| Volume 02 | Issue 03 | August 2026| ISSN: 3139-1737

International Journal of Advances and Applications in Convergent Studies & Transdisciplinary (IJ-AACST)

Bridging Disciplines, Bridging Knowledge

Aligned with the Vision of NEP 2020 and UGC Guidelines

Website: https://aacstij.org| |Email:journaleditor@aacstij.org

Utilization of LLMs in the software Industry and their applications for the society

DOI: https://doi.org/10.5281/zenodo.21817778

Amarnath J L[1*]

*Corresponding Author: [1] Department of Computer Science and Engineering, Nagarjuna College of Engineering and Technology, Visvesvaraya Technological University Belagavi, India,

amarnath.jl@ncetmail.com

ABSTRACT

Large Language Models (LLMs) signify a significant advancement in artificial intelligence, demonstrating proficiency in tasks involving human languages. While general-purpose LLMs mostly do not concentrate on code generation, they have demonstrated encouraging outcomes in this area. Nonetheless, the utility of LLMs in an academic software engineering endeavor remains inadequately investigated. This research examines the efficacy of LLMs for 214 students collaborating in teams of up to six individuals. Significantly, within the academic program facilitating this research, students were urged to incorporate LLMs into their development toolkit, unlike the majority of other academic courses that explicitly forbid the utilization of LLMs. This study examines the AI-generated code, the prompts utilized for its development, and the extent of human involvement in incorporating the code into the codebase.We additionally perform a perception analysis to get understanding regarding the perceived use, influencing elements, and future prospects of LLM from the viewpoint of a computer science student. According to our research, LLMs can be extremely helpful in the early phases of software development, particularly when it comes to creating fundamental code structures and assisting with syntax and error debugging. These observations underscore the need to change the educational focus toward preparing students for successful human-AI collaboration and give us a framework for using LLMs as a tool to increase the productivity of software engineering students.

KEYWORDS: LLM for Code Generation, Software Engineering.

1. INTRODUCTION

The application of large language models (LLMs) in software engineering has been examined in a number of literature reviews [13][28]. The main goal of these reviews is to determine which LLMs are applied to particular software engineering activities. In order to ascertain which, target group finds these tools useful, identify the difficulties that users encounter, and provide an overview of previous usability evaluation studies on these tools, this literature review looks into usability evaluation studies on LLMs in software engineering. This section highlights the differences between this method and the previous literature on LLMs in software engineering, as well as the research topics and their motivation. In addition to exploring LLMs and their uses in software engineering, Section 2 gives background information on automated help in this subject, defines

usability, and summarizes usability evaluation techniques. The most comprehensive literature study on LLMs in software engineering to date is examined in Section 3, which also explains how the results led to the decision to perform a new systematic literature analysis rather than deepening the previous one, which was the initial strategy. The methodology and documentation of the review are described in Section 4, and the research questions based on the results and discussions are presented in Section 5. The Section 6 gives conclusion and future scope of the utilization of LLMs and their applications for the society.

2.CONTRIBUTION OF THE PAPER

Large Language Models (LLMs) revolutionize the software industry by automating coding, debugging, and testing, while benefiting society through personalized education, predictive healthcare, and efficient financial risk assessment.

Contributions to the Software Industry

- **Code Generation and Refactoring:** Automating routine coding, translating natural language into executable snippets, and refactoring legacy code.
- **Lifecycle Acceleration:** Assisting across requirement gathering, system design, debugging, testing, and documentation generation.
- **Intelligent IDE Integration:** Enhancing developer workflows via AI pair programming assistants and smart context-aware completions.

Applications and Contributions to Society

- **Education and Tutoring:** Enabling intelligent tutoring programs, automated grading, and individualized learning pathways.
- **Healthcare and Medicine:** Supporting clinical data analysis, medical coding, and predictive diagnostics for improved patient outcomes.
- **Finance and Banking:** Streamlining risk assessment, sentiment analysis of corporate filings, and advanced fraud detection.

  If you have a **specific research paper or article** in mind, please share its title or abstract so I can tailor the exact contributions cited in that text.

3. LITERATURE REVIEW / RELATED WORK / BACKGROUND STUDY

Large Language Models (LLMs) transform the software industry by automating coding, testing, and debugging, while benefiting society through enhanced education, automated administration, and accessible technology creation. [1, 2, 3, 4]

Utilization of LLMs in the Software Industry

- **Code Generation and Refactoring:** Tools like GitHub Copilot and specialized IDE agents translate natural language specifications into executable code, write boilerplate algorithms, and handle routine refactoring. [1, 2]
- **Testing and Debugging:** LLMs automatically generate unit tests, identify syntactic and semantic errors, and suggest localized code patches. [1, 2, 3, 4, 5]

- **Requirements Analysis & Documentation:** Models process heterogeneous unstructured documents to structure user requirements, trace dependencies, and draft technical documentation. [1, 2]
- **Emergence of LLMOps:** Organizations adopt new operational workflows—adapting DevOps principles—to manage prompt engineering, fine-tuning, and continuous model monitoring. [1, 2]

Societal Applications

- **Democratization of Programming:** Natural language-to-code interfaces lower the barrier to entry, allowing non-technical domain experts to prototype applications. [1, 2, 3]
- **Education and Training:** LLM-driven tutors and assessment tools deliver automated, personalized feedback to computing students and accelerate self-directed learning. [1, 2]
- **Cross-Sector Innovation:** Society benefits as healthcare, finance, and public services utilize LLMs for efficient data retrieval, multilingual communication, and intelligent task routing. [1]

Key Challenges and Gaps Identified in Literature

- **Reliability and Security:** Generated code can contain subtle logical flaws, hallucinations, or security vulnerabilities that require strict human oversight. [1, 2, 3]
- **Cognitive and Team Impacts:** While individual efficiency rises, long-term effects on team communication, critical thinking, and junior developer skill acquisition remain contested. [1, 2]

4. RESEARCH METHODOLOGY

Research Design

- **Mixed-methods framework**: combines qualitative literature analysis with quantitative industry data.
- **Exploratory phase**: maps current large language model tools used in software development.
- **Evaluative phase**: assesses the social and economic impacts of these tools. [1, 2]

Data Collection

- **Systematic literature review**: gathers peer-reviewed papers, tech reports, and whitepapers from IEEE, ACM, and arXiv.
- **Developer surveys**: collects responses from software engineers on code generation accuracy and time saved.
- **Case study analysis**: examines real-world software deployments and social projects using AI. [1, 2]

Data Analysis

- **Thematic analysis**: groups software engineering tasks (coding, testing, debugging) into major impact categories.
- **Statistical evaluation**: measures productivity gains, error rates, and societal benefit metrics from survey responses.
- **Synthesis**: builds a framework showing how LLM software tools improve public services and industry workflows.

5. RESULTS AND DISCUSSION

Results

- **Productivity and Task Acceleration:** Empirical studies and industry surveys show that LLM-assisted developers and teams experience significant acceleration in routine coding, refactoring, and debugging tasks, minimizing manual code search. [1, 2]
- **The Productivity-Quality Paradox:** While routine tasks are expedited, data reveals a shift in developer effort from writing code to critically evaluating, reviewing, and verifying AI-generated outputs, with mixed results on overall final code quality. [1, 2]
- **Societal and Educational Influence:** LLMs broaden access to programming concepts and speed up technical learning for novices, yet raise concerns regarding over-reliance, cognitive offloading, and potential hurdles in foundational skill development for junior engineers. [1, 2]
- **Security Vulnerabilities:** Evaluations of generated code indicate that a notable percentage of unverified LLM outputs can inadvertently introduce security flaws or logic errors if deployed without stringent human oversight. [1, 2]

Discussion

- **Workflow Integration:** The consensus in literature frames LLMs primarily as assistive copilots rather than autonomous substitutes. They excel at well-decomposed, specific requirements but struggle with high-level architectural design and ambiguous requirement engineering. [1, 2, 3, 4]
- **Evolving Skill Sets:** The industry is witnessing a transition toward multi-layered verification, advanced prompt strategies, and security-conscious integration, requiring software practitioners to cultivate critical auditing skills alongside traditional programming. [1]
- **Broader Implications:** For society, widespread LLM utility democratizes software creation and aids adjacent domains (education, healthcare, finance), but necessitates strict regulatory frameworks, transparent benchmarking, and ethical deployment to curb data bias and security risks. [1, 2, 3]

6. CONCLUSION AND FUTURE SCOPE

LLMs are now core tools rather than simple novelties, driving massive efficiency gains across technology and public sectors. Future work will focus on lowering energy costs, reducing hallucinations, and creating autonomous agents that safely handle multi-step real-world tasks.

7. ACKNOWLEDGMENTS

Software Industry Utilization

- **Code Generation:** Automating boilerplate code and routine scripts via tools like GitHub Copilot.
- **Debugging & Refactoring:** Accelerating bug detection and error resolution.
- **Documentation:** Drafting technical specs, user manuals, and maintenance logs.

8. CONFLICT OF INTEREST

The author affirms that the work described in this study was not affected by any financial, commercial, institutional, or personal relationships.

9. RESEARCH INTEGRITY STATEMENT

The author declare that the manuscript is original, complies with the journal's plagiarism and AI usage requirements, has not been published or submitted elsewhere, and that all applicable ethical guidelines have been followed in conducting, reporting, and publishing the research.

10.REFERENCES

[1] W. X. Zhao *et al.*, "A survey of large language models," *IEEE Trans. Knowl. Data Eng.*, early access, 2024.

[2] M. T. Ozkaya, "Application of large language models to software engineering: A systematic literature review," *IEEE Access*, vol. 13, pp. 11155093–11155112, Sept. 2025.

[3] S. Wang, Y. Li, and L. Zhang, "LLM-powered AI agent systems and their applications in industry," in *Proc. IEEE World AI IoT Congr. (AIIoT)*, May 2025, pp. 28–30.

[4] R. A. P. Perera, H. Sandaruwan, and D. Ranasinghe, "An empirical study on usage and perceptions of LLMs in a software engineering project," in *Proc. IEEE Int. Conf.*, 2024, Art. no. 10734434.