Application of Artificial Intelligence in Software Development Life Cycle: A Systematic Mapping Study

Shilpi Singh and Saurabh Sambhav

Abstract Building a quality maintainable software product is the necessity of any organization and it depends on various factors. The selection of an appropriate model is one of the factors to achieve this objective. Primarily any Software Development Life cycle (SDLC) model consists of seven phases: requirement gathering, planning, designing, coding, testing, implementation, and maintenance. Each phase has its own importance and will have to optimize those to get an optimal result. And for that, artificial intelligence (AI) offers a lot of potential for improving all aspects of the software development life cycle phases. The revolution of AI in software engineering (SE) is showing an upsurge in research. It becomes evident that a variety of AI paradigms might be used to improve the process and address many of the fundamental issues that the SE sector has been dealing with. Incorporation of AI in SE significantly impacts the developer's productivity and product reliability which leads to achieving quality maintainable software products. This work presents a comprehensive study on AI methods in SDLC (model-wise) and AI methods in SDLC (phase-wise) with the help of different parameters. The author's aim is to provide insights to the researchers and practitioners belonging to the field of SE, who will benefit from this study and get the future direction as well.

Keywords Software engineering · Software development life cycle · Artificial intelligence (AI)

1 Introduction

Software is an essential component to make the human life more comfortable. Specifically, in the present scenario where the computer system is widely used in many fields

S. Singh (✉) · S. Sambhav
Amity University Patna, Patna 801503, India
e-mail: shilpi.singh.it@gmail.com

S. Sambhav
e-mail: ssambhav.iit@gmail.com

D. K. Sharma et al. (eds.), *Micro-Electronics and Telecommunication Engineering*, Lecture Notes in Networks and Systems 617,
https://doi.org/10.1007/978-981-19-9512-5_60

of our life. As per the history of software engineering (SE) and the survey results of software development methodologies, development organizations are highly dependent on the software that makes it more interesting and attractive as well. There are many types of software systems, from simple embedded systems to complex global information systems. Depending on different types of software product, development procedure and related complexities are also varying. Hence, software engineering needs optimized software development life cycle model. Some fundamental questions that are likely to arise in your mind are: How do you define software and software engineering? What exactly is artificial intelligence? What features of software engineering lend themselves to artificial intelligence concepts and techniques? The term software crisis and software engineering was coined in 1986 by NATO. The software engineering discipline incorporates different aspects of software development from requirement gathering to maintenance to delivery.

To develop software products in a disciplined and systematic manner, we need a software development life cycle (SDLC). Without a SDLC model, it is impossible to identify the phase's entry or exit criteria. Without software life cycle models, software project managers will struggle to keep track of the project's development. There are 6 phases in SDLC are: requirement analysis, planning, designing, developing, testing, and maintenance. Alternatively, artificial intelligence, which is part of computer science, seeks to make machines do things humans have traditionally performed. Its goal is to develop machines that can engage in actions that humans see as intelligent. Despite the complexity of AI, the introduction of machine learning and deep learning has led to a paradigm change in almost every area of the software business. In software engineering, AI plays a crucial role in streamlining processes, reducing waste, and automating repetitive manual tasks. A number of research and practice areas have demonstrated the effectiveness of AI techniques in software engineering, including probabilistic engineering, learning and prediction, and search-based engineering. This is true not only of the software systems we create, but also of the methods we use to create them, which are frequently based on estimates. As a result of its use, machine learning develops ways to compute that are more efficient. When AI tools are used successfully, human developers' creative potential is multiplied.

2 Literature

In the literature, the importance of AI on various phases of software development life cycle can be seen. Robert et al. presented AI in software engineering application levels taxonomy, which categorizes different applications based on the nature of AI technology employed and the level of automation allowed [1]. Their work demonstrates the use of presented taxonomy by classifying numbers of research work and various workshop editions. Their work is helpful for businesses in determining how to incorporate AI into their software products and developing AI strategy. Software engineers haven't paid much attention to artificial intelligence (AI), despite its

growing importance in self-organizing IT applications. The current state of AI application to software engineering, prospective future developments, and risks related to it is evaluated by Barenkamp et al. [2]. Several qualitative interviews with various stakeholders who are utilizing or desiring to utilize artificial intelligence tools are paired with a systematic evaluation of prior studies in the subject. Insights from the software development life cycle are also categorized in the study. Automated test tools and agile testing automation would also benefit from improved software development environments. According to Narayan et al. [3], artificial intelligence can be used to improve software quality and the software engineering process. A major focus of their research is improving software quality systems using artificial intelligence and reducing time to market. Al-Ahmad et al. [4] developed a swarm intelligence-based model with the goal is to improve the prediction rate as much as possible in the software development process. After comparison from most of the AI techniques, they found that as compared to distinctive ML classifiers and traditional model, their suggested model consistently outperforms them. Additionally, Ammar et al. [5] evaluated how artificial intelligence approaches can be applied to software engineering. They suggested that enterprises need to closely partner with Microsoft to utilize cloud-first technology and AI solutions improve software system quality overall, shorten time to market, and operational efficiency. It also necessitates a number of distinct stages in order to combine these different sorts of knowledge into a single final result. By considering this, Ebbah et al. [6] examined AI techniques from the point of view of their use in software development process. The technique examines AI techniques that can solve problems in software engineering procedures that have been discovered (or are looming in future). Rech et al. [7] presented and looked at an approach to SE that focuses on artificial intelligence: KBS, AmI, and CI. They review the strong ties between artificial intelligence and SE with emphasis on areas the publications focus on.

Artificial intelligence (AI) and machine learning (ML) approaches have been heavily applied in software engineering to enhance decision-making, programme quality, and developer productivity. Such AI/ML models for software engineering, however, continue to be unreliable, mysterious, and useless. These problems frequently hinder AI/ML model adoption in software engineering methods. Tantithamthavorn et al. [8] emphasized the need of explainable AI in software engineering. They provided a variety of case examples to validate the work. The impact of automation and AI on the practice of software engineering is not well understood. In their research, Latinovic et al. [9] interviewed skilled software practitioners to find a solution to this problem. Their findings demonstrate that incorporation of AI and SE with the effect of automation makes the SDLC process more effective in terms of cost, effort, and time.

Kulkarni et al. [10] proposed an integrated architecture to optimize software development processes. They have incorporated AI activities with the extended waterfall and agile model. In addition to that, five projects were also used to measure and validate the performance of the architecture. UGAM and IoI metrics they used to assess the project against proposed goals.

Hourani et al. [11] have made an attempt to present the AI pillars that can be applied specifically in the testing phase. In terms of AI and software testing, they have also recommended several potential future research directions.

Sofian et al. [12] have given a thorough mapping assessment of the contributions and trends involving AI approaches. The stages of SE, the AI methods utilized in SE, and performance evaluation have all been covered. Following a thorough investigation, their findings indicate that machine learning (ML) has been utilized most frequently for the automation and efficient improvement of SDLC procedures.

Dam [13] highlighted few AI applications and use of machine learning in SE. They have also discussed other areas of AI like: evolutionary computing, knowledge representation, and agent technology to solve software development problems.

3 AI in Software Development Life Cycle Phases

The problems that beset all low-quality software endeavours were the reason why software engineering was developed. The bulk of problems arise from software projects that go above their allotted spending, timelines, and quality standards. Because of how quickly the user environment is changing and the demands that applications must fulfil, there is a greater demand for software engineers [14–16].

Adding another layer of indirection can solve all problems in computer science except for those caused by too many layers of indirections [17, 18]. This is particularly true when the user-friendliness, rapidity, and ease of deployment of software development processes can be demonstrated that can be enabled by the aforementioned programming and modelling abstraction layers. Millions of computer programmes needed to be developed, updated, modified, or improved for specific tasks during the 1960s, and as a result, the need for software programmers increased tremendously. Everyone was writing code in a more complex, inefficient way that didn't take advantage of hardware capabilities, making it difficult for others to understand. This resulted in delays in finding bugs and troubleshooting, which delayed the creation of software and led to what was at the time referred to as a "software crisis".

It is the goal of artificial intelligence (AI) to make computers smart, yet some of the most challenging systems ever built by humans are defined, built, and implemented by software engineers. Current AI approaches to SE are geared towards solving specific problems, such as finding test data for certain branches or criteria and fitting equations to predict system quality. Instead of staying at the lowest level of abstraction, we can transcend individual instances of problems to whole classes of problems, and then present solutions methodologies. Several aspects of the software development process have benefited greatly from software intelligence automation. Design, deployment, quality assurance, and testing are the areas that have been most impacted. AI has been successfully used in earlier parts of the software engineering process, such as real software development and requirements engineering, according to numerous advances and studies [3]. Research suggests that incorporating AI into

the software development process might minimize work and risks throughout the process.

3.1 AI Techniques Used in SDLC Phases

In SDLC life cycle phases, we use different AI techniques to automate things which have their own advantages and disadvantages. Let us discuss them one by one.

A software project begins with the establishment of goals and client specifications by software developers and clients. It is of the utmost importance to schedule and plan software development projects to ensure technical efficacy and financial viability. In the project planning stage, we can utilize search-based software engineering techniques to make the process more effective [18–21].

Recently, numerous requirement engineering strategies have concentrated on determining the requirements for employing AI to solve client problems [17]. Some strategies have since looked into how to use AI with the RE phase. Techniques employed in this era include self-learning algorithms, ontology-based approaches, and big data strategies.

The software project is distinctly organized at the design phase, and development jobs are delegated. The analysis of conclusiveness of code or stories, testing programme logics, and probabilistic planning is done during the software design process. Search algorithms, genetic algorithms, etc., are used to optimize this phase.

Artificial intelligence (AI) techniques are used to convert natural language into code. Procedures for automatic encoding, debugging, and optimization decreased implementation costs and timelines, more effective teamwork.

We use different AI techniques in testing phase to check and test scripts, predict errors probabilistically using big data, shorten test times and reduce costs, integrate existing programmes, and increase efficiency through automated debugging and compiling. Self-adaptive software routines, pattern recognition, and artificial neural networks are employed in the software deployment and maintenance phase for classifying inquiries, evaluating mistakes, and other tasks. Eliminate unnecessary code, increase speed, and simplify maintenance.

4 Discussion

This section highlights major outcomes in the form of table based on our survey study. Table 1 presents the systematic mapping of AI techniques corresponding to each phase of software development life cycle. The first column represents the SDLC phase like: planning, requirement gathering, designing, testing, development, and deployment/ maintenance. Each of the mentioned phase has their own importance in the development of any software. And the aim of this paper is to provide an insight to optimize the product and quality of all those phases with the incorporation of

different AI techniques. The second column represents problem associated with each phase during the software development process. As per the study, there are various issues which hampered the effectiveness of development process. During the software development, the main factors that greatly affect the whole procedure is: customer requirements, project type with their associated risk and status of development team. So, the problem is formulated accordingly. Third and fourth column shows the AI techniques and tools that are identified from the previous work and can be used by the user/ developer to solve the associated problem with the phases. The next column is for methodologies that have been used extensively by many researchers to facilitate different activities in software development phases as per the problem associated with them. And we have observed that ontology-based approach, natural language processing (NLP), search-based approach, and many machine learning techniques that is increasingly being adopted in the system development phases. The last two columns are for outcome after using AI in each phase and limitation corresponding to each technique.

Study observed that major category of AI techniques like: machine learning, deep learning, data-driven approach, and heuristic algorithm in SDLC phases that are extensively used till now. We can also use possible combination of the mentioned AI techniques like: machine learning (ML) with data driven (DD), ML with deep learning (DL), ML with heuristic algorithm (HA), etc., to optimize the outcomes. Figure 1 shows the number of publications between year 2010 and year 2021. And it is being observed that the numbers are increasing gradually as per the importance of software engineering is also increasing.

5 Conclusion

Systematically mapping contributions to the SE literature using AI techniques provides an overview of trends. Several AI techniques have been used in SE in the literature, which is heterogeneous in this study. Literature on SE phases and AI is mainly devoted to these topics. Summarized table on various parameters is presented to get an insight on AI techniques used in each one of the phases of software development life cycle. Machine learning effectiveness and precision are always increasing. Machine learning has been heavily utilized mostly for automation and enhancement tasks throughout the requirement gathering stage. Furthermore, hybrid strategies that combine ML with other AI techniques improved the usefulness of the metrics in their respective assessments. In dynamic or uncertain contexts, this combination can handle multidimensional and multi-variety data effectively. Along with the benefits of AI, we have also listed phase-by-phase restrictions or problems that many researchers encountered when applying AI approaches. It was also noted that AI techniques have a great deal of potential to be effectively applied in numerous other software development operations. There are a couple stages of SE that have seen very little application of AI. The future study stage was to conduct additional

Table 1 Summarized table For AI in software engineering

Phase	Problem associated with each phase	Techniques	Tools	Methods	Outcome	Limitation
Planning	Predictions of effort spent on planning phase, effective project management are very dynamic in nature, software project scheduling problem	Neuro fuzzy model, NLP technique	Fuzzy inference system, TensorFlow, Scikit-learn, Pareto Front, MOCell, jMetal, strategy-driven simulation	Neuro fuzzy approach, ANN, ant colony optimization algorithm, multi-objective metaheuristics based on genetic algorithms	Good estimation capabilities, facilitate the process, fragmented output, alternative release plans, an optimized resolution for software project plan	The amount and size of the instances used More investigation, analysis, and evaluation of additional cases
Requirement analysis	Incomplete, ambiguous and vague requirements, communication gap between the stakeholders	Natural language requirements (NLR), knowledge-based systems (KBS), computational intelligence (CI)	IBM SPSS, object-based NLP, READS tool, and ontology-based software (ODE)	The ontology-based model	Detection of ambiguities in the NL requirements, improved decision-making and communication, managed requirements and model problem domains	The characteristics of the programme being tested frequently interfere with the methodologies that are being created, inadequate management of the development environment

(continued)

Table 1 (continued)

Phase	Problem associated with each phase	Techniques	Tools	Methods	Outcome	Limitation
Designing	There is insufficient communication between the construction and design teams, poor design procedures	DSPL language, case-based reasoning, artificial life, automate transformation, intelligent search, fuzzy system, neural network	QSynth	Genetic algorithms (GAs), worklist algorithm with a two-level optimization technique in the forward search as well as backward search	Improved designing processes and eliminate various issues relating to designing, achieves greater accuracy and scalability	
Development	Need improvement in automation process, time-consuming portion of development process, cost factor	Search-based approach, artificial agents, evolutionary algorithm, constraint programming	NSGA-II, pattern trace identification, detection and enhancement iJava (PTIDEJ)	Refactoring, NSGA-II encoding, analogical reasoning technique for reuse: case-based reasoning (CBR)	Support the reuse of software packages, an efficient method for identifying refactoring opportunities	How to handle systems without a history of applied refactoring is one of the search-based approach's primary constraints
Testing	Test case generation, test case selection, test case prioritization, test case classification, early defect detection	Search-based software engineering, (SBSE) technique, data driven, machine learning, and deep learning techniques	Knowledge-based interactive test script (KITSS), GAs and fuzzy logic tools	Ant colony optimization, genetic Algorithms (GA), ACO algorithm, natural language processing (NLP)	Auto test case generation, predictions and optimization, eliminating redundant test cases, automated selecting and prioritizing test cases	Certain resources become unavailable at unpredictable times are still time-consuming and difficult to define and generate

(continued)

Table 1 (continued)

Phase	Problem associated with each phase	Techniques	Tools	Methods	Outcome	Limitation
Deployment/maintenance	Estimating a software's maintainability effort and cost	Neural network, pattern recognition, genetic algorithm, machine learning	NN topology	Create a runtime decision engine to modify a program, use of neural network	Maintainability index can be predicted, AI can enable the classification of user queries	Only used data in one system, data collection process is semi-automated

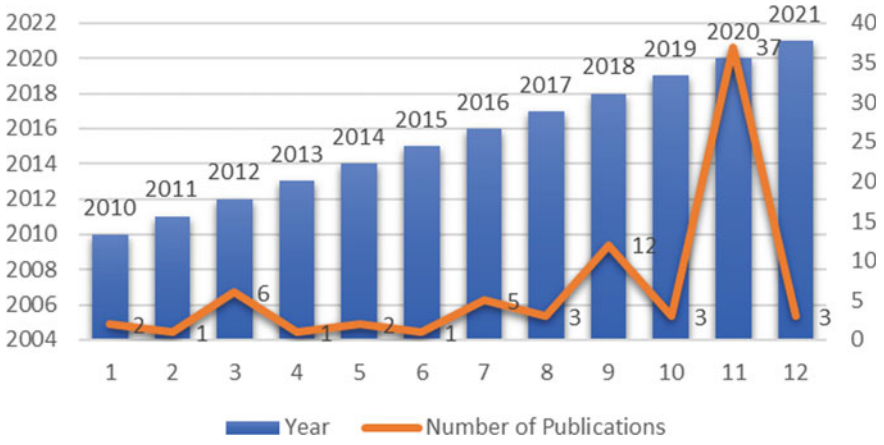

Fig. 1 Publications of artificial intelligence in software engineering (year 2010–2021)

research and review the major findings to pinpoint those SDLC phases in which AI techniques may be applied more frequently in future.

References

1. Feldt, R., de Oliveira Neto, F. G. & Torkar, R. (2018). Ways of applying artificial intelligence in software engineering. In *2018 IEEE/ACM 6th International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering (RAISE)* (pp. 35–41). IEEE.
2. Barenkamp, M., Rebstadt, J., & Thomas, O. (2020). Applications of AI in classical software engineering. *AI Perspectives, 2*(1), 1–15.
3. Narayan, V. (2018). The role of AI in software engineering and testing. *International Journal of Technical Research and Applications.*
4. Al-Ahmad, B. I., Ala'a, A. Z., Kabir, M. F., Al-Tawil, M., & Aljarah, I. (2022). Swarm intelligence-based model for improving prediction performance of low-expectation teams in educational software engineering projects. *Peer Journal of Computer Science, 8*, e857.
5. Ammar, H. H., Abdelmoez, W., & Hamdi, M. S. (2012). Software engineering using artificial intelligence techniques: Current state and open problems. In *Proceedings of the First Taibah University International Conference on Computing and Information Technology (ICCIT 2012)* (Vol. 52), Saudi Arabia: Al-Madinah Al-Munawwarah.
6. Ebbah, J. O. G. (2002). Deploying artificial intelligence techniques in software engineering. *American Journal Under graded Resources, 1*(1).
7. Rech, J., & Althoff, K. D. (2004). Artificial intelligence and software engineering: Status and future trends. *KI, 18*(3), 5–11.
8. Tantithamthavorn, C. K., & Jiarpakdee, J. (2021). Explainable AI for software engineering. In *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)* (pp. 1–2). IEEE.
9. Latinovic, M., & Pammer-Schindler, V. (2021). Automation and artificial intelligence in software engineering: Experiences, challenges, and opportunities. In *The 54th Hawaii International Conference on System Sciences* (pp. 146–155).

10. Kulkarni, R. H., & Padmanabham, P. (2017). Integration of artificial intelligence activities in software development processes and measuring effectiveness of integration. *IET Software, 11*(1), 18–26.
11. Hourani, H., Hammad, A., & Lafi, M. (2019). The impact of artificial intelligence on software testing. In *2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT)* (pp. 565–570). IEEE.
12. Sofian, H., Yunus, N. A. M., & Ahmad, R. (2022). Systematic mapping: Artificial intelligence techniques in software engineering. *IEEE Access*.
13. Dam, H. K. (2019). Artificial intelligence for software engineering. *XRDS: Crossroads The ACM Magazine for Students, 25*(3), 34–37.
14. Shehab, M., Abualigah, L., Jarrah, M. I., Alomari, O. A., & Daoud, M. S. (2020). (AIAM2019) artificial intelligence in software engineering and inverse. *International Journal of Computer Integrated Manufacturing, 33*(10–11), 1129–1144.
15. Harman, M. (2012). The role of artificial intelligence in software engineering. In *2012 First International Workshop on Realizing AI Synergies in Software Engineering (RAISE)* (pp. 1–6). IEEE.
16. Zohair, L. M. A. (2018). The future of software engineering by 2050s: Will AI replace software engineers?. *International Journal of Information Technology and Language Studies, 2*(3).
17. Wilson, G., & Oram, A. (2007). *Beautiful code: Leading programmers explain how they think*. O'Reilly
18. Ghezzi, C., Jazayeri, M., & Mandrioli, D. (2002). *Fundamentals of software engineering.* Prentice Hall PTR.
19. Mayhew, D. J. (1999). The usability engineering lifecycle. In: *CHI'99 extended abstracts on human factors in computing systems*) (pp. 147–148) https://doi.org/10.1145/632716.632805
20. Ferrucci, F., Harman, M., & Sarro, F. (2014). Search-based software project management. In: *Software project management in a changing world* (pp. 373–99). Springer.
21. Chicano, F., Luna, F., Nebro, A. J., & Alba, E. (2011). Using multi-objective metaheuristics to solve the software project scheduling problem. In: *Proceedings of the 13*th *Annual Conference on Genetic and Evolutionary Computation* (pp. 1915–22).