Journal of Mathematics and Computing Science, 10 (2): 25-30, 2024

eISSN: 0128-0767

Challenges in Developing Artificial Intelligence-based Systems using Classic Software Development Life Cycle

Noorihan Abdul Rahman[1]*

[1*]College of Computing, Informatics and Mathematics, Universiti Teknologi MARA Kelantan, Bukit Ilmu, Machang, Kelantan, Malaysia

Authors' email: noorihan@uitm.edu.my

[*]Corresponding author

Received 17 September 2024; Received in revised 15 October 2024; Accepted 13 November 2024
Available online 6 December 2024
DOI: https://doi.org/10.24191/jmcs.v10i2.4267

Abstract: Artificial intelligence (AI) tools have become an essential part of modern software systems, driving automation, predictive analysis, and intelligent decision-making. However, the development of AI tools using traditional software engineering methodologies, specifically the Software Development Life Cycle (SDLC), presents unique challenges. This paper explores the intricacies and limitations of the classic SDLC in the context of AI-based system development, covering issues related to requirements gathering, design, implementation, testing, deployment, and maintenance. The analysis highlights the need for flexibility and iterative development models to meet the dynamic nature of AI projects. This paper summarises possible considerations that need to be taken care of since building AI-based systems requires a more flexible and iterative approach to accommodate the uncertainty, complexity, and dynamism in its own development.

Keywords: Artificial intelligence, Development, Phases, Software development life cycle

1 Introduction

Artificial intelligence (AI) has emerged as a disruptive technology, influencing diverse sectors including healthcare, finance, manufacturing, and education [1], [2]. The increasing demand for AI solutions has prompted developers and organisations to integrate AI components into their software systems. Since the development of the digital computer in the 1940s, AI has demonstrated that computers can be programmed to carry out very complex tasks [3]. However, building AI-based systems is fundamentally different from traditional software engineering due to their reliance on data, models, and learning algorithms. Due to their complex behaviour, there is a crucial need for a tailored development process for such systems. However, there is still no widely used and specifically tailored process in place to effectively and efficiently deal with requirements suitable for specifying a software solution that uses machine learning [4].

The Software Development Life Cycle (SDLC) is a well-established methodology for guiding the development of traditional software systems [5], [6]. The classic SDLC typically follows a structured approach with distinct phases namely requirements gathering, system design, implementation, testing, deployment, and maintenance. While this linear process has been proven effective for well-defined software projects, it presents notable challenges when applied to AI tool development. AI-based systems like generative models are increasingly integrated into the SDLC. They can automate tasks like code generation, documentation, and comprehensive testing, which accelerates development and improves software quality [7]. This paper examines the unique issues that arise in each SDLC phase when dealing with AI-based development.

2 Overview of AI-based system

The use of AI systems in government and private industries has been documented since 1989 during the IAAI conference. Over the years, AI has expanded for more commercial usage, and this has caused tremendous feedback from many experts and consumers. Recently, the expansion of AI systems has advanced to machine learning (ML); hence, the AI-based development model has been challenging and difficult to explain [8]. In addition, the use of AI has been integrated in the information system environment, and this has led to the uncertainty issue in decision-making among AI-assisted system users. Table 1 shows examples of AI systems that have been widely used by various industries in recent years.

Table 1: Examples of AI-based systems [9]

Problems and System Types	**Specific Applications**
Rule-Based Systems: Widely applied base technology	TurboTax
Credit Card Fraud Alert	Netflix Recommender
Insurance	FareCast, Google Flights, Kayak price predictor
Scheduling: Maintenance, Crew, Gate	Narrative Science GameChanger
Video Games	IBM Watson
Search Engines	Dragon Speech Recognition
Augmented/Virtual Reality	Amazon Robotics / Kiva Systems
Photo Face Recognition	Roomba
Handwriting Recognition: Mail Sorting, ATM-Checks	Kinect
Translation	Driver-Assist / Self-Driving Vehicles
Deep Learning	Siri, Cortana, Amazon Echo
Robotics	TurboTax

To elaborate Table 1, Rule-Based Systems (RBS) are a type of AI system that uses a predefined set of rules to make decisions or solve problems. These rules are often expressed as “if-then” statements and form the basis for reasoning within the system. It contains the set of rules, which are typically derived from expert knowledge.

Explainable AI (XAI) tools can be used to provide clear reasons why a transaction was flagged as fraudulent. Netflix recommender can analyse user preferences and behaviour patterns for detecting anomalies in the use of Netflix by its users. In the insurance industry, AI is transforming the industry by optimising processes, improving decision-making, and enhancing customer experiences. This can increase operational efficiency, customer satisfaction, and profitability in the insurance sector, whereas in scheduling applications, AI can assist to automate and optimise scheduling itself, especially in resource and time allocation to accomplish certain transactions. AI also plays a crucial role in developing software applications across domains such as search engines, augmented and virtual reality, robotics, and deep learning by enhancing functionality, scalability, and user interactivity with AI capabilities such as Natural Language Processing (NLP), content generation, and autonomous decision-making.

The next section evaluates possibilities of challenges during SDLC activities in developing AI-based development environment.

3 Evaluation of SDLC phases for AI-based development environment

This section elaborates phases of SDLC which are related to AI-based system development issues.

A Requirements Gathering

Based on the literature, during the requirements gathering phase, the three main challenges identified in developing AI-based systems during this phase are uncertainty in problem scope, evolving requirements, and a lack of clear success criteria. In terms of uncertainty in problem scope, AI systems often operate in domains characterised by uncertainty and complexity. The problem to be solved may not have a clear and defined solution. For instance, an AI tool designed for image recognition might have unpredictable performance based on the quality and variability of the data provided [10]. Next, AI systems often start with exploratory research, which evolves as new data becomes available or as the understanding of the problem deepens. Traditional SDLC struggles to accommodate these changing requirements, which can lead to rework or delays [11]. There is also a weakness in clarifying success criteria. Defining success in AI systems is more difficult than in traditional software. AI performance is typically measured in terms of accuracy, precision, recall, or other statistical measures rather than functional correctness. The thresholds for acceptable performance can be ambiguous or may change over time, depending on the use case [12].

B System Design

Secondly, system design in the SDLC focuses on creating a blueprint for the software system, covering architecture, data flow, user interfaces, and system components. There are a few challenges identified in developing AI systems during this phase. The first challenge is related to model selection uncertainties. AI tools depend heavily on machine learning models, and the choice of model is not always clear at the outset. Different machine learning models, such as decision trees, neural networks, and support vector machines, have distinct strengths and weaknesses, and their suitability may depend on factors such as data quality, feature availability, and computational resources. This introduces uncertainty in the design phase, where traditional SDLC assumes that design choices can be made early and remain relatively stable [13]. Next, AI systems are data-driven, for which the developer needs to design efficient data integration. Challenges arise in integrating disparate data sources, handling unstructured data, and ensuring data quality. Unlike traditional software systems, where inputs are typically deterministic, AI systems must be designed to process large, heterogeneous datasets, which adds significant complexity to the design process [14]. Thirdly, AI models require experimentation and tuning, which often means that the design of the AI system needs to be flexible and open to change. The traditional SDLC, with its emphasis on upfront design, does not accommodate the iterative, experimental nature of AI development well [15].

C Implementation

The next phase reveals the implementation phase, which involves coding the system according to the design specifications. For AI tools, this phase presents several distinct challenges. Firstly, AI systems often require specialised programming languages and frameworks such as Python with TensorFlow or PyTorch [16]. These tools are different from those used in conventional software development, requiring developers to have a strong understanding of both software engineering principles and machine learning techniques. This can create a steep learning curve for development teams unfamiliar with AI [17]. Secondly, AI components must often be integrated into existing systems, which may have been developed using different programming languages or architectures [18]. Ensuring seamless integration between AI models and traditional software components is a complex task that can introduce additional technical debt if not managed properly. Next, AI systems, particularly those based on deep learning, can be resource-intensive, requiring significant computational power such as graphics processing units (GPUs) for both training and inference. Traditional SDLC assumes that computational constraints are predictable; however, in AI, these constraints can vary depending on the size and complexity of the models being trained [19].

D Testing

Testing is one of the most critical phases of the SDLC, ensuring that the system meets the specified requirements. In traditional software development, testing involves verifying that the system behaves as expected under predefined conditions. However, testing AI systems introduces unique challenges. Unlike traditional software systems, where outputs are deterministic, AI systems produce probabilistic results [20]. For example, in a predictive model, the same input might yield slightly different outputs depending on factors such as random initialisation, data shuffling, or stochastic learning processes. This makes it difficult to define clear test cases and to verify correctness in the traditional sense [21]. In addition, there are possible issues such as data dependency and during continuous testing. The performance of an AI system is highly dependent on the quality and diversity of the data used during testing [21]. In many cases, acquiring representative test data is a significant challenge. Moreover, traditional test case generation strategies may not be applicable since AI models are tested based on statistical performance measures such as accuracy and F1-score rather than pass or fail outcomes.

E Deployment and Maintenance

Once the AI system has been developed and tested, the deployment phase follows, where the system is moved to production. Maintenance involves ensuring the system's continued operation and addressing any issues that arise post-deployment. Deploying AI models is not as straightforward as deploying traditional software components [23]. AI models may need to be re-trained or fine-tuned in real-time based on new data. This requires a robust infrastructure to support ongoing model updates, which is not typically a part of the traditional SDLC's deployment framework. Over time, AI models may become less accurate due to changes in the underlying data distributions, a phenomenon referred to as model drift. Maintaining AI systems requires ongoing monitoring and retraining of models to ensure continued performance, which is a more dynamic and resource-intensive process than traditional software maintenance [24]. The deployment of AI systems introduces new ethical challenges, such as bias, transparency, and accountability [25]. Ensuring that AI systems operate within legal and ethical guidelines is an ongoing challenge during the maintenance phase, especially as regulations evolve.

4 Conclusion

The traditional SDLC offers a structured and linear approach to software development that has proven effective for many types of software projects. However, applying this methodology to AI tool development presents significant challenges. AI systems require a more flexible and iterative approach to accommodate the uncertainty, complexity, and dynamism inherent in machine learning and data-driven development. To address these challenges, organisations may need to adopt hybrid development methodologies, such as Agile or DevOps, which allow for more frequent iteration, testing, and adaptation throughout the AI development life cycle. DevOps addresses the challenges of developing AI systems by streamlining collaboration between data scientists, developers, and operations teams through shared tools and workflows. It automates repetitive processes like data preprocessing, model training, testing, and deployment.

Additionally, DevOps may help to ensure scalability, reliability, and consistent monitoring of AI models in production environments. Meanwhile, agile software development addresses the challenges of AI system development by emphasising iterative progress, enabling teams to refine models and algorithms incrementally based on feedback and changing requirements. It fosters collaboration between data scientists, developers, and stakeholders, ensuring alignment and adaptability throughout the development process. Agile focuses on short development cycles, and continuous improvement helps manage the complexities of integrating data pipelines, training models, and deploying AI systems effectively. In summary, regardless of the software development product, the software process must be carefully examined in order to produce an accurate mechanism for allowing desirable outputs and hence produce its expected outcome as the software output.

Acknowledgements

The author would like to thank Universiti Teknologi MARA Kelantan Branch for supporting the research process on Artificial Intelligence.

Conflict of Interest Statement

The authors agree that this research was conducted in the absence of any self-benefits, commercial or financial conflicts and declare the absence of conflicting interests with the funders.

References

[1] W. Liu, G. Zhuang, X. Liu, S. Hu, R. He, and Y. Wang, "How do we move towards true artificial intelligence," in *2021 IEEE 23rd Int Conf on High Performance Computing & Communications; 7th Int Conf on Data Science & Systems; 19th Int Conf on Smart City; 7th Int Conf on Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys)*, 2021, pp. 2156–2158.

[2] M. N. P. Ma'ady *et al.*, "Making Sense of Developing Artificial Intelligence-Based System in Software Development Life Cycle Manner and Addressing Risk Factors," in *2023 6th International Conference of Computer and Informatics Engineering (IC2IE)*, 2023, pp. 244–249.

[3] H. Jindal, D. Kumar, S. Kumar, and R. Kumar, "role of artificial intelligence in distinct sector: A study," *Asian J. Comput. Sci. Technol.*, vol. 10, no. 1, pp. 18–28, 2021.

[4] H. Belani, M. Vukovic, and Ž. Car, "Requirements Engineering Challenges in Building AI-Based Complex Systems," in *2019 IEEE 27th International Requirements Engineering Conference Workshops (REW)*, 2019, pp. 252–255.

[5] A. Gupta, A. Rawal, and Y. Barge, "Comparative Study of Different SDLC Models," *Int. J. Res. Appl. Sci. Eng. Technol*, vol. 9, no. 11, pp. 73–80, 2021.

[6] O. E. Olorunshola and F. N. Ogwueleka, "Review of system development life cycle (SDLC) models for effective application delivery," in *Information and Communication Technology for Competitive Strategies (ICTCS 2020) ICT: Applications and Social Interfaces*, 2022, pp. 281–289.

[7] A. Soni, A. Kumar, R. Arora, and R. Garine, "Integrating AI into the Software Development Life Cycle: Best Practices, Tools, and Impact Analysis," *Tools, Impact Anal. (June 10, 2023)*, 2023.

[8] S. Laato, M. Tiainen, A. K. M. Najmul Islam, and M. Mäntymäki, "How to explain AI systems to end users: a systematic literature review and research agenda," *Internet Res.*, vol. 32, no. 7, pp. 1–31, 2022.

[9] J. Smith, R. G. and Eckroth, "Building AI Applications: Yesterday, Today, and Tomorrow," *AI Magazine*, pp. 6–22, 2017.

[10] B. D. Radhakrishnan and J. J. Jaurez, "Explainable artificial intelligence (XAI) in project management curriculum: Exploration and application to time, cost, and risk," 2021.

[11] S. S. Gill *et al.*, "AI for next generation computing: Emerging trends and future directions," *Internet of Things*, vol. 19, p. 100514, 2022.

[12] L. Floridi, J. Cowls, T. C. King, and M. Taddeo, "How to design AI for social good: Seven essential factors," *Ethics, Governance, Policies Artif. Intell.*, pp. 125–151, 2021.

[13] I. Aradea, I. Supriana, and K. Surendro, "ARAS: adaptation requirements for adaptive systems: Handling runtime uncertainty of contextual requirements," *Autom. Softw. Eng.*, vol. 30, no. 1, p. 2, 2023.

[14] F. Badra, "A dataset complexity measure for analogical transfer," 2021.

[15] M. Haakman, L. Cruz, H. Huijgens, and A. Van Deursen, "AI lifecycle models need to be revised: An exploratory study in Fintech," *Empir. Softw. Eng.*, vol. 26, no. 5, p. 95, 2021.

[16] Y. H. Liu, Python Machine Learning by Example: Build Intelligent Systems Using Python, TensorFlow 2, PyTorch, and Scikit-Learn. Packt Publishing Ltd, 2020.

[17] O.-C. Novac *et al.*, "Analysis of the application efficiency of TensorFlow and PyTorch in convolutional neural network," *Sensors*, vol. 22, no. 22, p. 8872, 2022.

[18] A. Aldoseri, K. N. Al-Khalifa, and A. M. Hamouda, “Re-thinking data strategy and integration for artificial intelligence: concepts, opportunities, and challenges,” *Appl. Sci.*, vol. 13, no. 12, p. 7082, 2023.
[19] S. Deng, H. Zhao, W. Fang, J. Yin, S. Dustdar, and A. Y. Zomaya, “Edge intelligence: The confluence of edge computing and artificial intelligence,” *IEEE Internet Things J.*, vol. 7, no. 8, pp. 7457–7469, 2020.
[20] I. Van de Poel, “Embedding values in artificial intelligence (AI) systems,” *Minds Mach.*, vol. 30, no. 3, pp. 385–409, 2020.
[21] P. Hase and M. Bansal, “Evaluating explainable AI: Which algorithmic explanations help users predict model behavior?,” *arXiv Prepr. arXiv2005.01831*, 2020.
[22] I. H. Sarker, “AI-based modeling: techniques, applications and research issues towards automation, intelligent and smart systems,” *SN Comput. Sci.*, vol. 3, no. 2, p. 158, 2022.
[23] M. M. John, H. H. Olsson, and J. Bosch, “Developing ml/dl models: A design framework,” in *Proceedings of the International Conference on Software and System Processes*, 2020, pp. 1–10.
[24] Z. S. Ageed *et al.*, “A state of art survey for intelligent energy monitoring systems,” *Asian J. Res. Comput. Sci.*, vol. 8, no. 1, pp. 46–61, 2021.
[25] J. Borenstein and A. Howard, “Emerging challenges in AI and the need for AI ethics education,” *AI Ethics*, vol. 1, pp. 61–65, 2021.