International Journal of Emerging Research in Engineering and Technology
Pearl Blue Research Group| Volume 6 Issue 4 PP 72-79, 2025
ISSN: 3050-922X | https://doi.org/10.63282/3050-922X.IJERET-V6I4P109

Original Article

The Role of Artificial Intelligence in Software Engineering: A Review of Frameworks, and Impact on the Software Development Life Cycle

Satyadhar Kumar Chintagunta
Independent Researcher.

Received On: 30/08/2025 ***Revised On: 12/09/2025*** ***Accepted On: 05/10/2025*** ***Published On: 24/10/2025***

Abstract - *Computer programming Artificial intelligence is now the key that makes it possible to redesign, develop, and maintain software systems. The automation of software development life cycle (SDLC), predictive modelling, and intelligent decision assistance are all products of AI implementation in the SDLC, which has led to improved productivity and quality of products, in turn. The paper is a comprehensive evaluation of AI models and the reasons why they are relevant to the development of software engineering practice. General-purpose frameworks such as TensorFlow, PyTorch and Keras are very helpful in the creation of larger models, domain-specific models such as Code BERT, GPT-based tools and Auto ML platforms can perform particularly significant tasks, such as code generation, defect detection and automated testing. The paper explores how the different AI methodologies have been incorporated in the SDLC considering aspects like requirements engineering, system design, implementation, testing, deployment, and maintenance. Furthermore, the trend shifts to explainable AI, intelligent maintenance, and self-managed systems have new tendencies, and it is a broader shift towards sustainable and responsible usage of AI in software engineering. Overall, the given paper sums up the most significant progress and achievements, and it can be of great value to the researchers, practitioners, and other interested parties who want to apply AI to create a new layer in the software.*

Keywords - *Artificial Intelligence, Software Engineering, AI Frameworks, Software Development Life Cycle, Machine Learning, Deep Learning.*

1. Introduction

Software engineering is not an exception and the software development life cycle (SDLC) is comprised of several steps. These processes include requirements gathering, design, development, testing, deployment and maintenance. All of the above steps such as automated requirement analysis, intelligent code generators, predictive defect detectors, and self-healing systems are optimized with artificial intelligence (AI) practices such as deep learning, evolutionary algorithms, machine learning, and natural language processing [1]. With the application of AI, SE practices are being transferred towards high level of automation, flexibility, and quality control.

AI has turned into a truly disruptive trend within several industries and it has transformed the manner in which it formulates, develops and maintains systems [2]. In software engineering (SE) specifically, but they are seeing the radicalization of AI in accordance with the increasing demands of intelligent automation, predictive analytics and adaptive decision making [3]. The simplicity of the modern software systems and the growing demands to make them more efficient and reliable make us consider how AI can revolutionize the old-fashioned methods of software development.

Availability of efficient AI frameworks is among the most important facilitators of this change. Scalable AI model building is based on general-purpose frameworks such as TensorFlow, PyTorch, and Keras [4][5]. Software-specific activities, such as bug detection, code completion, and test-case development are only a few examples that can be achieved with the help of domain-specific frameworks. Those frameworks include Code BERT (for the programming language of the dataset), tools based on GPT (Generative Pre-trained Transformer), and Auto ML platforms. Understanding the strengths, weaknesses, and ways to integrate these frameworks is central to researchers and practitioners employing AI in SE projects.

Software engineering's use of AI remains in its infancy, despite promising developments in the field. While many studies have examined various aspects of the SDLC on its own or applicable AI technologies in isolation, there is not a composite view of frameworks and their implications across phases in the SDLC [6]. Understanding nuances between frameworks, approaches to integrating, and trends is important for recognizing gaps that are important to research, when using or deploying tools, or aligning progress in AI with discipline-specific needs. The function of AI in software engineering, with a focus on frameworks and the effects they have on the SDLC [7]. The author examines the effects of AI frameworks on each stage of the development life cycle, delves into their incorporation into software engineering (SE) practice, and thinks about both general-purpose and software-specific AI frameworks [8]. The author identifies challenges, comparative advantages, and emerging trends, providing guidance to researchers, practitioners, and stakeholders

working to move the needle on AI-enabled software engineering.

1.1. Structure of the Paper

The structure of the paper is as follows: Software engineering AI is discussed in Section II, AI frameworks and trends are examined in Section III, the influence of AI throughout the SDLC is discussed in Section IV, the literature review is presented in Section V, and future work is discussed in Section VI.

2. Artificial Intelligence in Software Engineering

Software engineering is booming in comparison to other areas of engineering due to its focus on solving the complex problems that have arisen as a result of recent technology developments. To keep up with new challenges, software engineering must evolve. For instance, green SE and social SE have developed into distinct domains due to the expansion of the IoT [9]. The amount of carbon dioxide equivalents emitted by data centers is comparable to that of traditional greenhouses. Engineers also need to think about how the proliferation of cellphones is causing power usage to grow. There is an increasing responsibility on software engineers to plan for the future of technological systems, such as cloud computing, due to the increasing significance of software (Figure 1). The bigger systems that engineer work with are dependent on and used by a variety of sociotechnical settings, therefore it's important for them to keep that in mind[10]. The technical understanding of a software engineer is just half the story. They also need to possess soft skills, which enable them to control, affect, bargain, and collaborate with other members of the team.

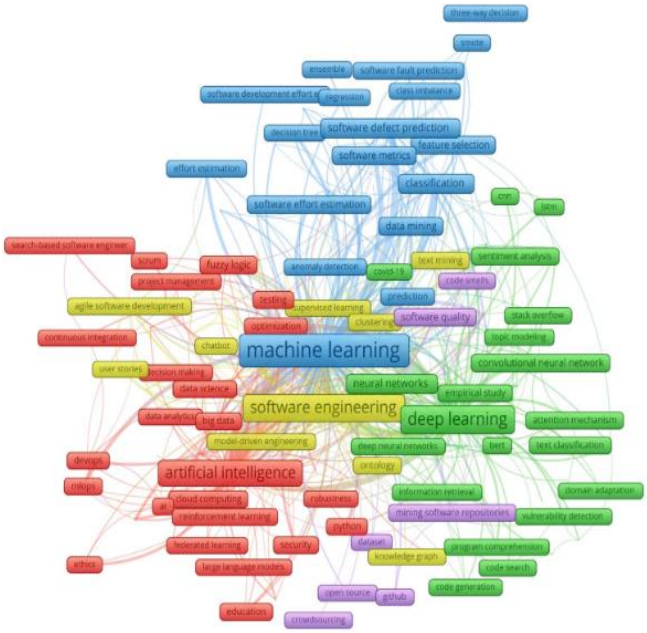

Fig 1: AI Use in Software Engineering

2.1. AI Applications in Software Engineering

The introduction of AI has caused revolutionary changes in several aspects of software development [11]. Improved software engineering processes have resulted from the use of AI-driven tools and approaches. In this part, we'll take a look at the most important ways AI is changing software engineering, namely how it's improving efficiency and transforming traditional methods. Selection Tools and datasets must meet these criteria: The following criteria were used to choose the datasets, examples, and tools that were included in this study:

- **Relevance to Industry Applications:** A focus was placed on datasets and techniques that see extensive usage in practical software engineering settings, such IBM's defect prediction tools and GitHub's Copilot.
- **Recency:** Preference was given to those tools and datasets that have been published or actively used in the past five years. This was in order to ensure that the study is representative of the field of status.
- **Accessibility:** Consistent results might be guaranteed by utilizing open-source datasets and technologies accompanied by clearly published documentation.
- **Coverage of Development Phases:** Software development has numerous stages, coding, testing, and maintenance are just some to be mentioned in accordance with the chosen examples, which attempt to demonstrate them.
- **Impact and Adoption:** Data and tools that have been proven to be effective in academic and industrial research received priority during selection.

2.2. Taxonomy of AI Techniques

AI methods are the blocks of the intelligent systems, which are the principles of the development of them [12]. Following these approaches as the guidelines, it is possible to learn new things and address the problems more effectively[13]. The salient AI methods that can be used with software engineering are as follows in Figure 2:

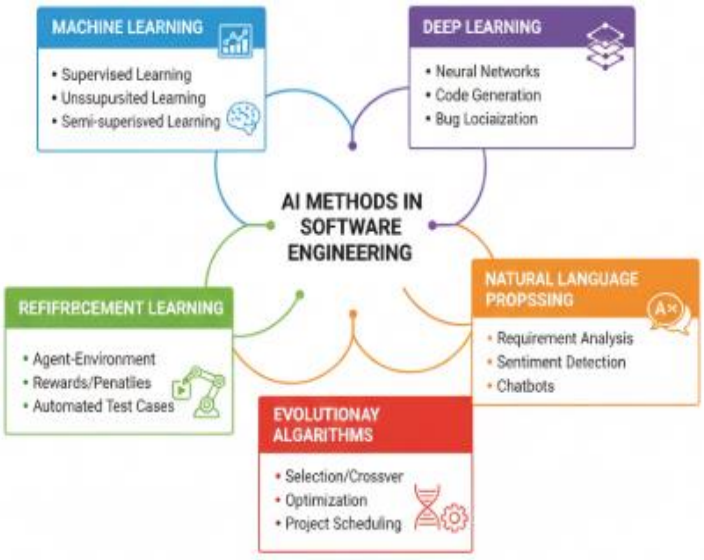

Fig 2: AI Methods in Software Engineering

- **Machine Learning (ML):** ML enables one to study using data and maximize performance without programmed learning. It has algorithms that identify patterns and are applicable in predictive modeling[14][15]. There are three main branches of ML: supervised learning, unsupervised learning, and semi-supervised learning. ML is useful in many different domains, including project analytics, effort estimation, and defect prediction.
- **Deep Learning (DL):** DL is a subfield of ML that deals with high-dimensional, unstructured data using ANN with several layers. It has proved to be successful in software engineering activities including code generation, bug localization and software documentation using natural languages.
- **Reinforcement Learning (RL):** The RL models are learning that takes place between an agent and the environment, driven by rewards and penalties. It works especially well in the situations where it is necessary to make decisions in a sequence, including automated test case generation, self-healing systems, and adaptive resource allocation.
- **Natural Language Processing (NLP):** NLP enables computers to learn and behave like their human counterparts. Software engineering Applications NLP generates requirement analysis assistance, sentiment detection in communication between software developers, chatbots and automatic documentation.
- **Evolutionary Algorithms (EA):** EA is guided by evolution in the biological world and it applies selection, crossover and mutation to search the best solutions in the iterative fashion. These algorithms have many uses in software project scheduling, test data generation and architecture optimization.

2.3. AI-Enhanced Tools for Automated Testing

Software testing has also brought advanced tools in AI usage that are more precise and effective in test generation, execution and validation. The tools are developed on the principles of machine learning and smart algorithms to detect defects, adaptability to alteration, and enhanced dependability of tests.

- **Selenium with AI Enhancements:** The development of test scripts, their execution, and maintenance can be significantly improved when Selenium is enhanced with AI abilities [16]. Machine learning facilitates easier construction of dependable test cases and prediction of possible problem areas through the application of pattern recognition algorithms that have been trained on test data.
- **Test AI:** This AI-powered tool streamlines testing by automatically detecting UI components, generating tests on the fly, and executing them in a variety of settings and on different devices. Adapting to application modifications and optimizing the testing suite, its machine learning models improve over time.
- **Applitools:** Visual testing is this platform's forte; it employs AI to spot UI inconsistencies and flaws. It does this by automatically comparing the app's current state to a baseline image, which makes visual regression testing more accurate.
- **Rainforest QA:** This technology integrates AI capabilities with human testers. It employs machine learning to simplify the process of creating and running test cases, enabling teams to increase the volume of their testing without sacrificing quality.

Bioenergy refers to electricity and gas that is generated from organic matter, known as biomass. This can be anything from plant and timber to agriculture and food waste and even sewage. Bioenergy includes the production of fuel from

organic matter as well. Energy from biomass can be used for electricity, heating, and transportation, and can be replenished anywhere. Around seventy-five percent of the world's renewable nergy is composed of biomass energy due to its potential and wide use [7]. Also, it is carbon-neutral, meaning that it adds no net carbon dioxide to the atmosphere. In addition, it reduces the level of trash in the ground by as much as 90 percent by burning solid waste. Biomass fuels, on the other hand, are not completely clean and can also cause deforestation. They are also less efficient than fossil fuels. But proper management and planning of its disadvantages will improve its potential.Bioenergy refers to electricity and gas that is generated from organic matter, known as biomass. This can be anything from plant and timber to agriculture and food waste and even sewage. Bioenergy includes the production of fuel from organic matter as well. Energy from biomass can be used for electricity, heating, and transportation, and can be replenished anywhere. Around seventy-five percent of the world's renewable energy is composed of biomass energy due to its potential and wide use [7]. Also, it is carbon-neutral, meaning that it adds no net carbon dioxide to the atmosphere. In addition, it reduces the level of trash in the ground by as much as 90 percent by burning solid waste. Biomass fuels, on the other hand, are not completely clean and can also cause deforestation. They are also less efficient than fossil fuels. But proper management and planning of its disadvantages will improve its potential.

3. Frameworks for AI in Software Engineering

AI frameworks for software engineering reach beyond productivity to take into account ethical, legal, and operational aspects of the SDLC [17]. They identify possible pitfalls, such as bias, fairness, transparency, privacy, and human-centered design, and propose structured solutions or guidelines for reducing ethical and legal risks in the software development process. Regarding SE, ethical AI frameworks provide salient concepts to support requirement elicitation, system design, and testing, assuring that AI-enhanced tools and applications produced are efficacious and also responsible and trustworthy [18]. These frameworks are essential in embedding principles of transparency, data privacy, and accountability as foundational guiding principles in SE practices, and demonstrate practical ways for integrating AI in projects while preserving ethical and legal compliance.

3.1. Ethical AI Frameworks

The objective of numerous ethical AI frameworks is to anticipate potential ethical issues and provide solutions or methods to mitigate the risks associated with them [19]. Potential ethical principles and concerns, as well as laws or solutions to address them, may be outlined in such frameworks, which can also serve as a starting point for discussions on the effects of AI systems on society. The second kind includes cautions about possible dangers or suggestions for improving AI system design. Typically, the conceptual part is devoted to elucidating ideas that form the basis of ethical norms, such as fairness and prejudice in artificial intelligence systems. Ideal characteristics of an AI system, such as openness, privacy, and respect for human dignity in its use, are frequently articulated as principles. Principles and notions are almost interchangeable.

3.2. Integration Testing in Software Projects

Testing modules in pairs is the most effective and dependable approach to do integration testing in a big system. A big system's interfaces may not be tested all at once due to the complexity involved and the likelihood of inaccurate results [20]. After the first set of pairs has been evaluated, the system is considered partly integrated. An incremental version can be used to incorporate more modules in order to obtain a valid test. The SDLC begins with the design phase, when a test plan is created and documented[21]. That plan is then used throughout integration testing. The objective of integration testing is to ensure that the system satisfies the functional need, the performance requirement (the degree to which the system's sections interact with one another), and the reliability requirement (the degree to which the system produces the desired result for the client). Choose the correct SDLC model throughout the design process. Visual illustration of software integration testing in relation to modules testing is shown in Figure 3.

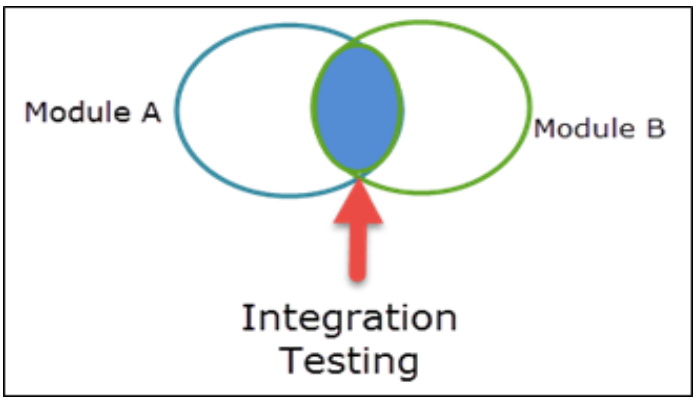

Fig 3: Software Integration Testing

3.3. Emerging Trends in AI Frameworks

AI-driven software engineering holds exciting possibilities, with trends pointing towards self-managing systems and intelligent maintenance [22]:

- **Self-Managing Systems:** AI can enable the development of self-managing systems that autonomously monitor and optimize their performance. Less human interaction is required since these technologies can identify and fix problems instantly. Self-managing systems are able to adjust to the changing conditions and workloads resulting in optimum performance and reliability.
- **Intelligent Maintenance:** The AI-based maintenance systems have the potential to forecast possible problems and recommend remedial responses to address them, preventing their effects on the software. It comes with predictive maintenance, whereby AI models are used to forecast failures in the future by analyzing past data, and hence planning to maintain the equipment. Smart maintenance can significantly reduce the downtimes, and increase the overall stability of software systems.

4. Impact of AI across the Software Development Life Cycle (SDLC)

AI has transformed several processes of software development life cycle (SDLC), which offers both automation and prediction, and offers adaptation to learn in an adaptive manner, which help in enhancing efficiency and quality [23]. The following phases may be summarized as its effects.

1. Implementation (Coding): GitHub Copilot is an AI-generated code generation system and Code BERT and GPT-based models can provide developers with auto-completion, bug detection, and refactoring proposals. These aids in speeding up the coding process, enhance consistency as well as assist beginner developers in overcoming knowledge gaps.
2. Testing and Verification: Automated test case development, intelligent regression testing, and defect prediction are three areas where AI makes a substantial contribution. ML models that have been trained on defect data from the past can identify modules that are likely to have faults, and reinforcement learning techniques can improve testing methodologies to ensure greater coverage.

4.1. AI in Software Project Planning

Developers and customers work together to define the project's goals and needs during the planning phase of software development projects. For software projects to be technically effective and economically efficient, scheduling and planning their development is essential. The original goal of search-based software engineering was to find the best possible solution under certain limitations in order to optimize project objectives including time, money, and quality [24]. Classical linear programming was the backbone of early models; today's models integrate non-linearities, uncertainty, numerous decision layers, dynamic contexts, and coefficient interdependencies. Innovative scheduling systems are increasingly being labelled as artificial intelligence due to the rising complexity of decision layers and the reliance on external databases for documentation and past experiences. Project duration and cost seem to be competing goals, and human planners have a hard time finding a middle ground. AI technologies can be helpful in facilitating this procedure.

4.2. Requirements Engineering and Planning

The foundation of every software system is its requirements analysis and planning, the initial step in software engineering. By outlining the primary functions and features of an application, software requirements provide a picture of the program's future. Many academics attempted to include AI into this typically first period of the lifecycle because of its significance [25]. That which a system must have in order to fulfil the terms of a legally binding document, such as a contract, standard, specification, or user-imposed goal or problem-solving need. By outlining the functions, behaviour, features, qualities, and limitations that the system must meet, requirements establish the groundwork for software development processes [26]. The field of software engineering known as requirements engineering (RE) focusses on gathering and documenting user needs and requirements for software systems. To guarantee that the system requirements are comprehensive, consistent, and applicable, RE places an emphasis on the use of methodical and repeatable methodologies.

4.3. System Design and Development

Artificial intelligence (AI) may help with new system design builds by guiding the selection of suitable architectural patterns [27][28]. Based on the system definitions given by technical leaders, it may produce system design from the ground up. In addition, for existing designs – AI can aid in the below-mentioned ways:

- Analyze weaknesses in the designs,
- Suggest alternate design approaches by evaluation of design proposals ensuring uniformity in design
- Provide testing capabilities for performance and scalability eg: load management
- Documentation of system design
- UX (User-experience) mockups

By creating a blueprint for the software (front-end UX mockups, system architecture, data structure and algorithms inclusive), AI can assist in making informed decisions based on best practices

Development (used interchangeably with ‘Build’ or ‘Engineering’) is the heart of the SDLC and most respected tech companies are developer or engineer driven. This is the phase that is most vulnerable to change, among SDLC phases. The system design can be converted into code within a little amount of time with the help of AI which provides code suggestions, syntax check, algorithm optimization, and code formatting and debugging. Code optimization can be motivated by generative AI which enhances the performance and resource use by ensuring that memory is efficiently used and that data structures are properly organized.

5. Literature of Review

This literature Summary compiles the emerging uses of AI in the field of software engineering, including requirements prioritization, life cycle engineering, SDLC integration and domain specific frameworks, at the same time pointing out some unresolved issues related to benchmarks, data quality, ethical considerations and international cooperation in the field, as future research directions.

Anwar and Bashir (2023) a fresh angle by conducting a parametric study of needs prioritization techniques based on AI; these parameters were established after a thorough review of the relevant literature. The chosen parameters span the gamut from general (related to prioritization) to specific (AI-related) to generic once more. This work has been extremely useful in classifying AI-based approaches and identifying their optimal use cases. As a consequence of research, stakeholders such as researchers and requirement analyzers are better able to choose the most effective needs prioritization approach [29].

Rahman et al. (2022) Several AI-LCE papers address the Sustainable Development Goals, with a focus on responsible consumption and production, sustainable cities and communities, and industry, innovation, and infrastructure. These articles, when read collectively, provide an overview of the many AI techniques used by LCE. Production design and maintenance and repair have received the lion's share of LCE research, whereas logistics and procurement have received comparatively little attention. Countries with substantial research budgets and an emphasis on Industry 4.0 tend to have disproportionately high concentrations of AI-LCE researchers [30].

Shafiq et al. (2021) A Framework Review of AI's Role in Software Engineering and Its Impact on the SDLC. In order to make existing software into self-improving systems, software engineers are rapidly using machine learning. The software engineering community is always coming up with new ways to incorporate machine learning into the software development life cycle, but there are still many areas that might need some improvement [31].

Martínez-Fernández et al. (2021) AI-powered systems prioritize reliability and security. Multiple SE techniques for AI-based systems may be identified and categorized according to the SWEBOK categories. Though research on software quality and testing is abundant, software maintenance studies are noticeably under-researched. Among the most common difficulties are those touching on data. Researchers can quickly understand the present state of the art and identify areas that need more investigation; practitioners can know the methodologies and problems of SE for AI-based systems; and educators may utilize findings to assist students realize the connection between SE and AI in the classroom [32].

Moreb et al. (2020) determine the user's needs, the system's connected objects' functions, and the dataset's necessary machine learning techniques. The data used for this study really covers the last three years and came from a Palestinian government-run hospital. The seven-step SEMLHI methodology consists of the following: software application release; information structure; security and privacy assurance; performance testing and evaluation and workflow design, implementation, maintenance, and definition [33].

Wallure and Naaz (2019) AI in software engineering: its background, methods, and real-world uses. Software testing, software development lifecycles with AI assistance, and AI-driven project management are all covered. Also, have a look at how AI is being used in SaMD, software measurement, and software engineering in general. Better efficiency, precision, and decision-making are just a few of the many advantages that may be gained from combining AI with software engineering. Nevertheless, there are obstacles to overcome, including issues with data quality, the interpretability of models, and ethical considerations [34].

Table I consolidates significant studies on AI in software engineering, highlighting diverse approaches, findings, challenges, and proposed directions, thereby offering a comprehensive overview of advancements and research gaps in this domain.

Table 1: Summary of a Study on the Role of Artificial Intelligence in Software Engineering

Author	Study On	Approach	Key Findings	Challenges	Future Directions
Anwar & Bashir (2023)	A statistical evaluation of AI-powered needs prioritization	Literature-based parametric analysis of techniques	Identified generic and AI-specific parameters to classify techniques; assists stakeholders in selecting optimal methods	Lack of standard benchmarks; diversity of AI techniques complicates comparisons	Develop unified evaluation metrics; extend prioritization methods to dynamic environments
Rahman et al. (2022)	AI in Life Cycle Engineering (LCE) addressing SDGs	Systematic review of AI-LCE studies	AI is mainly applied in production design and maintenance; logistics and procurement underexplored; research concentrated in few countries	Limited coverage of subfields; uneven global participation	Expand AI research to less explored LCE areas; encourage global collaboration
Shafiq et al. (2021)	Role of AI in SE: frameworks	Literature review	Machine learning increasingly adopted	Limited understanding of	Explore novel ML applications in

	and SDLC impact		across SDLC; potential for self-learning systems	ML's impact across all SDLC phases	underexplored SDLC phases
Martínez-Fernández et al. (2021)	methods for SE systems that utilize AI	Survey that is organized according to SWEBOK regions	High prevalence in testing & quality; maintenance underexplored; recurring data-related issues	Data quality, safety, and dependability concerns	Strengthen research in maintenance and data management; integrate AI-SE in curricula
Moreb et al. (2020)	SEMLHI methodology for healthcare software development	Case study using hospital dataset; ML integration in workflows	Proposed 7-phase methodology (design to release); addressed privacy, security, testing	Handling sensitive data; ensuring privacy/security	Extend SEMLHI to broader domains; automate phases with AI
Allured & Naaz (2019)	Historical context & applications of AI in SE	Exploratory review of methodologies and applications	AI enhances project management, SDLC automation, and SaMD; boosts efficiency and accuracy	Ethical issues, interpretability, integration challenges	Investigate scalable AI integration in SE; improve explainability and trust

6. Conclusion and Future Work

In recent years, AI has become a game-changer in software engineering, impacting the SDLC at every level. Integration of AI methods into requirements engineering, system design, coding, testing, deployment, and maintenance is highlighted in this study. AI techniques include DL, evolutionary algorithms, ML, NLP, and reinforcement learning. General-purpose frameworks like TensorFlow and PyTorch provide the computational foundation, while domain-specific frameworks such as Code BERT and GPT-based tools support software-centric tasks, including defect prediction, code generation, and test optimization. Ethical frameworks further ensure that principles of fairness, transparency, and accountability are embedded into SE practice. Collectively, these developments indicate a clear trend toward higher automation, improved adaptability, and enhanced quality. However, challenges remain in data quality, integration complexity, cost-effectiveness, and explainability, which require careful consideration. Moreover, research is still limited to studies published up to 2023 and practical industrial adoption remains underexplored.

Future work should focus on developing unified AI frameworks tailored for software engineering, with stronger emphasis on explainability and integration into Agile and DevOps environments. Expanding empirical studies, establishing benchmarks, and addressing ethical challenges can be essential to ensure sustainable, trustworthy, and industry-ready applications of AI in SE.

References

[1] A. Mishra and D. Dubey, "A Comparative Study of Different Software Development Life Cycle Models in Different Scenarios," *Int. J. Adv. Res. Comput. Sci. Manag. Stud.*, vol. 1, no. 5, pp. 2321–7782, 2013.

[2] C. Collins, D. Dennehy, K. Conboy, and P. Mikalef, "Artificial intelligence in information systems research: A systematic literature review and research agenda," *Int. J. Inf. Manage.*, vol. 60, no. July, p. 102383, 2021, doi: 10.1016/j.ijinfomgt.2021.102383.

[3] A. Goyal, "Driving Continuous Improvement in Engineering Projects with AI-Enhanced Agile Testing and Machine Learning," *Int. J. Adv. Res. Sci. Commun. Technol.*, vol. 3, no. 3, pp. 1320–1331, Jul. 2023, doi: 10.48175/IJARSCT-14000T.

[4] A. Paszke *et al.*, "PyTorch: An imperative style, high-performance deep learning library," *Adv. Neural Inf. Process. Syst.*, vol. 32, no. December, 2019, doi: 10.48550/arXiv.1912.01703.

[5] A. R. Duggasani, "The Integration of Java-Based Applications in Software Development and AI: A Conceptual Review," *Int. J. Adv. Res. Sci. Commun. Technol.*, vol. 5, no. 1, p. 11, 2025.

[6] I. Jada and T. O. Mayayise, "The impact of artificial intelligence on organisational cyber security: An outcome of a systematic literature review," *Data Inf. Manag.*, vol. 8, no. 2, p. 100063, Jun. 2023, doi: 10.1016/j.dim.2023.100063.

[7] V. S. Thokala, "Utilizing Docker Containers for Reproducible Builds and Scalable Web Application Deployments," *Int. J. Curr. Eng. Technol.*, vol. 11, no. 6, pp. 661–668, 2021, doi: 10.14741/ijcet/v.11.6.10.

[8] H. H. Ammar, W. Abdelmoez, and M. S. Hamdi, "Software Engineering Using Artificial Intelligence Techniques : Current State and Open Problems," *Softw. Eng. Using Artif. Intell. Tech. Curr. State Open Probl.*, no. October 2014, pp. 24–29, 2012.

[9] I. V. D. Srihith, R. Varaprasad, Y. R. Mohan, T. A. S. Srinivas, and Y. Sravanthi, "A Review on New Challenges in AI and Software Engineering," *Int. J. Adv. Res. Sci. Commun. Technol.*, vol. 2, no. 1, pp. 34–42, Oct. 2022, doi: 10.48175/IJARSCT-7137.

[10] S. K. Chintagunta, "Generative AI Approaches to Automated Unit Test Case Generation in Large-Scale Software Projects," *ESP J. Eng. Technol. Adv.*, vol. 4, no. 1, pp. 150–157, 2024, doi: 10.56472/25832646/JETA-V4I1P121.

[11] D. Gonzalez, T. Zimmermann, and N. Nagappan, "The State of the ML-universe," in *Proceedings of the 17th International Conference on Mining Software Repositories*, New York, NY, USA: ACM, Jun. 2020, pp. 431–442. doi: 10.1145/3379597.3387473.

[12] M. Elahi, S. O. Afolaranmi, J. L. Martinez Lastra, and J. A. Perez Garcia, *A comprehensive literature review of the applications of AI techniques through the lifecycle of industrial equipment*, vol. 3, no. 1. Springer International Publishing, 2023. doi: 10.1007/s44163-023-00089-x.

[13] P. Gupta, S. Kashiramka, and S. Barman, "A Practical Guide for Ethical AI Product Development," in *2024 IEEE 11th Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering (UPCON)*, IEEE, Nov. 2024, pp. 1–6. doi: 10.1109/UPCON62832.2024.10983504.

[14] N. Prajapati, "The Role of Machine Learning in Big Data Analytics: Tools, Techniques, and Applications," *ESP J. Eng. Technol. Adv.*, vol. 5, no. 2, 2025, doi: 10.56472/25832646/JETA-V5I2P103.

[15] R. Q. Majumder, "Machine Learning for Predictive Analytics: Trends and Future Directions," *SSRN Electron. J.*, 2025, doi: 10.2139/ssrn.5267273.

[16] Y. K. Dwivedi, A. Sharma, N. P. Rana, M. Giannakis, P. Goel, and V. Dutot, "Evolution of artificial intelligence research in Technological Forecasting and Social Change: Research topics, trends, and future directions," *Technol. Forecast. Soc. Change*, 2023, doi: 10.1016/j.techfore.2023.122579.

[17] E. Mnkandla, "About software engineering frameworks and methodologies," in *AFRICON 2009*, IEEE, Sep. 2009, pp. 1–5. doi: 10.1109/AFRCON.2009.5308117.

[18] G. Modalavalasa, "The Role of DevOps in Streamlining Software Delivery: Key Practices for Seamless CI/CD," *Int. J. Adv. Res. Sci. Commun. Technol.*, vol. 1, no. 12, pp. 258–267, Jan. 2021, doi: 10.48175/IJARSCT-8978C.

[19] E. Prem, "From ethical AI frameworks to tools: a review of approaches," *AI Ethics*, vol. 3, no. 3, pp. 699–716, Aug. 2023, doi: 10.1007/s43681-023-00258-9.

[20] J. E. T. Akinsola, M. Adeagbo, S. O. Abdul-Yakeen, F. O. Onipede, and A. A. Yusuf, "Qualitative Comparative Analysis of Software Integration Testing Techniques," *J. Sci. Logics ICT Res.*, vol. 7, no. 2, 2022.

[21] V. Prajapati, "Advances in Software Development Life Cycle Models: Trends and Innovations for Modern Applications," *J. Glob. Res. Electron. Commun.*, vol. 1, no. 4, 2025.

[22] P. G. Todorov, "The Application of Artificial Intelligence in Software Engineering," *Int. J. Adv. Multidiscip. Res. Stud.*, vol. 2, no. 5, pp. 835–842, 2022, doi: 10.2139/ssrn.4943407.

[23] A. Saravanos and M. X. Curinga, "Simulating the Software Development Lifecycle: The Waterfall Model," *Appl. Syst. Innov.*, vol. 6, no. 6, 2023, doi: 10.3390/asi6060108.

[24] M. Barenkamp, J. Rebstadt, and O. Thomas, "Applications of AI in Classical Software Engineering," *AI Perspect.*, vol. 2, no. 1, Dec. 2020, doi: 10.1186/s42467-020-00005-4.

[25] F. A. Batarseh, R. Mohod, A. Kumar, and J. Bui, "The application of artificial intelligence in software engineering," in *Data Democracy*, Elsevier, 2020, pp. 179–232. doi: 10.1016/B978-0-12-818366-3.00010-1.

[26] V. S. Thokala, "A Comparative Study of Data Integrity and Redundancy in Distributed Databases for Web Applications," *Int. J. Res. Anal. Rev.*, vol. 8, no. 4, pp. 383–389, 2021.

[27] A. S. Pothukuchi, L. V. Kota, and V. Mallikarjunaradhya, "Impact of Generative AI On the Software Development Life Cycle," *Int. J. Creat. Res. Thoughts*, vol. 11, no. 8, 2023.

[28] H. S. Chandu, "Advancements in Asynchronous FIFO Design: A Review of Recent Innovations and Trends," *Int. J. Res. Anal. Rev.*, vol. 10, no. 02, pp. 573–580, 2023.

[29] R. Anwar and M. B. Bashir, "A Systematic Literature Review of AI-Based Software Requirements Prioritization Techniques," *IEEE Access*, vol. 11, no. October, pp. 143815–143860, 2023, doi: 10.1109/ACCESS.2023.3343252.

[30] H. Rahman, R. S. D'Cruze, M. U. Ahmed, R. Sohlberg, T. Sakao, and P. Funk, "Artificial Intelligence-Based Life Cycle Engineering in Industrial Production: A Systematic Literature Review," *IEEE Access*, vol. 10, no. October, pp. 133001–133015, 2022, doi: 10.1109/ACCESS.2022.3230637.

[31] S. Shafiq, A. Mashkoor, C. Mayr-Dorn, and A. Egyed, "A Literature Review of Using Machine Learning in Software Development Life Cycle Stages," *IEEE Access*, vol. 9, pp. 140896–140920, 2021, doi: 10.1109/ACCESS.2021.3119746.

[32] S. Martínez-Fernández *et al.*, "Software Engineering for AI-Based Systems: A Survey," *ACM Trans. Softw. Eng. Methodol.*, vol. 31, no. 2, pp. 1–59, Apr. 2021, doi: 10.1145/3487043.

[33] M. Moreb, T. A. Mohammed, and O. Bayat, "A novel software engineering approach toward using machine learning for improving the efficiency of health systems," *IEEE Access*, vol. 8, pp. 23169–23178, 2020, doi: 10.1109/ACCESS.2020.2970178.

[34] V. Wallure and S. Naaz, "The Influence of Artificial Intelligence on Modern Software Engineering.," *MIS Rev. Int. JOURNAL, A Peer Rev. Multidiscip. Ref. Res. J.*, vol. 11, no. 3, Mar. 2019, doi: 10.5281/zenodo.14922695.