NextGen Scientific Publication
American International Journal of Computer Science and Technology

Volume 4 Issue 1, Pg. No. 44-53, AIJCST-V4I1P105, 2022
https://doi.org/10.63282/3117-5481/AIJCST-V4I1P105

Original Article

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

Sridhar Reddy Bandaru[1], Dhuli Shyam[2], Prabu Manoharan[3], Muzaffer Hussain Syed[4], Uday Kumar Ragireddy[5], Prasanth Varma Addepalli[6]

[1]*Program Management, IT, University or Client: Microsoft, Role: Senior ACE Engineer, State and Country: Redmond, WA.*

[2]*Business Application, IT, University or Client: Nagase Holdings America Corp, Role: Manager, Application & Software Development, State and Country: NYC, NY.*

[3]*Information Technology, University or Client: Bourns Inc, Role: HRIS Manager, State and Country: California, USA.*

[4]*Sr Software Developer, Visual Technologies, Plano, TX.*

[5]*Sr Technical Program Manager, Vdrive IT Solutions, Inc, Richardson, Texas.*

[6]*Data Engineer II, Cox Automotive Corp Svcs LLC, Atlanta, Georgia.*

Abstract:

Advances in intelligent software engineering (ISE) have unfolded in the last years. The development of intelligence software engineering as a new field that exemplifies collaboration between artificial intelligence (AI) and software engineering (SE) is one of these advancements. The integration of AI capabilities into the Software Development Life Cycle (SDLC) across all phases through machine learning applications and frameworks, adaptive learning, and data-driven decision-making techniques is a recent trend in contemporary software engineering. With reference to the fundamental frameworks of machine learning, agent-based, and knowledge-based systems that enable efficient software planning, development, testing, and maintenance, the article delivers a thorough overview of application of AI in SE. This paper addresses way AI improves requirement analysis, shortens design and coding time, boosts defect prediction and reinforces testing and maintenance processes with AI tools and automated processes. Also, the paper discusses challenges in implementing AI-based solutions, including data quality concerns, integration risks, model explainability, ethical risks, and high computational costs. A deep analysis of existing literature also shows the growing synergy of AI studies and software engineering practice. Combining structures, effects, and constraints, this paper provides insightful critique on the changing relationship between AI and SE and on the way forward in forthcoming developments.

Keywords:

Software Engineering, Artificial Intelligence, AI Frameworks, SDLC, Machine Learning, Agent-Based Systems, Knowledge-Based Systems, Automation.

Article History:

Received: 02.12.2021

Revised: 20.12.2021

Accepted: 29.12.2021

Published: 17.01.2022

1. Introduction

The need for Software Engineering (SE) research to take into account the social, cultural, and human elements of software development is becoming more widely recognized. As a result, academics have embraced a variety of social science research techniques [1]. The field of intelligent software engineering has emerged recently as an example of using the collaboration between AI and SE. Issues related to intelligent software engineering are frequently covered in this field. The former, intelligent software engineering, concentrates on incorporating intelligence into techniques developed to handle various software engineering tasks to attain high efficiency and effectiveness. The latter type of ISE focuses on managing various software problems, like AI applications. In fact, the two previously described traits may overlap when integrating intelligence into techniques designed to manage various SE activities for ISE. ISE is by definition a topic of study that encompasses at least the software engineering and AI research fields [2] [3].

The domains of management operational and science research are reportedly seeing an increase in the use of AI. In general, intelligence is defined as the ability to obtain knowledge and apply that knowledge to resolve challenging circumstances. The study and creation of intelligent hardware and software that is capable of reasoning, learning, gathering data, communicating, manipulating, and seeing objects is known as AI. The phrase was first used in 1956 by John McCarthy to describe a field of computer science that aims to replicate human behaviour. Studying computers makes Logic can be seen and acted upon. AI varies from computer science in that it places more emphasis on thinking, perception, and action than psychology does on computers. It makes machines more intelligent and useful [4]. Software engineering and AI have grown independently. Perception, reasoning, and action are made possible by AI research methodologies. The goal of software engineering research is to help engineers create better software more quickly. Software agents are crucial research objects in both Agent-Oriented Software Engineering (AOSE) and distributed AI (DAI). Knowledge-based systems (KBS) are being studied for both knowledge engineering (KE) and learning software organizations (LSO). The software industry uses the AI SDLC to design, create, and test high-quality software. The SDLC seeks to provide high-quality software that is finished on schedule, within budget, and either meets or beyond customer expectations [5].

The paper provides a comprehensive survey of the changes by AI in software engineering through the use of current frameworks, intelligent automation, and improved decision-making capabilities. The paper first covers the theoretical bases of AI and then delves into ML, agent-based, and knowledge-based frameworks that are broadly utilized. It also assesses their involvement in different stages of the SDLC improvement. In addition, the paper identifies AI challenges, limitations, and ethical issues with references to the recent literature. By coupling research and application insights, this work conveys the increasingly mutual relationship of AI with software engineering and points to the future as an open field of innovation.

1.1. Structure of the Paper

The following describes the paper's structure: Section II describes the overview of the AI, including its applications and benefits. Section III describes frameworks of AI in software engineering, including ML, agent-based, and knowledge-based frameworks. Section IV describes the impact of AI in the SDLC. Section V discusses the recent research on the AI-driven software engineering. Section VI concludes the overall research on AI-driven software engineering and shows the future work suggestions.

2. Overview of Artificial Intelligence

Models and Theories of natural and artificial brains and minds must be established in order to understand intelligence. This has been a major philosophical issue since the earliest Greek and Indian literature. This also became a major worry for computer scientists with the introduction of the digital computer in the 1950s. By analyzing, building, and evaluating computers and programs that display characteristics of intelligent behaviour, such as the capacity to identify and categorize patterns, reason from starting points to logical conclusions, and learn from experience, the concurrent development of the theory of computation offered a new set of tools to address this problem [6].

2.1. Applications of AI and Machine Learning

The most beneficial problem in human life is AI. AI is employed in numerous facets of daily life. Watson by IBM, Siri by Apple, Google Now by Google, and Windows Mobile's Cortana are examples of intelligent digital personal assistants for a variety of operating systems. Without requiring physical presence, these assistants help users locate and arrange all necessary items by using speech and gesture recognition [7].

Figure 1. Siri by Apple

Figure 1, depicting Siri on an iPhone interface, illustrates how a voice assistant processes a request, in this case, setting a reminder for a specific date. It showcases the conversational and actionable capability of AI-powered personal assistants to manage tasks based on natural language input.

Figuer 2. Watson by IBM

Figure 2 displays the IBM Watson system housed in a server rack arrangement powered by POWER7 servers. It represents the physical hardware infrastructure that enabled this famous example of cognitive computing and AI.

Figure 3. Cortana by Microsoft

Microsoft Cortana is shown in Figure 3. The user receives all information, like "college, or bus station? where is the nearest restaurant?", or it notifies the user of essential meetings to attend, unfinished work, an alarm clock, personal information, a friend's birthday, etc.

Scientists are conducting numerous future and current studies on humanoids, often known as robotics, as well as human behavior and emotions. High-performance vehicles, radar-equipped missiles, satellites, and navigational systems are also available.

Figure 4. Valkyrine

Figure 4 shows a bipedal humanoid robot in a laboratory setting, likely for testing and development. It illustrates an example of advanced robotics and AI applied to creating human-like machines capable of complex movement.

Figure 5. Google self-driving Car

"Waymo," a Google project, is an autonomous vehicle that runs without a driver. Again, NASA and Google worked together to develop "Valkyrie," the first humanoid astronaut and a shining example of artificial intelligence (Figure 5).

2.2. Benefit of Artificial Intelligence

The advantages of AI in data management are particularly useful for gathering and analyzing massive volumes of data to boost efficiency and personalization [8].

- Increase work efficiency: AI-powered products are very good at completing a particular repetitive activity with exceptional efficiency. A simple argument is that they constantly generate accurate results by removing human error from their work. As a result, they do away with the requirement to assign two groups of workers to do crucial duties during day and night shifts.
- Work with high accuracy: Researchers are trying to teach AI-powered devices to solve difficult problems and carry out essential jobs independently, producing results that are more accurate than those of human equivalents. Due to the important nature of the work, these machines' great accuracy has made them essential for use in the medical industry.
- Reduce cost of training and operation: AI learns new ideas similarly to humans by using ML techniques like neural networks and Deep Learning. In this method, students only need to learn new things instead of constantly writing new code. Globally, a

lot of research and development is being done to build AI systems that optimize their ML capabilities so they can learn new processes much faster.

- Improve Processes: The best thing about employing AI-powered devices at work is that they enable us to collect vast volumes of data regarding their operations. In order to better optimize the processes, such data can be processed to get profound insights through quantitative analysis.

3. AI Frameworks in Software Engineering

AI systems create a formalized platform where AI models can be created, trained, and deployed in software engineering processes. These models are critical in making automation, intelligence, and efficiency in the SDLC possible. They streamline the complex tasks like ML, NLP and knowledge representation, which have become increasingly necessary in contemporary software engineering [9].

3.1. Machine Learning Frameworks

ML frameworks have become a popular resource in software engineering, and are often employed for various tasks such as predictive modelling, automated code generation, defect prediction, and software quality analysis [10].

- TensorFlow: TensorFlow is a heterogeneous, large-scale ML system. TensorFlow leverages dataflow graphs to describe shared state, computation, and the actions that impact that state. It translates nodes of a dataflow graph between multiple processing units (TPUs) within a single machine and across several machines in a cluster [11].
- PyTorch: These two objectives are in fact compatible, as demonstrated by the ML library PyTorch, It offers a Pythonic and imperative programming language It maintains performance and supports hardware accelerators like GPUs while supporting code as a model, simplifying debugging, and integrating with other widely used scientific computing tools [12].
- Scikit-learn: The majority of modern machine learning techniques for medium-scale (unsupervised and supervised) issues can be found in the Python module Scikit-learn. The application uses a general-purpose high-level language to teach the generalist about machine learning [13].

3.2. Agent-Based Frameworks

Agent-oriented software engineering (AOSE) frameworks aid in the construction of intelligent agents capable of operating autonomously to carry out software engineering tasks [14][15].

- JADE: A software framework called JADE (Java Agent Development Framework) was created exclusively in Java. With the aid of a middleware that claims to adhere to FIPA requirements and a collection of tools that facilitate the debugging and deployment process, it simplifies multi-agent systems.
- MadKit: The Multiagent Development Kit (MaDKit), a modular, open-source, scalable multiagent platform developed in Java, is built on the AGR (Agent/Group/Role) organizational model.

3.3. Knowledge-Based Frameworks

Knowledge-based AI frameworks aim at recording and reusing domain knowledge so that the decision-making process in software engineering can be more informed [16].

- Drools: Drools is a Business Rule Management System (BRMS) solution that delivers a rule engine that processes rules and facts to produce output. The main advantage of having the business logic centralized is that changes can be introduced in a very short time and with little money.
- Prolog: Prolog (Programming in Logic) is a declarative programming language that is explicitly intended to be the main logic reasoning and representation of knowledge engine in knowledge-based structures. It enables developers to specify a body of knowledge with facts and rules and then apply an inbuilt inferential process to make conclusions and solve problems

Table 1. Comparison of AI-Based- Frameworks in Software Engineering

Framework Type	Examples	Primary Use	Strengths	Limitations
Machine Learning	TensorFlow, PyTorch, Scikit-learn	Prediction, defect detection, automation	High accuracy, handles big data, strong community support	Requires large datasets; computationally heavy
Agent-Based	JADE, MadKit	Simulation, distributed systems, autonomous	Good for modeling complex interactions; scalable	Difficult to integrate with traditional SE tools

		decision-making		
Knowledge-Based	Drools, Prolog	Rules processing, reasoning, expert systems	High explainability, strong for logic-driven tasks	Limited learning ability; needs expert-defined rules

Table 1 provides comparison of major AI frameworks that are used in software engineering to achieve the core objectives, highlight the main strengths and open up the possible application areas within the SDLC. It compares ML, agent-based, and knowledge-based frameworks in terms of scalability, automation capability, task suitability, and user-friendliness for defect prediction, planning, testing, and knowledge reasoning. It acts as a reference to the most suitable AI technique by listing the features and limitations of the different frameworks. Besides, it shows the different AI paradigms that result in smarter decision-making and higher productivity in software engineering.

4. Impact of AI on Software Development Life Cycle

Every stage of the software development process including design, testing, and deployment is being revolutionized by AI. As a result of automation, prediction, and intelligent decision-making, AI has penetrated nearly every phase of the SDLC [17]. The impact of technology is visible in increased efficiency, better precision, prompt detection of defects, and quick delivery [18].

The subsequent points demonstrate the manner in which AI influences various phases of the SDLC as follows:

4.1. Requirements Analysis

The requirement gathering is easy with the use of AI by analyzing large data like documents, feedback, or reviews in less amount of time. Tools of ML also predict conflicting or missing requirements and make gathering more efficient. The process becomes clear and accurate, and reduces time and effort by using AI.

4.2. Design and Planning

AI-based tools here guide for the required time, possible risk, and suggest possible ways for planning. AI-based tools can analyze previous data and guide for challenges and needs to save time and effort for planning and designing, making the process accurate and fast.

4.3. Coding and Implementation

AI-based tools can guide with better logic building, enhanced code structure, and auto-generated code for saving the time and efforts of the coder. Also, AI-based tools help coders to find bugs easily within the code and suggest multiple possible solutions for fixing them.

4.4. Testing and Quality Assurance

AI-based tools are more useful for testing by providing auto-generated test cases, proper testing reports, and better solutions for improvements, which can be beneficial than manual testers in terms of time, effort, and cost. ML models can help analyze previous bugs from past datasets and help to generate improvement version of code for better quality assurance.

4.5. Deployment and Maintenance

AI-based tools can help monitor real-time or deployed applications and automatically test with number of test cases and find potential bugs and can guide in details for the improvement areas. AI can be beneficial for maintenance by keeping training for future updates in code version and can guide for improvement time to time.

5. Challenges and Limitations of AI in Software Engineering

AI is a significant aspect of contemporary software development, and there are no barriers to its implementation. Although it increases automation, prediction, as well as decision-making, there are a number of obstacles that curtail its success and use in practice [19]. The following is a summary of the difficulties and limitations that could occur while using AI in software engineering [20].

5.1. Data Quality and Availability

Large, tidy, and structured datasets are key determiners of the AI systems, whereas software engineering data is usually noisy, incomplete, or distributed among various tools. This decreases the capacity of AI models and reduces their capability to acquire correct patterns. Organizations would find it difficult to ensure that data is collected and pre-processed appropriately.

5.2. Lack of Explainability and Trust

Most AI systems, particularly DL systems, are black boxes, so the rationale of these systems is not easily readable. Workers might not trust AI-based recommendations or robotic decisions unless they are able to confirm the reasoning that underlies them. This is also an issue while debugging and validation critical software projects due to lack of transparency.

5.3. Integration and Workflow Challenges

The implementation of AI solutions into the current development environment, legacy systems or CI/CD pipelines can be highly costly and necessitate modifications. Such integrations are time-consuming and could create new compatibility problems [21]. Consequently, not all teams can effectively integrate AI tools into their workflows without interruption.

5.4. Ethical, Security, and Bias Concerns

Unless looked at closely, AI models do introduce biases, produce insecure code or reveal sensitive information. This may result in unjust decisions, weaknesses, or non-compliance in software engineering. Good governance, ethics, and constant monitoring are needed in order to make AI responsible in usage.

5.5. High Computational and Maintenance Costs

A lot of computational power is needed for the training, implementation, and upkeep of AI models, which can be costly for small to medium-sized businesses. Moreover, AI tools need to be updated regularly in order to remain precise with the changing codebases and technologies. This is a long-term project costing the company more in operation and accessibility.

6. Literature Review

A complete overview of research on "AI role in SE" is given in this part, along with a brief summary of the findings shown in Table II.

Washizaki et al. (2019). The goal of researchers and practitioners looking into best practices is to create ML systems and software that tackle problems with software quality and complexity. By encapsulating reusable solutions to typical problems in specific contexts, these design techniques are occasionally formalized as design and architecture patterns. Nevertheless, a comprehensive study to gather, categorize, and assess these SE design patterns for machine learning techniques has not yet been completed. Research aggregates both positive and troublesome SE design patterns for ML approaches to give developers a full classification of such patterns. These are the initial findings of an SLR of successful and unsuccessful ML design patterns [22].

Hourani, Hammad and Lafi (2019) This paper covers the primary AI pillars that can be used for software testing. Additionally, it clarifies possible future advancements in software testing and artificial intelligence. The findings imply that AI could improve testing for software results, and AI-driven testing could soon spearhead a new stage of QA initiatives. AI software testing boosts output, shortens time to market, and enables the company to produce more complex software and smarter automated testing [23].

Sahito et al. (2019). The purpose of this study is to address this issue and offer direction to the scientific community. Approximately 57 research publications from the prestigious Journal of Software Engineering that were only published in 2018 were taken into consideration for this investigation. The title, abstract, content, contribution type, area, occurrences, and citation are all taken into consideration while evaluating these papers. According to the results, software testing is more common than development, maintenance, management, and refactoring. This information certainly helps MS academics and PhD aspirants. The main goal of the work is to make suggestions that help the academic community grow stronger [24].

Wangoo (2018) This paper looks at three AI techniques that employ business intelligence, data mining, and ML to promote automated software reuse for overall software development and software production. Business intelligence technologies are used to find code intelligently so that programs and components can be reused. For automated software reuse and the identification of possible research opportunities in the field, a study of numerous AI techniques in the software reuse domain of SE is conducted [25].

Feldt, de Oliveira Neto and Torkar (2018) This study presents the AI in SE Application Levels (AI-SEAL) taxonomy, which classifies applications based on their intended use, the kind of AI technology employed, and the permitted degree of automation. Classify fifteen papers from earlier RAISE workshop editions to demonstrate the utility of this taxonomy. The findings reveal that the taxonomy

permits the identification of diverse AI applications and delivers information about the hazards related to them. It argues that this is essential for companies learning how to integrate AI into software and creating strategies for doing so [26].

Tadapaneni (2017) this poses threats to various companies and causes additional problems for various software engineers. An engineering system's design, construction, and testing present a plethora of difficult tasks. One benefit that software engineers have over other engineers is that they can utilize their own materials and software to tackle the problems that arise from the creation of this system. Because AI algorithms are made to handle some of the increasingly difficult problems in software engineering, they are ideal for these kinds of complex problems. The majority of useful algorithms, techniques, and methodologies have been implemented and utilized by software engineers. The AI community is where it originated. Each of these tasks has an important and useful application that affects several software engineering domains [27].

Table 2. Summary of A Review of Frameworks, and Impact on the Software Development Life Cycle

Reference	Focus On	Key Findings	Challenges	Limitations
Washizaki et al. 2019	An organized analysis of ML system software engineering design patterns	emphasized reusable architectural techniques, identified good and problematic SE design patterns for ML, and offered a basic classification system.	Lack of unified, formalized design patterns for ML systems; high complexity in ML-based software	Preliminary results only; limited number of patterns identified; SLR not yet comprehensive
Hourani, Hammad & Lafi 2019	AI's function in software testing and prospects for AI-driven QA	AI can significantly improve testing efficiency, reduce time-to-market, and enable smarter automated testing	Integration of AI tools into existing QA pipelines; need for expert knowledge to train AI models	Conceptual discussion; lacks empirical testing or validation using real-world datasets
Sahito et al. 2019	Analysis of 57 Software Engineering research articles (2018)	Software testing is the most researched area compared to development, maintenance, and management; provides guidance to researchers and scholars	Ensuring completeness of article selection; rapidly evolving research themes	Restricted to articles from 2018 only; focuses mainly on quantity of research rather than qualitative depth
Wangoo 2018	Use of AI (data mining, BI, ML) for automated software reuse	AI supports intelligent knowledge discovery for identifying reusable code components and improving development efficiency	Complexity in mining and categorizing reusable assets; dependency on data quality	Lacks experimental implementation; focuses mainly on conceptual and theoretical analysis
Feldt, de Oliveira Neto & Torkar 2018	AI-SEAL taxonomy for classifying AI applications in SE	Offers structured classification based on AI type, application point, and automation level; helpful for risk assessment and strategy planning	Difficulty in categorizing hybrid AI systems; evolving AI technologies challenge static taxonomy	Evaluated using only 15 papers; may not generalize to all SE domains
Tadapaneni 2017	Application of AI algorithms to address engineering design, testing, and system challenges	AI algorithms are well-suited for solving complex SE problems; widely applied to development, testing, risk detection, etc.	Complexity of engineering systems; need for effective algorithm selection and tuning	Lacks empirical case studies; primarily describes general applicability rather than specific outcomes

6. Conclusion and Future Work

Artificial Intelligence has been a game-changing technology in domain of SE. The changes to the SDLC are remarkable as the AI-powered technologies not only automate, but also offer analytics with high accuracy and make better decisions. The review reveals that ML, agent-based and knowledge-based models are effective tools in requirements, design, code, testing and maintenance. As a result of AI adoption, software becomes faster, more precise and of higher quality, which is why organizations get the opportunity to build more

reliable and scalable systems. However, problems such as quality of data, lack of explainability, workflow integration, ethical issues, and heavy resource consumption are still factors that prevent the widespread adoption of AI. The surveyed material points to the strong inclination of collaboration between AI and SE and also depicts the potential and the complications. Thus, the AI-powered software engineering is a great avenue to explore, but to see the complete impact, it has to be handled with care, using high-quality datasets and constantly updated intelligent tools.

The studies in the future should investigate more open and explainable AI frameworks that can be smoothly incorporated into intricate software engineering processes. Creating standardized data, better methods of benchmarking, and hybrid models that utilize ML, symbolic reasoning, and agent-based methods will enhance the adoption of AI. Also, it will be necessary to study ethical governance, bias limitation, and cost-effective deployment policies. The increased amount of empirical research and actual case implementation will contribute to the confirmation of the effectiveness of AI in various SE settings and projects of different sizes.

References

[1] K.-J. Stol, P. Ralph, and B. Fitzgerald, "Grounded theory in software engineering research," in *Proceedings of the 38th International Conference on Software Engineering*, May 2016, pp. 120–131. doi: 10.1145/2884781.2884833.
[2] T. Xie, "Intelligent Software Engineering: Synergy between AI and Software Engineering," in *Proceedings of the 11th Innovations in Software Engineering Conference*, Feb. 2018, pp. 1–1. doi: 10.1145/3172871.3172891.
[3] P. C. H. Padmanaban and Y. K. Sharma, "Implication of Artificial Intelligence in Software Development Life Cycle: A state of the art review," *2019 IJRRA all rights reserved*, vol. 6, no. 2, 2019.
[4] A. Pannu and M. T. Student, "Artificial Intelligence and its Application in Different Areas," *Int. J. Eng. Innov. Technol.*, vol. 4, no. 10, pp. 79–84, 2015.
[5] V. Kumari and S. Kulkarni, "Use of Artificial Intelligence in Software Development Life Cycle Requirements and its Model," pp. 1857–1860, 2018.
[6] V. Honavar, "Artificial Intelligence : An Overview," pp. 1–14, 2016.
[7] M. Rupali and P. Amit, "A Review Paper on General Concepts of ' Artificial Intelligence and Machine Learning ,'" vol. 4, no. 4, pp. 79–82, 2017, doi: 10.17148/IARJSET/NCIARCSE.2017.22.
[8] W. Mar, Y. Myo, and K. Khine, "An Analysis of Benefits and Risks of Artificial Intelligence," *Int. J. Trend Sci. Res. Dev.*, vol. 3, no. 5, pp. 1447–1449, 2019.
[9] V. Sresth, S. Nagavalli, and A. Srivastava, "Artificial Intelligence in Software Engineering: Transforming Development Paradigms in Financial Services through Machine Learning Innovations," *World J. Adv. Res. Rev.*, vol. 2, no. 1, pp. 050–062, Apr. 2019, doi: 10.30574/wjarr . 2019.2.1.0007.
[10] S. Mishra, "A machine learning framework for data-driven acceleration of," pp. 1–23, 2018.
[11] P. Barham *et al.*, "TensorFlow : A system for large-scale machine learning," 2016.
[12] A. Paszke *et al.*, "PyTorch : An Imperative Style , High-Performance Deep Learning Library," no. NeurIPS, 2019.
[13] P. Szymański and T. Kajdanowicz, "scikit-multilearn : A scikit-based Python environment for performing multi-label classification," 2018.
[14] H. Fujii, H. Uchida, and S. Yoshimura, "Agent-based simulation framework for mixed traffic of cars, pedestrians and trams," *Transp. Res. Part C Emerg. Technol.*, vol. 85, no. October, pp. 234–248, Dec. 2017, doi: 10.1016/j.trc.2017.09.018.
[15] Geeta, S. Gupta, and S. Prakash, "QoS and load balancing in cloud computing access for performance enhancement using agent-based software," *Int. J. Innov. Technol. Explor. Eng.*, vol. 8, no. 11 S, pp. 641–644, 2019.
[16] A. Aljamel, "A Knowledge-Based Framework For Information Extraction And Exploration," no. January, 2018.
[17] R. K. Vaghela, "A Comparative Analysis of Software development life cycle Models," no. 2277, pp. 4–7, 2015.
[18] D. Jagli and S. Yeddu, "CloudSDLC: Cloud Software Development Life Cycle," *Int. J. Comput. Appl.*, vol. 168, no. 8, pp. 6–10, Jun. 2017, doi: 10.5120/ijca2017914468.
[19] H. Belani, M. Vuković, and Ž. Car, "Requirements Engineering Challenges in Building AI-Based Complex Systems," 2019.
[20] L. E. Lwakatare, A. Raj, J. Bosch, H. H. Olsson, and I. Crnkovic, "A Taxonomy of Software Engineering Challenges for Machine Learning Systems: An Empirical Investigation," in *Agile Processes in Software Engineering and Extreme Programming*, 2019, pp. 227–243.
[21] V. Nerella, "Observability-driven SRE practices for proactive database reliability and rapid incident response," *Int. J. Recent Innov. Trends Comput. Commun.*, vol. 7, no. 8, pp. 32–38, 2019.
[22] H. Washizaki, H. Uchida, F. Khomh, and Y.-G. Guéhéneuc, "Studying Software Engineering Patterns for Designing Machine Learning Systems," in *2019 10th International Workshop on Empirical Software Engineering in Practice (IWESEP)*, 2019, pp. 49–495. doi: 10.1109/IWESEP49350.2019.00017.
[23] H. Hourani, A. Hammad, and M. Lafi, "The Impact of Artificial Intelligence on Software Testing," in *2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT)*, 2019, pp. 565–570. doi: 10.1109/JEEIT.2019.8717439.
[24] S. F. Sahito, A. R. Gilal, R. A. Abro, A. Waqas, and K. Shaikh, "Research Publication Trends in Software Engineering," in *2019 13th International Conference on Mathematics, Actuarial Science, Computer Science and Statistics (MACS)*, 2019, pp. 1–4. doi: 10.1109/MACS48846.2019.9024767.
[25] D. P. Wangoo, "Artificial Intelligence Techniques in Software Engineering for Automated Software Reuse and Design," in *2018 4th International*

Conference on Computing Communication and Automation (ICCCA), 2018, pp. 1–4. doi: 10.1109/CCAA.2018.8777584.
[26] R. Feldt, G. de O. Neto, and R. Torkar, "Ways of Applying Artificial Intelligence in Software Engineering," in *2018 IEEE/ACM 6th International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering (RAISE)*, 2018, pp. 35–41.
[27] N. R. Tadapaneni, "Artificial Intelligence In Software Engineering," *SSRN Electron. J.*, 2017, doi: 10.2139/ssrn.3591807.
[28] Polu, A. R., Buddula, D. V. K. R., Narra, B., Gupta, A., Vattikonda, N., & Patchipulusu, H. (2021). Evolution of AI in Software Development and Cybersecurity: Unifying Automation, Innovation, and Protection in the Digital Age. *Available at SSRN 5266517*.
[29] Padur, S. K. R. (2020). From centralized control to democratized insights: Migrating enterprise reporting from IBM Cognos to Microsoft Power BI. *Int. J. Sci. Res. Comput. Sci. Eng. Inf. Technol*, *6*(1), 218-225.
[30] Bitkuri, V., Kendyala, R., Kurma, J., Mamidala, V., Enokkaren, S. J., & Attipalli, A. (2021). Systematic Review of Artificial Intelligence Techniques for Enhancing Financial Reporting and Regulatory Compliance. *International Journal of Emerging Trends in Computer Science and Information Technology*, *2*(4), 73-80.
[31] Padur, S. K. R. (2019). Machine learning for predictive capacity planning: Evolution from analytical modeling to autonomous infrastructure. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, *5*(5), 285-293.
[32] Attipalli, A., Enokkaren, S., BITKURI, V., Kendyala, R., KURMA, J., & Mamidala, J. V. (2021). Enhancing Cloud Infrastructure Security Through AI-Powered Big Data Anomaly Detection. *Available at SSRN 5741305*.
[33] Singh, A. A. S., Tamilmani, V., Maniar, V., Kothamaram, R. R., Rajendran, D., & Namburi, V. D. (2021). Predictive Modeling for Classification of SMS Spam Using NLP and ML Techniques. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *2*(4), 60-69.
[34] Padur, S. K. R. (2020). AI augmented disaster recovery simulations: From chaos engineering to autonomous resilience orchestration. *International Journal of Scientific Research in Science, Engineering and Technology*, *7*(6), 367-378.
[35] Reddy Padur, S. K. (2021). From Scripts to Platforms-as-Code: The Role of Terraform and Ansible in Declarative Infrastructure Rollouts. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, 621-628.
[36] Kothamaram, R. R., Rajendran, D., Namburi, V. D., Singh, A. A. S., Tamilmani, V., & Maniar, V. (2021). A Survey of Adoption Challenges and Barriers in Implementing Digital Payroll Management Systems in Across Organizations. *International Journal of Emerging Research in Engineering and Technology*, *2*(2), 64-72.
[37] Padur, S. K. R. (2018). Autonomous cloud economics: AI driven right sizing and cost optimization in hybrid infrastructures. *International Journal of Scientific Research in Science and Technology*, *4*(5), 2090-2097.
[38] Rajendran, D., Namburi, V. D., Singh, A. A. S., Tamilmani, V., Maniar, V., & Kothamaram, R. R. (2021). Anomaly Identification in IoT-Networks Using Artificial Intelligence-Based Data-Driven Techniques in Cloud Environmen. *International Journal of Emerging Trends in Computer Science and Information Technology*, *2*(2), 83-91.
[39] Padur, S. K. R. (2021). Bridging Human, System, and Cloud Integration through RESTful Automation and Governance. *the International Journal of Science, Engineering and Technology*, *9*(6).
[40] Attipalli, A., BITKURI, V., KURMA, J., Enokkaren, S., Kendyala, R., & Mamidala, J. V. (2021). A Survey of Artificial Intelligence Methods in Liquidity Risk Management: Challenges and Future Directions. *Available at SSRN 5741342*.
[41] Padur, S. K. R. (2021). From Control to Code: Governance Models for Multi-Cloud ERP Modernization. *International Journal of Scientific Research & Engineering Trends*, *7*(3).
[42] Routhu, K. K. (2021). Harnessing AI Dashboards in Oracle Cloud HCM: Advancing Predictive Workforce Intelligence and Managerial Agility. *International Journal of Scientific Research & Engineering Trends*, *7*(6).
[43] Padur, S. K. R. (2021). Deep learning and process mining for ERP anomaly detection: Toward predictive and self-monitoring enterprise platforms. *Available at SSRN 5605531*.