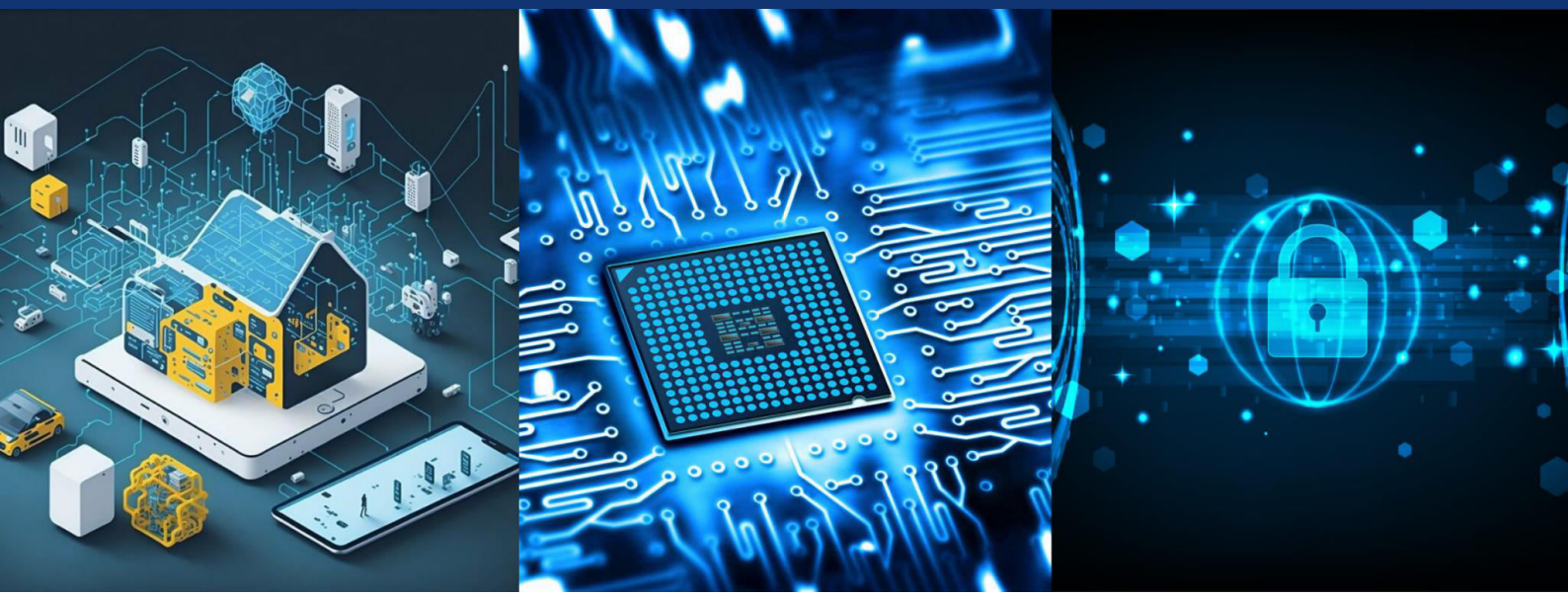
IJIRCCE

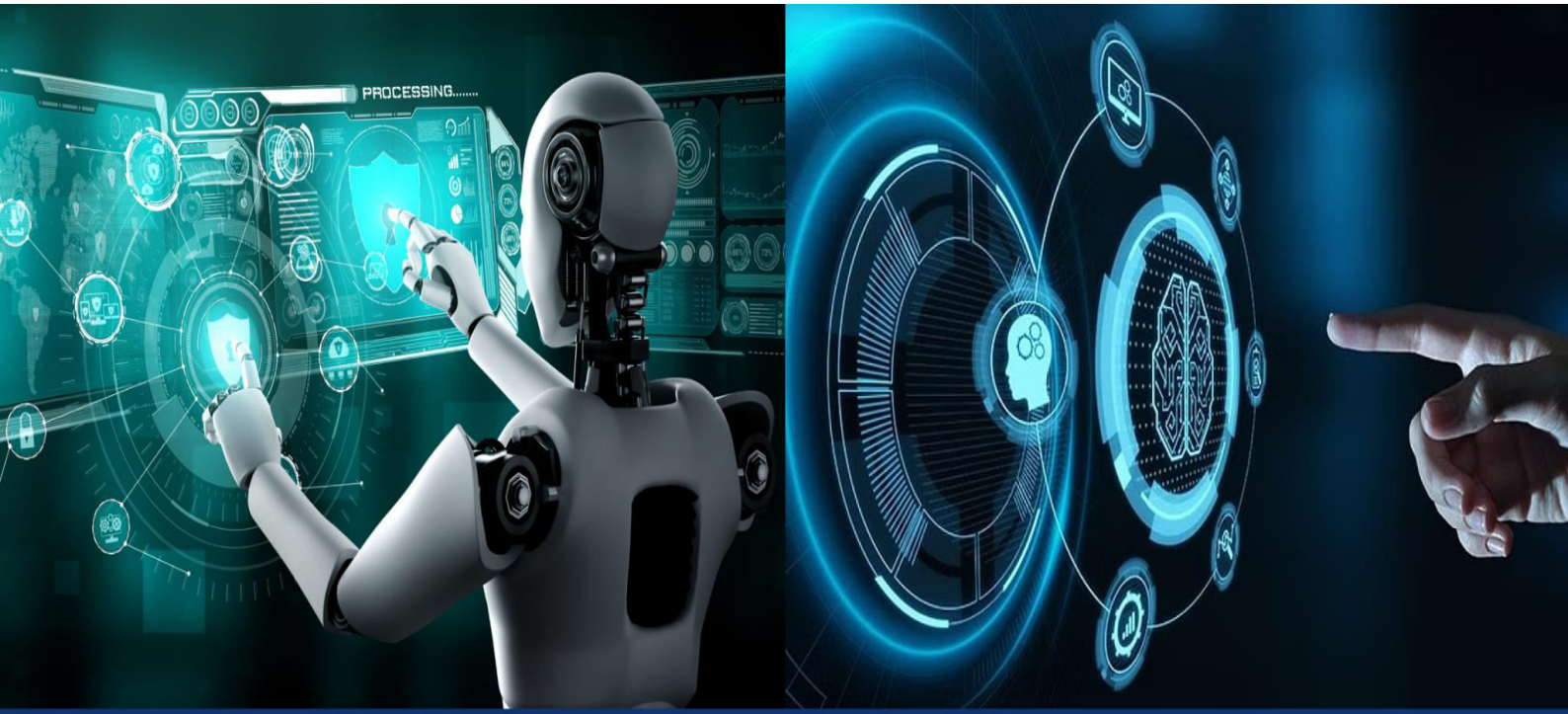
ISSN(O): 2320-9801

ISSN(P): 2320-9798



International Journal of Innovative Research in Computer and Communication Engineering

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)



Impact Factor: 8.771

Volume 13, Issue 4, April 2025

© 2025 IJIRCCE | Volume 13, Issue 4, April 2025 |DOI:10.15680/IJIRCCE.2025.1304277

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

Revolutionizing Software Engineering through Artificial Intelligence

Amruta Navale, Bharati Bhamare, Sneha Pawar

Assistant Professor, Department of Computer Science, Dr. D. Y. Patil A.C.S College, Pimpri, Pune, Maharashtra, India

ABSTRACT: This study considers the impacts of AI on software engineering, a discipline that has undergone drastic changes due to AI evolve. It analyzes the impact of AI on the qualitative and quantitative aspects of software development, testing, maintenance, and project management. Moreover, address the challenges posed by A\Mail Artificial Intelligence (AI implications of technological evolution, automation and optimization for software lifecycle functions. It also covers advances in AI as well as the developments and AI can bring in the future in this area.

KEYWORDS: AI-driven Development, AI Advancements, AI Impact, Automation, Optimization, Software Maintenance

I. INTRODUCTION

The design, development, maintenance, and testing of computer systems falls under the umbrella of software engineering. For centuries, methods of software engineering have heavily relied on the human craft and skills. Traditional methods, however, are being upended by the rapid development of tools powered by artificial intelligence (AI), forecasting models, and automation, ushering in a new era of intelligent systems. A plethora of software systems engineering now incorporate AI technologies like machine learning (ML), natural language processing (NLP), and automated reasoning to enhance efficiency, reduce the margins of error, and aid in more informed choices.

AI application has shifted from requirements gathering to maintenance, marking its influence on every phase of the software life cycle, and along with it the development, testing, and management processes of software. The potential artificial intelligence technologies hold in enhancing the quality of software, cutting down costs, and optimizing the timelines for development becomes more pronounced as these technologies continue to evolve.

II. RELATED WORK

1. AI in Software Development

Artificial intelligence (AI) is transforming software development by enhancing accuracy, efficiency, and automation in key tasks such as algorithm creation, defect localization, and code refinement. AI tools are being employed to help developers write, debug, and maintain code with improved quality and speed.

1.1 Code Generation and Assistance

AI-powered code assistants, such as GitHub Copilot, Amazon CodeWhisperer, and Tabnine, are revolutionizing the software development process. These tools utilize deep learning models, particularly based on the Transformer architecture (Vaswani et al., 2017) [1], to generate code snippets, suggest code completions, and help developers navigate through complex codebases. By analyzing vast amounts of publicly available code, these assistants predict the most likely code a developer might need next, allowing developers to focus on higher-level design and problem-solving, rather than repetitive coding tasks.

1.2 Code Refactoring and Optimization

AI tools also contribute to code improvement by suggesting optimizations. These tools analyze existing code to *id2 in S*oftware development.Artificial intelligence (AI) algorithm changes software development by increasing accuracy, efficiency and automation in main functions such as building, dash location and code processing. AI tools are used to help developers write, troubleshoot and maintain better quality and speed code.

© 2025 IJIRCCE | Volume 13, Issue 4, April 2025 |DOI:10.15680/IJIRCCE.2025.1304277

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

2. Code Production and Assistance

AI-operated code assistants, such as Github Copilot, Amazon Codewehispers and Tabnine, bring revolution in the software development process. These devices use deep learning models, especially to help generate on transformer architecture (Vaswani et al., 2017) [1], to generate encodes, suggest code perfection and help developers to navigate through complex code base. By analyzing the huge amount of publicly available codes, this accessory is most likely to predict the code and requires a developer, so developers can focus on high -level design and problem constitution instead of repeated coding tasks.

2.1 Code Refact and adaptation

AI equipment also contributes to the improvement of code by suggesting optimization. These devices analyze existing codes to identify areas of refraction, such as eliminating fruitless codes, improving performance and simplifying complex arguments. The AI-operated system helps to improve code base maintenance, ensure better scalability and reduce technical debt in the software development life cycle [2].

3. AI in software test

The AI test generation increases the software testing by automating the error detection and the result analysis. These innovations helped to identify problems quickly, and ensure that the software is of high quality and the test time is to reduce.

3.1 Test Case Generation

AI can automatically generate test cases based on code analysis and historical test then entify areas for refactoring, such as eliminating redundant code, improving performance, and simplifying complex logic. AI-powered systems help improve the maintainability of codebase, ensuring better scalability and reducing technical debt in the software development lifecycle [2].

3. AI in Software Testing

AI is significantly enhancing software testing by automating test generation, bug detection, and result analysis. These innovations help identify issues early, ensuring that software is of higher quality and reducing testing time.

3.1 Test Case Generation

AI can automatically generate test cases based on code analysis and historical testing data. By learning from previous test scenarios, AI systems can identify the most critical parts of the code to test and ensure comprehensive coverage, including edge cases. This reduces the manual effort involved in creating test cases and improves the testing process's effectiveness [3].

3.2 Bug Detection and Prediction

AI systems can detect and predict bugs by analyzing historical bug data and identifying patterns that are indicative of new issues in the codebase. This predictive capability enables developers to focus on code sections that are likely to introduce new defects, speeding up the debugging process. Furthermore, AI can suggest fixes based on previous solutions, improving troubleshooting efficiency and reducing manual intervention [4].

3.3 Test Result Analysis

AI can automate the analysis of test results, a process that is often time-consuming when working with large test suites. Using machine learning, AI tools can classify and prioritize errors based on their severity, helping developers focus on the most critical issues first. This speeds up the overall debugging and testing cycle [5].

4. AI in Software Maintenance

Software maintenance, which includes updating, modifying, and improving software after its release, benefits greatly from AI. AI is used to automate various maintenance tasks such as fault detection, code refactoring, and predictive maintenance, reducing the manual labour required for software upkeep.

© 2025 IJIRCCE | Volume 13, Issue 4, April 2025 |DOI:10.15680/IJIRCCE.2025.1304277

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

4.1 Predictive Maintenance

AI-based systems use historical performance data to predict when a software component might fail, enabling proactive maintenance. By identifying potential system errors early, maintenance teams can take pre-emptive actions, reducing downtime and improving system reliability. This predictive capability is especially useful in complex systems where manual detection of issues can be time-consuming and error-prone [6].

By checking the previous performance data, AI can predict when software components are likely to cause malfunction. Maintenance teams can take active measures to solve problems using machine learning models so that they can predict potential system errors. Reducing downtime, increasing system dependency and streamlining the maintenance process is made possible by future maintenance.

4.2 Code Refactoring and Enhancement

Refactoring is wanted to decorate code shape, readability, and commonplace universal performance as software systems develop. AI can assist find out code that wishes to be refectories and routinely propose adjustments that might decorate the general brilliant of the device. This improves the overall performance of the safety technique and lessens the paintings required for guide code reviews [7].

5. AI in project management

AI in the same way affects the software control for the software program, where it allows choice, resource allocation and prediction of danger. AI can provide the necessary insight into the company's timeline, viable risk and useful resource requirements by comparing historical missionary facts.

5.1 Risk Prediction and governance

Project data can be analysed using AI-displaced fashion, which will predict delays in the time table, value overturns and expectations related to possible technical problems. To guarantee the meeting of the completion of an assignment, they can help the model challenge managers to create a record keeping option and provide hazard dem Bitation techniques.

5.2 Resource allocation

AI can optimize useful resource allocation, by analysing the performance registries, beyond the requirements for crew membership talent and challenge. The AI program program contributes to timely and pricing of the work, which streamlines the allocation of sources to one-differentiated duties [9].

6. Challenges and limitations

While AI provides great benefits for software technique, many challenges must be solved:

6.1 Data quality and availability: For the AI model proper convenience, they require incredible, well -strange facts. Incorrect forecasts and minimal perfect pointers can eliminate the result from negative numbers.

6.2 Complexity of AI model: Many AI models, especially deep learning models, are calculated expensive and require important resources for training and implementation. These models are also complex, making them difficult to interpret and distribute them.

6.3 Integration with inheritance systems: AI equipment can withstand difficulty integrating with old systems that were not created and remembered AI. There may be questions about compatibility, especially when an attempt is made to include AI in the inheritance software infrastructure.

6.4 Ethical and security concerns: To use AI in software technique raises moral issues of openness, responsibility and justice. In addition, the AI systems are unsafe for unwanted attacks, which can put the safety of the software at risk [10].Additionally; AI systems are vulnerable to adversarial attacks, which could jeopardize the security of the software [10].

© 2025 IJIRCCE | Volume 13, Issue 4, April 2025 |DOI:10.15680/IJIRCCE.2025.1304277

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

III. PSEUDO CODE

STEP1. INITIALISE SOFTWARE PROJECT ENVIRONMENT (CODBASE, REQUIREMENTS, ETC.)
STEP 2. CREATE PRELIMINARY SOFTWARE DESIGN AND ARCHITECTURE USING AI MODEL
(EG GANS FOR DEEP LEARNING, CODE)
STEP 3. ACCESS CODE GENERATION BASED ON REQUIREMENTS USING AI
(EG LANGUAGE MODEL, LEARNING DEEP REINFORCEMENT)
STEP 4. TEST THE CODE GENERATED FOR PERFORMANCE, FUNCTIONALITY AND SECURITY WEAKNESSES:
USE AI-POWERED TEST GENERATION TECHNIQUES TO CREATE DIFFERENT TEST CASES
(EG FADES TESTS, MODEL-BASED TESTS)
B. RUN TESTS AND ANALYZE RESULTS USING AI-BASED TOOLS
(EG NERVE NETWORK TO IDENTIFY ERRORS)
STEP 5. FIND AND FIX THE ERRORS:
USE AI-BASED BUG DETECTION TECHNIQUES
(EG STATIC ANALYSIS, DEEP EDUCATION FOR BUG PREDICTION)
B. CREATE A BUG-FITTY PATCH USING AI-OPERATED TECHNIQUES AUTOMATICALLY
(EG PROGRAM SYNTHESIS, NERVE CODING SEARCH)
STEP 6. ADAPTATION CODE:
USE AI MODELS TO DETECT DISABILITIES IN THE CODE AND SUGGEST ADAPTATION
(EG LEARN DEEP REINFORCEMENT FOR PERFORMANCE SETTING)
B. AI-REFACTORY CODE USING AI-OPERATED REFINE TOOLS
STEP 7. RUN TEST AGAIN TO CONFIRM FAULT FIXATION AND CUSTOMIZATION:
USE CONTINUOUS TESTING USING AI TO ENSURE THAT FUNCTIONAL, SAFETY AND PERFORMANCE STANDARDS.
STEP 8. DOCUMENTS AND REPORTS REPORTS:
USE AI TO AUTOMATICALLY GENERATE DOCUMENTATION BASED ON CODE BASE
(EG COMMENT GENERATION, DOCUMENTATION SYNTHESIS)
B. CREATE SOFTWARE MAINTENANCE PLANS USING AI FOR LONG -TERM PROJECT HEALTH
(EG AUTOMATED CHANGELOGS, CODEBASE HEALTH MONITORING)
STEP 9. IMPROVE SOFTWARE WITH AI -REQUESTSLETS (LEARNING FROM NEW PROBLEMS, FACILITIES OR REQUIREMENTS).
MONITOR THE SOFTWARE PERFORMANCE AND COLLECT FEEDBACK FROM
B. USE AI TO SUGGEST N

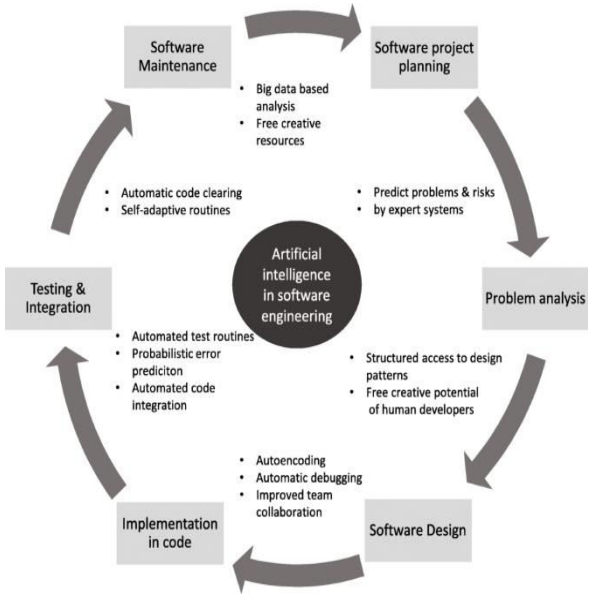


Fig.1Ai in Software Engineering

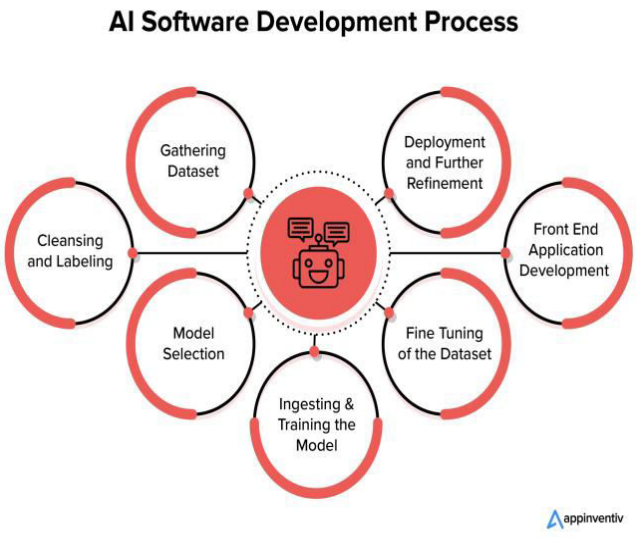


Fig.2Ai in Software Development Process

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

IV. CONCLUSION AND FUTURE WORK

Through automation, better sample constructing and multiplied software program software change the incredible, artificial intelligence software technique. Software becomes greater powerful and green due to its position in technical allocation manage, trying out, protection and software program development. The use of AI in software technique appears to be a sparkling destiny, whether or not it is horrible records, version complexity and integration with inheritance structures irrespective of troubles. Programs are anticipated for the AI era to develop as they growth the existence cycle of software improvement.

REFERENCES

1. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., Kaiser, Ł., & Polosukhin, I. (2017). "Attention is all you need." *Advances in Neural Information Processing Systems, 30*.
2. Gall, H., Jazayeri, M., & Wernick, P. (2018). "Automated Code Refactoring Using AI Techniques." *Journal of Software Maintenance and Evolution: Research and Practice, 30*(4), e2047.
3. Chen, X., Zhang, X., & Wei, Y. (2021). "AI-driven Test Case Generation for Software Reliability." *Journal of Software Testing, Verification & Reliability, 31*(1), e2125.
4. Kotsiantis, S. B., Zaharakis, I., & Pintelas, P. (2016). "Machine Learning Techniques for Bug Prediction and Debugging." *International Journal of Software Engineering & Applications, 10*(2), 31-48.
5. Bener, A., Arslan, E., & Demirörs, O. (2009). "Test Result Analysis using Artificial Intelligence." *Software Testing, Verification & Reliability, 19*(3), 177-198.
6. Kim, H., Park, S., & Lee, C. (2020). "Predictive Maintenance in Software Systems using Machine Learning." *IEEE Access, 8*, 207287-207295.
7. He, J., Liu, X., & Zhang, T. (2020). "AI for Refactoring Legacy Software Systems." *Journal of Software Maintenance and Evolution, 32*(1), e2123.
8. De Moura, F. R., Batista, T., & Alves, D. (2020). "AI in Predicting Software Project Risks." *International Journal of Project Management, 38*(2), 78-89.
9. Elish, M. C., McKinley, A., & Barnes, A. (2018). "Resource Allocation in Software Projects Using AI." *Proceedings of the 2018 IEEE International Conference on Software Engineering*, 370-379.
10. He, J., Liu, X., & Zhang, T. (2020). "Challenges of AI Integration in Legacy Software Systems." *Journal of Systems and Software, 158*, 110431.
11. https://aiperspectives.springeropen.com/articles/10.1186/s42467-020-00005-4
12. https://appinventiv.com/blog/how-to-build-ai-software/

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

INNO SPACE SJIF Scientific Journal Impact Factor

doi crossref

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

9940 572 462 6381 907 438 ijircce@gmail.com

www.ijircce.com

Scan to save the contact details