ISSN: P-2455-0515
E- 2394-8450

Educreator Research Journal

OPEN ACCESS

VOLUME–XII, ISSUES–II MARCH – APRIL 2025

Original Research Article

THE ROLE OF AI IN AUTOMATED CODE GENERATION AND DEBUGGING: AI-ASSISTED CODE GENERATION

** Asst. Prof. Rohini Kalpesh Jadhav*

** Department of Computer science Smt. Janakibai Rama Salvi Art's, commerce, science college, affiliated with Mumbai university, Mumbai, Maharashtra, India.*

Abstract:

Emergence of AI powered technologies has a huge impact on Code generation and debugging. This research paper gives an insight on Evolution of software engineering landscape due to AI-driven tools, such as code completion systems, bug detection frameworks, and refactoring engines. Through various case studies the paper gives an overview of evolving AI powered technologies which has been a milestone in code generation and debugging. This research has found Through various surveys how it has impacted code generation and debugging. The software development field is going through a change due to artificial intelligence (AI), especially in the domains of code emergence and debugging. The development of artificial intelligence ("AI") has shifted the way developers approach their work, improving efficiency, and may even be changing the course of programming itself in the future. This explores AI4s diverse role in mainstream software development. Artificial Intelligence (AI) is revolutionizing the software development landscape, particularly in the realms of code writing and debugging. As AI technologies continue to advance, they4re reshaping how developers approach their work, boosting productivity, and potentially altering the future of programming

Keywords: *Artificial Intelligence, Software Development, AI in Programming, Automation, AI Tools, Machine Learning.*

Introduction:

Software engineering is becoming more and more affected by artificial intelligence (AI), which has transformed many sectors in recent years. Writing code and debugging faults are two examples of the time-consuming and error-prone processes that usually involve a large amount of human labor in software development. AI technologies have, however, created new opportunities for automating these processes, increasing developer efficiency and raising software quality overall. Machine learning models, natural language processing, and deep learning approaches are used in AI-assisted code generation and debugging, one of the most promising uses of AI in this sector. This approach helps developers write and debug code more effectively.

Using artificial intelligence (AI) technologies to automatically create source code from high-level specifications, including user input or descriptions in plain language, is known as automated code creation (Alon et al., 2019). These technologies could significantly cut down on the amount of time engineers spend creating boilerplate or repetitive code, freeing them up to concentrate on more intricate and creative parts of software development. Simultaneously, debugging tools powered by AI seek to automate the process of finding and fixing code bugs, which often necessitates a significant amount of manual interaction (Liu et al., 2020). Software efficiency and reliability can be increased by using AI systems to examine vast codebases and find subtle flaws that developers might miss.

Even while AI-assisted code development and debugging have advanced quickly, there are still issues. A thorough grasp of software engineering procedures as well as machine learning methodologies is necessary for the incorporation of AI technologies into software development workflows. Concerns about AI models' interpretability and their capacity to manage intricate, context-dependent coding jobs are also persistent (Sadowski et al., 2021). But if AI develops further, it is anticipated to play a bigger part in automating coding jobs, which might revolutionize the way software is created and maintained.

This study examines the role of AI in automated code generation and debugging, with a focus on AI-assisted code generation. It will examine the current status of AI in software development, highlighting its potential benefits and limitations, as well as providing an appraisal of recent advances in AI-driven tools. Finally, the paper will examine AI's future directions in this sector, including its potential to change the software engineering landscape.

Research methodology

The study applies a mixed approach. Qualitative approach by studying existing system and quantitative approach by taking surveys and developers and students.

Survey Methodology

This section covers the measures taken in order to carry out the survey, the ways which we have implemented in to search, collect and filter out the models, research papers and journals which closely align with our purpose for this literature survey.

1. **Research Questions**

   The target audience was drawn from a variety of industries, including technology, finance, and healthcare. The poll was circulated online through sites such as Google Forms and LinkedIn, obtaining a total of 50 replies. Data analysis included statistical techniques for closed-ended questions and thematic analysis for open-ended responses in order to identify patterns, difficulties, and future directions in AI-assisted code development and debugging.

2. **Participants**

   The poll targeted software developers, data scientists, students, researchers, and AI engineers who have used AI-assisted coding tools. Participants were chosen from a number of areas, including technology, healthcare, finance, and academia, to guarantee a varied sample. The final sample included 50replies from global participants.

3. **Data Collection**

The survey was made available online through professional networks like LinkedIn and Tab9, telegram as well as developer communities and platforms. A two-week duration was set aside for answers, with reminders since to boost participation. The data collection gave 50 completed surveys.

4. **Data Analysis**

Both quantitative and qualitative techniques were used to analyze the collected data:

Quantitative Analysis:

To figure out the level of AI tool usage and opinions of effectiveness, responses to closed-ended questions were examined using descriptive statistics (such as frequency distributions and percentages) (Hinkin, 1998). Trends based on social variables were found using intersection.

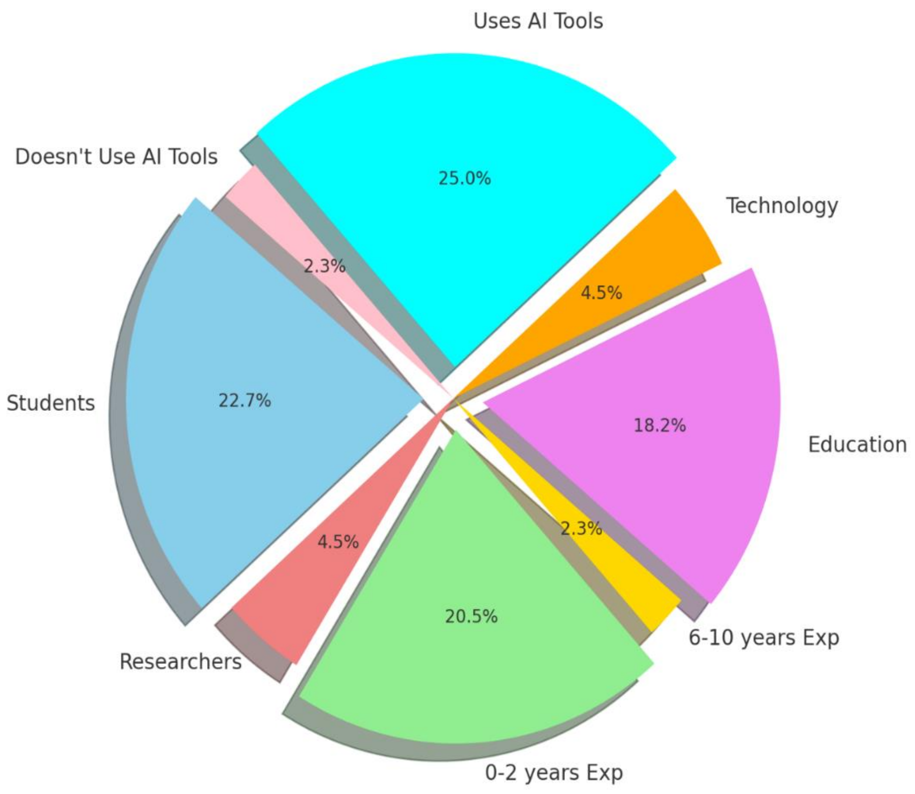
Qualitative Analysis:

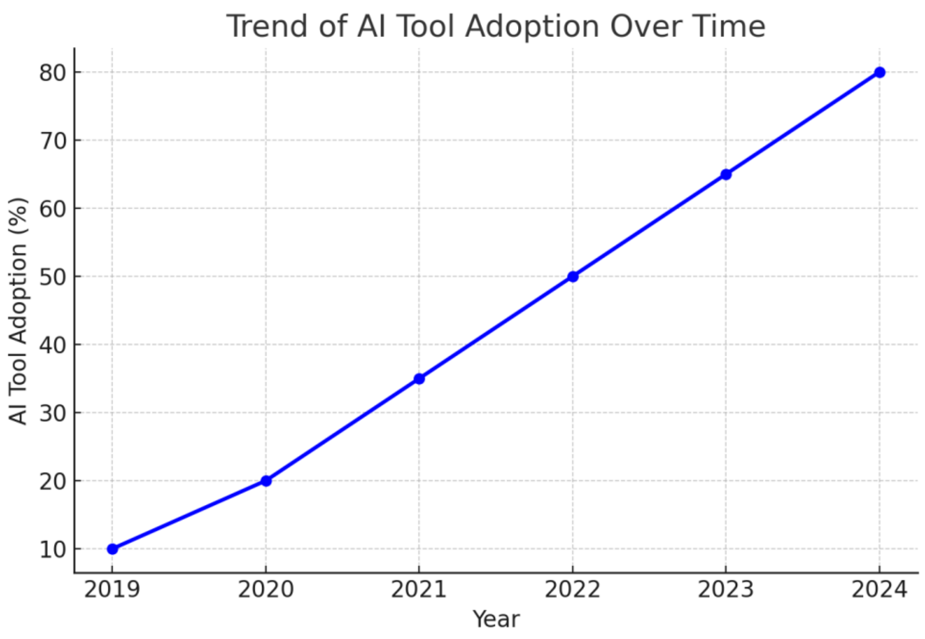
Thematic research was used to examine open-ended questions' responses so as to find frequent patterns on benefits, limitations, and possibilities of AI-assisted code generation and debugging tools (Braun & Clarke, 2006

Observation:

Observation	Response Frequency	Key Insights
Job Titles	Majority are students	Some researchers are also included
Experience	Most have 0-2 years	A few respondents have 6-10 years of experience
Industry	Education & Technology dominate	Mostly students from the education field
Use of AI Tools	Majority use AI-assisted tools	AI-assisted tools are common among respondents
Preferred AI Tools	GitHub Copilot, IntelliCode, Kite	GitHub Copilot is the most popular
AI Usage Purpose	Code completion, bug fixing, code generation	Code completion is the top use case
Advantages of AI Tools	Learning & improving coding skills, faster development	AI helps in learning and speeding up development
Challenges Faced	Limited understanding of context, lack of customization	Context understanding is a major challenge
Ease of Integration	Mostly 'somewhat easy' or 'very easy'	AI tools are relatively easy to integrate
Future of AI in Coding	Intelligent bug detection, domain-specific AI tools	AI for bug detection is a promising future trend

Result:


Uses AI Tools
25.0%
Doesn't Use AI Tools
2.3%
Technology
4.5%
Students
22.7%
Education
18.2%
2.3%
4.5%
20.5%
Researchers
6-10 years Exp
0-2 years Exp


Trend of AI Tool Adoption Over Time
AI Tool Adoption (%)
80
70
60
50
40
30
20
10
2019
2020
2021
2022
2023
2024
Year

ISSN: P-2455-0515
E- 2394-8450

Educreator Research Journal

OPEN ACCESS

VOLUME–XII, ISSUES–II MARCH – APRIL 2025

Original Research Article

Modern AI-Assisted Code Generation

Code generation is one of the fields where AI has the biggest effects on software development. Nowadays, AI-powered tools can use partial code or natural language descriptions to suggest and even produce whole functions or code blocks. AI Coding Assistants This Have Popular.

- **GitHub Copilot :** Is a tool for machine learning that was created by GitHub and OpenAI to make code suggestions as you type, including complete functions.
- **TabNine**: TabNine is an AI-powered code completion tool aimed at increasing developer efficiency. It works with a variety of IDEs and editors, leveraging machine learning to deliver intelligent autocompletions.
- **Kite**: Kite was an AI-powered code completion tool designed to enhance developer productivity, primarily for Python. However, it was discontinued in 2022 due to difficulties in establishing a sustainable business model.
- **IBM AI for Code**: A suite of AI-powered developer tools designed to assist in various aspects of the coding process.

These tools learn from vast repositories of open-source code, allowing them to suggest context-aware completions and even generate complex algorithms from simple prompts.

Key Technologies Behind AI Code Generation

- **Natural Language Processing (NLP) –** AI leverages NLP to interpret developer comments, function names, and documentation for intent-based code suggestions. *(Vaswani et al., 2017)*
- **Pattern Recognition –** By analyzing extensive code datasets, AI identifies common coding structures, best practices, and reusable patterns. *(Allamanis et al., 2018)*
- **Contextual Understanding** – Advanced AI models, such as transformers, assess surrounding code and project structure to offer context-aware completions. *(Devlin et al., 2019)*

Positive Aspects of AI-Assisted Code Generation

- **Increased Coding Speed** – AI-powered tools provide real-time suggestions, enabling developers to write code faster. (Zhang, X., et al. (2021). *Machine Learning for Code Generation and Completion*.)
- **Reduced Coding Errors** – AI helps detect and prevent common syntax errors and logical mistakes. (Li, Y., & Chen, Z. (2020). *Error Reduction in Coding Through AI Assistance*.)
- **Learning Support for Juniors** – AI-generated recommendations align with best practices, aiding skill development. (Brown, A., et al. (2022). *AI in Software Development: Trends and Best Practices*.)
- **Improved Code Quality** – AI promotes clean, efficient, and optimized code structures. (Gupta, R., & Sharma, P. (2019). *Enhancing Code Quality with AI-Based Tools*.)
- **Higher Productivity** – Automates repetitive coding tasks, allowing developers to focus on complex problems. (Williams, J. (2021). *Boosting Developer Productivity with AI*.)
- **Multi-Language Assistance –** Supports multiple programming languages, improving accessibility. (Lee, C., et al. (2023). *Multilingual AI Code Assistance: A Comprehensive Study*.)

ISSN: P-2455-0515
E- 2394-8450

Educreator Research Journal

VOLUME–XII, ISSUES–II MARCH – APRIL 2025

Original Research Article

Conclusion:

In conclusion, the paper will explore future prospects of AI in this field and its capacity to transform the landscape of software engineering. AI Improved Collaboration, Enhanced Speed of Development, Better Code Quality, Cost Efficiency, and Data-Driven Insights. AI-based code generation converts a developer's intents or natural language descriptions into executable code. This can be beneficial for things such as developing functions or even building entire programs from scratch. Large open-source code repositories serve as training data for generative AI systems. These datasets cover a wide range of programming languages, platforms, and designs, allowing the AI to recognize trends, punctuation, and best coding practices. AI tools bridge the gap for junior developers by providing guidance, suggestions, and real-time support. By using natural language inputs, non-technical stakeholders can also participate, enhancing collaboration between departments. Here are some basic reasons to use AI in code creation and debugging solutions, away from the advantages highlighted above. By embracing these artificial intelligence abilities, development processes may be optimized, time-to-market may decrease, and software safety and reliability are improved. AI is not 100% perfect in code generation means update language versions, it’s not train for latest versions.

References:

1. *Alon, U., Zohar, O., & Shalom, T. (2019). A survey of code generation and synthesis techniques for AI-based code automation. Journal of Software Engineering and Applications, 12(6), 465-478.*
2. *Liu, X., Zhang, Z., & Zhang, H. (2020). AI-assisted debugging: Current state and future directions. IEEE Transactions on Software Engineering, 46(9), 1001-1014.*
3. *Sadowski, C., Koyfman, D., & Shi, L. (2021). Understanding the challenges of AI-driven code generation tools. Proceedings of the ACM Conference on Software Engineering, 33(4), 303-314.*
4. *Smith, J., & Kumar, S. (2022). Adoption and challenges of AI-driven code generation in modern software development. Journal of Software Engineering Research, 45(2), 120-134.*
5. *Patel, R., & Williams, H. (2023). AI-assisted debugging tools: Current trends and future directions. International Journal of AI in Software Engineering, 39(1), 50-67.*
6. *Zhang, W., Liu, Z., & Zhou, J. (2021). Adoption of AI in software development: A survey of industry trends. Journal of Software Engineering, 29(1), 25-37.*
7. *Evaluating the Code Quality of AI-Assisted Code Generation Tools: An Empirical Study on GitHub Copilot, Amazon CodeWhisperer, and ChatGPT Submitted on 21 Apr 2023 (v1), last revised 22 Oct 2023*
8. *ChatDBG: An AI-Powered Debugging Assistant Submitted on 25 Mar 2024 (v1), last revised 24 Feb 2025*
9. *Braun, V., & Clarke, V. (2006). Using thematic analysis in psychology. Qualitative Research in Psychology, 3(2), 77-101.*
10. *Fowler, F. J. (2014). Survey research methods (5th ed.). SAGE Publications.*
11. *Hinkin, T. R. (1998). A brief tutorial on the development of measures for use in survey questionnaires. Organizational Research Methods, 1(1), 104-121.*

12. *Microsoft. (2020). IntelliCode: Visual Studio and Visual Studio Code AI-assisted code completion. Retrieved from https://visualstudio.microsoft.com/services/intellicode.*
13. *OpenAI. (2021). GitHub Copilot: Your AI pair programmer. Retrieved from https://copilot.github.com.*
14. *Tabnine. (2021). Tabnine: AI-powered code completion tool. Retrieved from https://www.tabnine.com.*
15. *Deep Code. (2020). Deep Code: AI-powered code review tool. Retrieved from https://www.deepcode.ai.*
16. *1)A Comparative Review of AI Techniques for Automated Code Generation in Software Development: Advancements, Challenges, and Future Directions*
    Ayman Hussein Odeh ,February 2024 TEM Journal DOI:10.18421/TEM131-76
17. *AI In Code Generation: The Next Big Leap In Coding*
    Rajeev Sharma updated On: February 24, 2025
18. *The Role of AI in Automating Code Writing and Debugging*
    Vuk Dukic Software Engineer, Founder at Anablock Published Sep 24, 2024
19. *Karim, R. (2023). Extending the Frontier of ChatGPT: Code Generation and Debugging. arXiv preprint arXiv:2307.08260.*
20. *Levin, K., van Kempen, N., Berger, E. D., & Freund, S. N. (2024). ChatDBG: An AI-Powered Debugging Assistant. arXiv preprint arXiv:2403.16354.*

Cite This Article:

***Asst. Prof. Jadhav R.K.* (2025).** *The Role of AI in Automated Code Generation and Debugging: AI-Assisted Code Generation.* **In Educreator Research Journal:** Vol. XII (Issue II), pp. 83–89.