integrate AI-powered tools such as ChatGPT into their workflows, showcasing how these technologies can streamline processes, generate UML diagrams and class definitions, and assist in task allocation for improved project outcomes.

C. *Enhancing Collaboration and Prediction Insights*

Examine how AI-powered tools foster better collaboration, improve decision-making, and provide predictive insights. This includes their ability to generate accurate UML representations, refine class hierarchies, and enhance team dynamics by reducing discrepancies in task understanding especially in complex software projects.

D. *Unveiling the AI Behind the Scenes*

AI-powered tools are transforming the Software SDLC by automating key tasks such as requirements generation, user stories, architectural design, and coding. Tools like *ChatGPT*, *GitHub Copilot*, and *Tabnine* leverage advanced AI models to assist in creating textual descriptions, generating user stories, and providing intelligent code suggestions. Beyond coding, platforms like *Slack* and *Monday.com* enhance collaboration and project management, while tools such as *Probole AI* assist with generating UML diagrams and visualizing designs. By integrating natural language processing, machine learning, and deep learning algorithms these tools complement human efforts, streamline workflows, and enhance productivity across SDLC phases.

III. Methodology, Participants, and Research Setup

The study is designed to mimic real-world software development scenarios through a structured, multiphase approach, integrating tools like ChatGPT, powered by Generative AI, to enhance key aspects of the SDLC. This study was conducted as part of software engineering courses, allowing students to apply theoretical concepts to practical, hands-on experiences throughout the term. Each phase spans several meetings throughout the course term, providing students with ample time to engage in each step of the process. Below is a brief overview of each phase:

A. *Phase 1: Brainstorming and Developing Use Cases*

In this initial phase, participants as students formed teams and were engaged in brainstorming sessions to propose software projects based on their interests. This activity allowed teams to generate unique ideas while fostering creativity and ownership. The outcome was a diverse set of 12 use case studies, categorized into web, game, and mobile app development, covering a range of domains such as education, healthcare, games, entertainment, and e-commerce.

B. *Phase 2: Alternating Stakeholder and Developer Roles*

In this second phase, each team was randomly assigned a use case study proposed by another team assuming the role of developers, while the proposing team acted as the stakeholder. The teams engaged in dual-role interactions to review and clarify project requirements. These meetings were essential to ensure that both stakeholders and developers were aligned and shared a clear understanding of the project's scope and objectives.

C. *Phase 3: Developer-Generated User Stories*

In this phase, developer teams collaborated over several meetings to create user stories for each requirement of their assigned project. These user stories, crafted throughout multiple sessions, described the system's functionality and non-functionality, ensuring that both functional and non-functional requirements were addressed comprehensively. It is worth noting that teams were not informed about using AI to generate user stories until after they had completed and submitted their own. This element of surprise encouraged genuine reflection as they compared their human-generated user stories with those created by AI.

D. *Phase 4: AI-Generated User Stories*

In this phase, teams utilized Generative AI-Powered tools to produce user stories based on the project requirements discussed in Phases 2 and 3. This approach allowed students to explore how AI interprets and translates requirements into user stories, offering a unique perspective that complemented their own work. Additionally, it's important to note that while AI-generated user stories can be efficient and consistent, they may not fully capture the nuances of user needs and experiences. Therefore, it's advisable to use AI-generated user stories as a starting point and refine them through collaboration with stakeholders to ensure they accurately reflect user requirements.

E. *Phase 5: AI vs. Developers: A Comparative Study*

In the fifth phase, teams were engaged in several class meetings focused on comparing and analyzing developer-created user stories with AI-generated ones. Over multiple sessions, teams identified similarities, differences, and potential gaps between the two sets. This process allowed teams to evaluate AI's potential in assisting with requirements refinement and provided valuable insights into how AI can complement and enhance the work of human developers in improving project requirements.

F. *Phase 6: UML and Class Diagrams Integration*

In Phase 6, teams focused on creating UML components and class diagrams by blending the strengths of AI-generated textual descriptions with their own work. The AI generated textual representations of UML components and class diagrams, outlining the structure and relationships within the system. However, these descriptions lacked the visual clarity typically associated with UML diagrams and also omitted key elements such as interfaces. Teams then refined and expanded upon these descriptions, adding interfaces and translating the textual content into visual diagrams. This collaborative process allowed students to evaluate how AI could assist in the design phase while also emphasizing the critical role of human judgment and creativity in shaping the final software architecture.

G. *Phase 7: Interactive Scrum-Style Presentations*

In Phase 7, each team presented their findings in a scrum-style format, comparing their developer-created user stories with those generated by AI. Teams shared insights, highlighting key similarities, differences, and reflections on the process. Following each presentation, an engaging Q&A session encouraged collaboration, allowing teams to ask questions and exchange ideas. To guide the teams, a detailed rubric outlining the key points for their presentations was provided before their scrum presentations.

IV. Three Key Projects and Twleve Use Case Studies

This section presents three representative use case studies from the 12 assigned ones, spanning web development, game development, and mobile app development. Each team categorized user stories to structure the comparison between AI-generated and team-developed outcomes. Below are three examples of applications selected from different domains. Each example includes the application type, project title, a brief summary, and a sample of features. Due to space constraints, we do not include all functional and non-functional requirements for these projects.

A. *Use Case Study: Web Application Development*

1) Project Title: University Portal Development: FERPA-Compliant Student and Faculty Access

2) Summary: FERPA, the Family Educational Rights and Privacy Act, is a U.S. federal law protecting the privacy of student education records. This project involves developing a website to ensure FERPA compliance by providing role-based access to student information for staff and faculty. Key features include employee timesheet management, advisor access to transcripts and degree audits, and student access to update personal information, waivers, and agreements. The site integrates with a database, supports tab bookmarking, and offers intuitive navigation. Students can view records, degree audits, and course registration through the university portal. Authentication is required for all users to access the site.

3) Sample of Key Features: User Authentication, Faculty and Advisor Access, Data Base Access, User Experience and Interface, Security and Access Control.

B. *Use Case Study: Game Development*

1) Project Title: Mercenary's Quest: A Dynamic RPG of Combat, Progression, and Strategy

2) Summary: This project involves developing a dynamic RPG (Role-Playing Game) where players control a mercenary character in fast-paced combat. The game features fluid level design for seamless movement and battles, with enemies offering group-based challenges and unique skills for strategic gameplay. Dramatic visual effects enhance the power-fantasy experience, while character-based attacks add to player immersion. A progression system allows players to level up through quests and combat, customize abilities via a branching skill tree, and upgrade gear to strengthen their character.

3) Sample of Key Features: Combat-Friendly Level Design, Gameplay Mechanics, Group-Oriented Enemy Design, Performance Optimization.

C. *Use Case Study: Mobile App Development*

1) Project Title: Note-Taking App

2) Summary: This project involved developing a simple note-taking app with key features, including note-taking, customizable macros, and version control for saved files. The app also offered QR code generation, AI integration, and online access, allowing users to view their notes anytime.

3) Sample of Key Features: User Authentication, Create, Edit, Organize, and Delete Notes, Search and Filter Options, Reminders and Alerts, Export and Backup Funtionality, Security and Access Control.

V. A Comparative Analysis: Developers vs AI

This analysis compares team-generated and AI-generated requirement specifications and design across three application domains: web applications, mobile apps, and game development. The focus is on clarity, completeness, consistency, and relevance. Team specifications and design are created using structured frameworks (as outlined in Section III), while AI-generated specifications rely on carefully crafted prompts. Each feature is analyzed to compare strengths, weaknesses, and any innovative ideas introduced by AI.

AI performed well in web and mobile app development but struggled with game development, where its specifications and design failed to capture the domain's complexity and creativity. The findings highlight areas where the team excelled in specificity, where AI contributed innovative ideas, and any common gaps. The evaluation also examines how AI suggestions improved specifications, their adaptability to the project, and alignment with key criteria.

The goal is to assess the value of AI in enhancing the specification and design process and offer recommendations for balancing AI assistance with human expertise, especially in creative and complex domains like game development. The following summary presents findings from 12 use case studies completed by the teams across the three application domains: web applications, mobile apps, and game development.

A. A Web Application Development Use Case Study

1) Strengths of Human-Developer Team: The developers' user stories offered detailed insights into the website's design and functionality, demonstrating a strong understanding of the client's needs for user experience and interface.

2) Strengths of AI's Approach: The AI-generated user stories were general, providing a clear overview of broader user needs while effectively breaking down user roles and creating detailed stories for each type of user. Additionally, the AI generated these stories quickly and efficiently. The AI generated several user stories related to Student Access Feastures that the team had missed. Below are examples of user stories generated by AI and developers.

Student Access Features

- *Developers*: "*As a student, I want to update my personal information, such as contact details, so my information stays current*."
- *AI: "As a student, I need to view and acknowledge agreements (like laptop agreements or waivers) directly from my profile."*
- *AI: "As a student, I want to view my academic holds, eligibility, and awards, so I am aware of my academic standing."*

Advisor and Faculty Access Feature

- *Developers: "As faculty or staff, I need limited access based on my role to perform my job effectively."*
- *AI: "As a student advisor, I need the ability to view the transcripts and degree audits of the students I advise."*
- *AI: "As a faculty member with advising duties, I need access to academic records, within the limits of FERPA compliance."*

The AI version provided greater detail, specifically breaking down the roles of advisors and faculty members, while the developers took a broader, more general approach.

3) Common Gaps: The developer team grouped users by similar roles, whereas the AI generated more specific user stories. However, the AI missed certain navigation updates based on client feedback and overlooked critical aspects, such as database integration, that the developers had also not considered.

4) Conclusion: In summary, both human and AI-generated user stories addressed similar requirements but differed in their level of details AI provided a broad overview, while human input was essential for refining designs and understanding user needs. Developers recommend starting with human-generated stories, leveraging AI for initial drafts, and refining them to better align with client expectations.

AI failed to generate any UML diagrams or interface designs, instead providing textual descriptions. While these descriptions were helpful for outlining broader requirements, they lacked the visual clarity needed for detailed design work. Increased client interaction may also be necessary to capture specific needs more accurately. While AI can assist with generating general user stories and initial drafts, human expertise remains crucial for creating project-specific details, visual models, and ensuring alignment with unique requirements.

B. A Game Development Use Case Study

1) Strengths of Human-Developer Team: The developers' user stories were more comprehensive, clearly outlining the systems needed to deliver the gameplay experience the clients envisioned. They effectively conveyed the feel and features of the game.

2) Strengths of AI's Approach: AI excelled in specificity, linking each story directly to a feature requested by the user.

3) Common Gaps: Both versions struggled to capture the "game feel," which is challenging to express in text. While the AI's list format might be more suitable for an investor meeting, the human-generated stories were better suited for guiding the development team.

4) Conclusion: AI helped the development team generate a feature checklist but was less useful for game design. Its suggestions were generic and lacked originality. Additionally, it overlooked key project criteria, such as feature completeness and stability, which were crucial for the client.

C. A Mobile App Development Use Case Study

1) Strengths of Human-Developer Team: The strengths of the developer teams were consistency, conciseness, relevance, and completeness. The team ensured that all components were aligned with the system as a whole, using consistent terminology and addressing both user and admin functionality. The team's model was more consistent in meeting all requirements for both sides.

2) Strengths of AI's Approach: AI generated more user stories per point and quickly captured key aspects, but there were some gaps.

3) Common Gaps: Both models, developer teams and AI, missed certain details, often due to unclear instructions or misunderstandings. Additionally, the AI used terms interchangeably, which led to confusion.

4) Conclusion: AI lacked accuracy and consistency in some details. The team's model was stronger in completeness and relevance. AI is useful for quickly generating key features but needs careful review, as it can make mistakes or focus on irrelevant aspects. It's a helpful tool, but human oversight is essential.

VI. AI -Powered Tools: Enhancing the Research Process

ChatGPT, a key AI tool, excels in generating user stories, summarizing stakeholder requirements, and suggesting task allocations. However, it fails to generate UML component and call diagrams, instead providing textual, often fuzzy descriptions. These tools help Scrum teams stay aligned, improve productivity, and streamline processes. Several AI-powered tools are enhancing this analysis. Jira with Automation and Monday.com help manage and prioritize tasks by automating routine activities such as updating boards and generating reports. Slack AI and Microsoft Teams AI improve communication by summarizing meetings, tracking decisions, and sharing updates. Otter.ai automates meeting transcriptions, providing accurate summaries and action items, which enhance communication and documentation in Scrum.

While AI's suggestions were generally useful, it lacked an understanding of UML best practices, organizing priorities sequentially instead of by priority tiers. As a result, its output would need restructuring to align with UML standards and the architectural design. For future projects, we recommend using AI to generate initial user stories, followed by refinement based on UML guidelines, architectural design principles, and client needs. By structuring the user stories in this manner, each requirement is categorized based on user roles, with functional and non-functional aspects clearly defined and prioritized. This approach helps align the development process with project goals and stakeholder needs.

VII. CONCLUSION

This research compared human-developed and AI-generated requirements elicitation and analysis, highlighting the strengths and limitations of Generative AI in software development. Conducted within the context of Software Engineering courses—where students work as teams playing dual roles as both developers and stakeholders, the research explored how AI and human expertise complement each other. The teams collaborated in teams on 12 use case studies spanning three application domains: web applications, mobile apps, and game development, with projects covering diverse areas such as health, education, gaming, e-commerce, events, sports, and more.

Human developers, who were the teams, excelled in understanding context, stakeholder needs, and detailed requirements, while AI provided valuable insights, particularly in structured tasks like web and mobile app development. However, AI's limitations were evident in game development, where it struggled to capture the creativity and complexity essential for success. Additionally, attempts to integrate AI with modeling tools like UML diagrams revealed that AI currently only generates textual descriptions, falling short of producing the necessary graphical models for system architectures. AI-powered tools show promise in requirements elicitation and analysis, especially in structured and contextual domains, but their challenges in creative fields, such as game development, call for further refinement. Future research should focus on improving AI tools or better integration into software processes, particularly in generating graphical models and architectural designs. There is also potential to explore how AI can enhance decision-making, collaboration, and the broader software engineering lifecycle. This underscores the need for a balanced approach, where AI complements but cannot replace the creative and contextual decision-making needed in certain projects.

Acknowledgment

This research was supported by funding from Northwest Missouri State University. I would like to express my gratitude to the university for its support in facilitating this study. In addition, my thanks go to the students enrolled in the Software Engineering course for their participation in this study.

References

Ahmad, K., Bano, M., et al. (2021). What's up with requirements engineering for artificial intelligence systems? Proceedings of the 2021 IEEE 29th *International Requirements Engineering Conference* (RE), 1–12. https://doi.org/10.1109/RE51729.2021.00011

Amershi, S., Begel, A., Bird, C., et al. (2019). Software engineering for machine learning: A case study. *Proceedings of the 41st International Conference on Software Engineering (ICSE '19)*, 291–300. https://doi.org/10.1109/ICSE.2019.00042

Barenkamp, M., Rebstadt, J., & Thomas, O. (2020). Applications of ai in classical software engineering. AI Perspectives, 2 (1), 1. https://doi.org/10.1186/s42467-020-00005-4

Bhandari, K., Kumar, K., & Sangal, A. (2023). Artificial intelligence in software engineering: Perspectives and challenges. 2023 Third *International Conference on Secure Cyber Computing and Communication* (ICSCCC), 133–137.

Briand, L. C., Bianculli, D., Nejati, S., et al. (2021). The case for context-driven software engineering for AI-based systems. *IEEE Software, 38*(6), 51–60. https://doi.org/10.1109/MS.2021.3106993
https : / / doi . org / 10 . 1109 /ICSCCC58608.2023.10176436

de Oliveira Santos, P., Figueiredo, A. C., et al. (2024). Impacts of the usage of generative artificial intelligence on software development process. *Proceedings of the 20th Brazilian Symposium on Information Systems* (SBSI '24), 1–9. https://doi.org/10.1145/ 3658271.3658337

Habiba, U., Haug, M., et al. (2024). How mature is requirements engineering for ai-based systems? a systematic mapping study on practices, challenges, and future research directions. Requirements Engineering, 29, 567–600. https://doi.org/10.1007/s00766-024-00432-2

Heyn, H.-M., Knauss, E., et al. (2021). Requirement engineering challenges for AI-intense systems development. *2021 IEEE/ACM 1st Workshop on AI Engineering – Software Engineering for AI (WAIN)*, 89–96. https://doi.org/10.1109/WAIN52551.2021.00020

Horkoff, J., Hadar, I., et al. (2023). AI-assisted software engineering: Promises, pitfalls, and research opportunities. *ACM Computing Surveys, 55*(7), 1–38. https://doi.org/10.1145/3579777

Khlood, A., Mohamed, A., et al. (2023). Requirements engineering for artificial intelligence systems: A systematic mapping study. *Information and Software Technology, 158*, 107176. https://doi.org/10.1016/j.infsof.2023.107176

Lorenzoni, G., Alencar, P., Nascimento, N., & Cowan, D. (2021). Machine learning model development from a software engineering perspective: A systematic literature review. arXiv preprint, arXiv:2102.07574. https://arxiv.org/abs/2102.07574

Marar, H. (2024). Advancements in software engineering using AI. *Computer Software and Media Applications, 6*(3906). https://doi.org/10.24294/csma.v6i1.3906

Smith, J., Johnson, E., Davis, M., et al. (2024). Exploring the future of software engineering with ai. *Journal of Software Engineering Research*, 40 (5), 123–145. https://doi.org/10.1016/j.jsser.2024.01.001

Terragni, V., Roop, P., & Blincoe, K. (2024). The future of software engineering in an AI-driven world. *Journal of Software Engineering*. https://doi.org/10.48550/arXiv.2406.0773

Todorov, P. G. (2022). The application of artificial intelligence in software engineering. *International Journal of Advanced Multidisciplinary Research Studies, 2*(5), 835–842. https://doi.org/10.2139/ssrn.4943407

Yoshioka, N., Husen, J., et al. (2021). Landscape of requirements engineering for machine learning-based AI systems. *Proceedings of the Asia-Pacific Software Engineering Conference Workshops (APSECW'21)*, 5–8. https://doi.org/10.1109/APSECW53869.2021.00011