International Journal of Emerging Trends in Computer Science and Information Technology
ISSN: 3050-9246 | https://doi.org/10.63282/30509246/IJETCSIT-V4I3P104
Eureka Vision Publication | Volume 4, Issue 3, 32-40, 2023

Original Article

Bridging the Gap Between Traditional Software Engineering and Modern AI Development Practices

Santhosh Chitraju Gopal Varma
Software Developer, United States of America (USA).

Abstract - AI and ML have been advancing at a very fast rate and this has led to a number of challenges in the field of SE. AI-based systems are the creation of new paradigms, tools, and workflow, which are quite different from traditional SE processes. This paper discusses the differences between conventional SE and AI development to establish the gaps that separate both fields before presenting a structured approach to filling such gaps. The areas that may be addressed by the research include modifications in SDLC, the training of an AI model, practices for version control and other appropriate ethical concerns. Also, this paper provides applications where authors make an effort to introduce AIDD into traditional SE practices. Last, the future trends and recommendations for further enhancement of the integration of AI-SE are presented.

Keywords - Artificial Intelligence, Software Engineering, Machine Learning, Agile Development.

1. Introduction

Software engineering has been dominated by methods like the Waterfall Model, Agile, and DevOps for quite some time now. However, with the current development of more sophisticated applications based on AI, some of these paradigms have become anomalous. The AI models do not work on classical programming paradigms but are data-centric in their development and continuously learn from the data.

1.1. Importance of Traditional Software Engineering and Modern AI Development

Traditional SE has a central function of constructing dependable, sustainable, and scalable systems. It offers well-proven processes like Waterfall, Agile, and DevOps to provide a procedure to follow when developing, testing, and deploying software applications. Another advantage of SE is in its modularity and reusability: developers can design the program in such a manner as to enable the flexibility of continuing to add new abilities without worrying about having to restructure the whole system. [1-4] Software engineering also focuses on the code structure, testing, and controlling versions of created applications to reduce the number of glitches or bugs. Also, SE is deterministic, which would imply that software, as a result, can be predicted and controlled; therefore, it would always be easy to debug and solve existing errors. It does not stop at enterprise applications and large digital systems and continues to reach embedded systems and large-scale distributed systems where it is imperative to have well-developed, reliable, and well-commented software codebases that allow for stability of operations. In addition, as a result of DevOps, both CI/CD have also been adopted primarily to improve software deployment in terms of both speed and stability. On the other hand, modern AI Development has altered the ways that communications, transactions, and almost every software utility, perform and operate.

AI is stochastic in its development and aims at creating models that can learn from huge volumes of data, unlike static SE which works according to program algorithms. This capability makes AI so useful in its current areas of use like speech-to-speech or speech-to-text recognition, object recognition, suggestion, as well as Natural Language Processing (NLP). Unfortunately, it is also a general-purpose technology that mostly provides important prospects for various fields, especially with the integration of large and unstructured data such as healthcare, finance and services, and cybersecurity. Thus AI models are capable of identification of anomalous conditions, pattern recognition and prediction and can supply solutions that conventional typed programs cannot. The combination of Traditional Software Engineering and AI Development is getting more and more pivotal as AI models must be integrated into conventional software to provide reliability, security as well as scalability. Whereas software engineering gives a more systematic approach towards development, AI includes intelligence in the processing making the applications intelligent and smart. These two domains are very relevant in the contemporary technological world since there is still a need for the deterministic design of software but also with provision for AI engagement. Thus, the use of MLOps, AI-aware version control, and explainability frameworks with standard SE practices will become more critical in developing new-generation powerful and reliable AI products.

1.3. Challenges in Bridging SE and AI Development

- **Lack of AI-Aware Version Control Systems:** Version control systems such as Git are commonly used in traditional software engineering to track the changes made to code, manage the authors of such changes and preserve the change history. Dynamics of developing AI brings other forms of software artefact peculiarities: datasets, model

architectures, hyperparameters, and training pipelines.' While in SE, versioning primarily concerns the source code, in AI, it is crucial to have model versioning to guarantee replicability and stability across the versions. There is software to handle this problem, for instance, DVC for Data Version Control together with MLflow, but these are not seamlessly integrated into traditional software engineering environments.

- **Difficulty in Debugging AI Models:** Different from conventional software, it is challenging to track how the decision process unfolds because of the black-box nature of functions in AI models, as well as the fact that the models contain logical errors. In a decision tree, minor alteration in the data and a few hyperparameters leads to vagaries in the model. Alongside, ordinary debugging methods like utilizing logs and step-by-step-tracing do not suffice for the AI models; thereby, tools like SHAP and LIME are utilized to explain the model predictions. To ensure reliable integration of AI in SE it is necessary to develop tools that would correspond to AI-specific requirements of SE for debugging with real-time monitoring, boundary conditions detections and root causes.

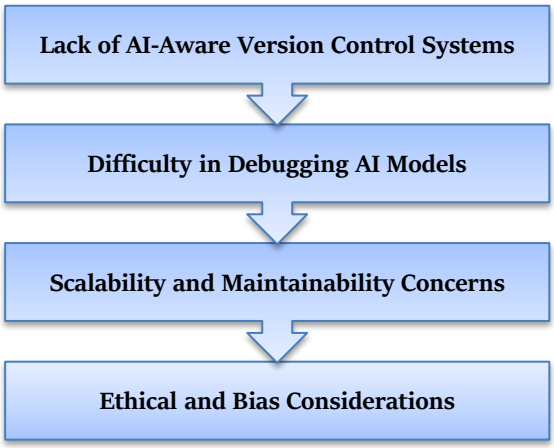

Figure 1. Challenges in Bridging SE and AI Development

- **Scalability and Maintainability Concerns:** Traditionally, SE mainly aims at modularity and code reuse to maintain the product in terms of feature enhancements and defect correction in the long run. On the other hand, AI models or Artificial Intelligence models need to be trained and updated from time to time to prevent ageing when data distribution changes. This brings scalability issues because AI models have to be easily deployable, monitorable and upgradable without affecting the system's performance. While there are frameworks like MLOps and Kubeflow, as well as TensorFlow Extended (TFX) applied to machine learning to deliver an integrated solution that enhances the model operating process, their integration with traditional SE processes does not remain easy. One of the consistencies faced while implementing AI into context with software engineering sessions is the issue of scalability possibility to implement AI-supported applications in a way which will keep them manageable in terms of scale.
- **Ethical and Bias Considerations:** Deep learning models are not immune to these prejudicial and discriminated characteristics; they tend to learn prejudice if it is present in data sets. SE unlike traditional SE, where the behaviour of the code is coded, uses learnt patterns from data and hence, fairness and accountability of models are difficult to determine. Some of the issues like data privacy, the existence of bias and lack of transparency need performance assurance on how biases can be detected and rectified. It means techniques like fairness in the algorithms, training diversified data sets, and compliance with regulatory requirements such as GDPR and AI ethics. Overcoming these ethical issues is relevant to creating robust, explainable, and non-biased AI-based software systems.

2. Literature Survey

2.1. Traditional Software Engineering Practices

There have been many changes seen throughout software engineering where applications were developed in monolith architecture which are now changing to be broken down into many microservices architectures which focus on such things as modularity, reusability and maintainability. At the beginning of software development, the Waterfall Model was used, which is a linear model of software development that in itself is divided into phases, namely requirement collection and analysis, designing, coding and implementation, testing, deploying and finally the phase of maintenance. [5-9] Fewer suitable for projects with easily changing requirements because the Waterfall Model did not involve flexibility. Due to this, Agile Development was developed which has short cycles where the team can deliver incremental improvements, be able to take feedback from users and quickly adjust to the change. Additionally, advancing the software engineering practice, there DevOps as a method of development and operations for providing Continuous Integration and Continuous Delivery (CI/CD). DevOps involves several factors of using deployment, testing, and monitoring that aim to provide quicker deliveries and a higher stability rate for the applications. There are subsequent advancements in conventional software engineering practices that have enhanced productivity, reliability as well as the satisfaction of the users.

2.2. Modern AI Development Practices

AI engineering is different from traditional software engineering as it is mainly a data-driven field of development that aims at creating ML models rather than relying on a set of strict rules. AI is the process of collection, cleaning, training, and testing of models based on data sets that must be of good quality and varied. Unlike ordinary coding, the AI models, being learning models, learn from the new data fed to them and need to be retrained from time to time. Moreover, AI development leads to ambiguity of output certification since AI programmes apply probabilistic reasoning that is quite unlike the deterministic axiomatic rigor. This implies that testing and debugging using AI models is complicated as the predictions obtained may differ in terms of inputs. To address this issue, AI developers rely on statistical verification and qualification, the use of such strategies as A/B testing, as well as methodologies that allow for the interpretation of the models. The path to building, deploying and maintaining the AI models is known as MLOps and is incorporated into the AI model to make it scalable and easily manageable in production.

2.3 Comparison of SE and AI Development Practices

This can be explained by the fact that traditional SE is largely based on a paradigm that emphasizes code, that is, a set of operations that are programmed to accomplish certain functionalities. However, AI development operates on a fundamentally different basis since this functioning emerges from data-based reasoning and does not depend on strict sets of rules. Compliance with this goal is high; their data dependency signifies that the quality and quantity of data used for training directly affect the performance of an AI system. The verification of the output of an AI system is challenging since it is not probable like other software that uses deterministic testing where a specific input expects a certain output. Moreover, whilst updating a feature involves enhancing it in some way or the other for better performance, AI models need to be retrained from time to time to match the performance of the current data.

2.4. Challenges in Integrating AI into SE

The implementation of AI in traditional software engineering poses some challenges that should be met to kick-start innovation. Among them, one of the significant challenges that can be mentioned is the absence of AI-aware version control systems. Unlike the traditional development used by SE, the models need to manage datasets, parameters, and pipelines requiring special tools for version control such as DVC and MLflow. The last important concern relates to the issue of a need for new methods for debugging information. While conventional approaches in debugging involve identifying errors in logic or syntax, deep learning models act as a black box which presents issues as to why such a certain prediction was made. To enhance the interpretability, transparency techniques are required including explainability tools like SHAP and LIME as well as real-time AI monitoring solutions. Last but not least, limitations such as the highest ethical issues and concerns are still crucial since biases are learned by the AI system through the dataset which it was trained with. To follow the guidelines on how to integrate AI into SE in a non-biased way, it is necessary to incorporate bias detection frameworks, use the data by several authorships, and apply fairness-aware algorithms. Meeting these challenges will be instrumental in enhancing the creation of more dependable, increases in credibility and expansiveness of AI-enabling software solutions.

3. Methodology

3.1. AI-Augmented SDLC Stages

- **Requirement Analysis:** The requirements gathering process, goes beyond conventional functional requirements, to specify datasets to include, and dataset quality dimensions in the AI-augmented software development. [10-15] This includes identifying the kind of data, the quantity, and how to acquire it and at the same time satisfying data protection laws. Each party decides on additional or unique criteria related to knowledge and understanding of AI, including performance characteristics, unnecessary bias, and managerial dilemmas. Moreover, it encompasses the identification of the proper set of data labelling techniques as well as the identification of challenges of the given domain that may affect the accuracy of the model.

  Design: The conceptual phase of AI deployment for development can be concentrated on such steps as defining the features to include or exclude, defining the architecture for a model, and planning for system integration. Feature selection is the next step where one needs to select only those features that have the best impact in regards to model prediction with the least inclusion of additional features. When it comes to models, it is decided if one should use typical machine learning or deep learning networks or whether a mixture of both will suffice depending on the requirements of a project. Moreover, this phase also encompasses the aforementioned four attributes of the AI system: interface with other components, data flows, data-input and data-output control, and conformance to best practices of the particular industry.
- **Implementation:** In the implementation phase, AI models are programmed and the AI algorithms are inserted into the software system. Machine learning scripts are coded by the developers using TensorFlow, PyTorch, or Scikit-learn with the data pipeline to set ingest, transform, and store data. AutoML tools may be used in the process of model building: For similar reasons, engineers also consultant and enhance the performance as well as scalability of the code to run efficiently within a production environment. This step may include back-and-forth training and adjusting the model depending on the performance and results of the model and the needs of the business.
- **Testing:** The last step in AI integration within the context of SDLC entails thorough exercises aimed at testing the model as regards reliability, fairness as well as accuracy. Several types of tests, including those for single AI

components, for the integration of components, and testing against new data, for example. The effectiveness of the models is evaluated by performance indicators including accuracy, precision, recall, F1-score, as well as mean squared error. Finally, adversarial testing and explanation analysis are performed to unveil bias security and ethical issues in AI decision-making.

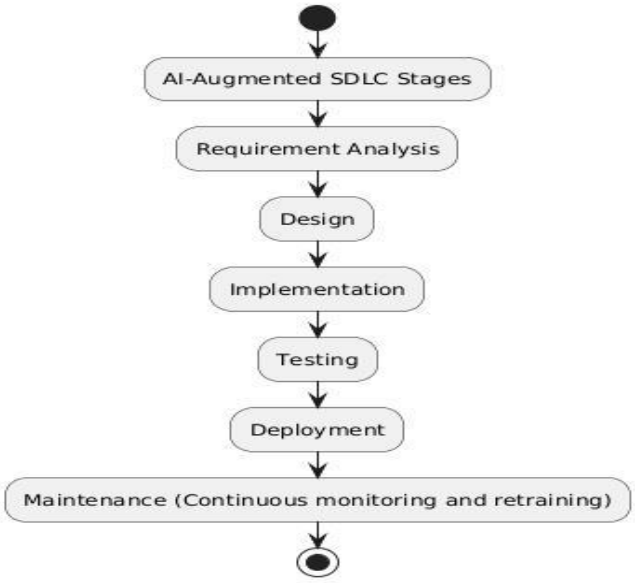

Figure 2. AI-Augmented SDLC Stages

- **Deployment:** SAFMC defines the process of deploying bifurcating trained models ready to make predictions in production environments. Considering using such containers as Docker or Kubernetes for the containerisation of the software, implementing APIs for its integration with other software, and boosting up the Inference Speed so there would not be high latency time for prediction. They also pay attention to the selection of monitoring tools to be used to track model performance in real-world environments and the identification of plans B and C in the event of failure. The following continuous deployment strategies may be used to address updating the models without having to shut down the system:
- **Maintenance:** The final phase involves regular updates of the AI models' effectiveness in their proper functioning, adjustments to the models, etc. The major disadvantage of AI models is that they are subject to decay, by a process known as concept drift where the distribution of the data changes which means low accuracy. There are always opportunities to refine the results, contain unsatisfactory performance in the future, introduce new vendors, and ensure that the best tools are actively used by providing constant auditors, feedback, and retraining programs for models. Moreover, AI governance has provisions which embrace the ethical use of artificial intelligence while routine updates help in tackling new threats. This phase is important for the continuity of AI-driven software solutions in the future periods.

3.2. Framework for AI-SE Integration

3.2.1 Data Versioning Techniques:

Explaining data about versions implies that the versioning of the datasets that are used in training or evaluation processes is tracked. [16-20] This is important to sustain model integrity, as well as for fixing problems and meeting the legal guidelines. DVC (Data Version Control) is a program where dataset changes are monitored, metadata tracked, and coordinating teamwork in projects related to AI is done using tools such as MLflow. It is therefore possible for the resulting AI to be more transparent and reliable by focusing on data versioning techniques that could be used for rebuilding the model from the ground up by using the correct and reliable data.

3.2.2 MLOps for Automated AI Model Deployment:

MLOps is the discipline that covers the processes needed after the model's development in terms of training, validation, deployment, and monitoring. With regard to AI, it incorporates CI/CD, namely continuous integration and continuous deployment, of models. The various tools such as Kubeflow, MLflow, SageMaker, and others for the management of MLOps help in automating retraining, model tracking, and versioning that aid in decreasing the time to time and incorporating the scale and reliability of the AI solutions. This cumulatively improves the work process of the data scientists and software engineers in the particular field of AI model deployment.

3.2.3 AI-Enabled DevOps Pipeline:

An Industrialized DevOps is the integration of Artificial Intelligence (AI) related automation and data analytical process in the ordinary DevOps process to enhance the software delivery practices. AI techniques can assist in the allocation of

resources, forecasting of system downtimes, and boosting the security checking during the CI/CD process. The application of AI for testing, anomaly detection, and performance tuning thereby helps organizations to deliver software faster with better security features. This means that regardless of where in the software system some component lies, it will be constantly optimized for performance and reliability.

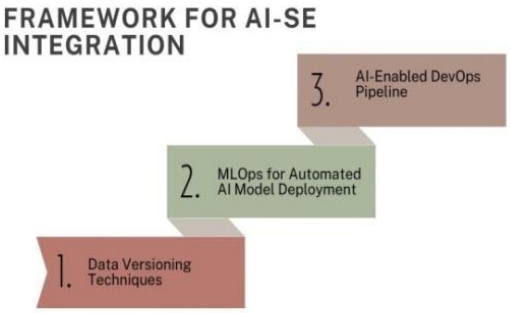

Figure 4. Framework for AI-SE Integration

3.3. AI Development Lifecycle

- **Requirement Analysis:** It involves the establishment of project objectives, requirements and expected achievement of the Artificial Intelligence project. The activities that are to be performed during this phase are a definition of the nature of the problem to be solved, the choice of data sources and the choice of measures of the problem solution efficiency. This also includes assessing the domain-specific concerns, rules or laws that might affect the use of AI, as well as the possible ethical issues arising from the use of the AI. Requirement analysis should always provide a clear guide for the development of AI solutions especially to meet the existing business objectives and the users' expectations.

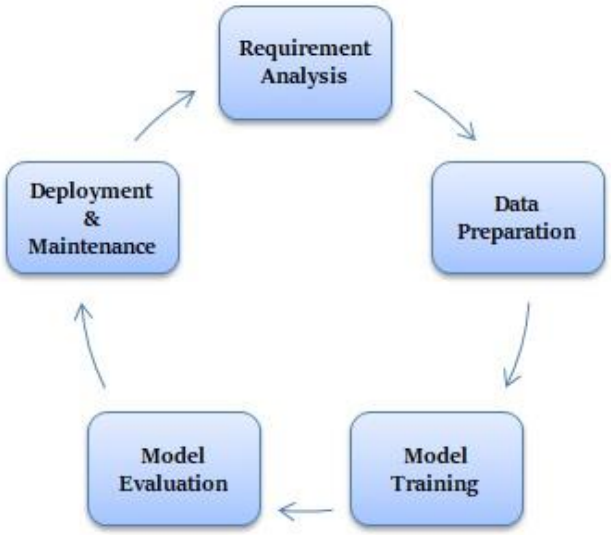

Figure 5. AI Development Lifecycle

- **Data Preparation:** Pre-processing is an indispensable phase in developing ideas foundational on data, which includes data gathering, cleansing, normalization as well as enhancement. Raw data is cleaned to ensure that it is of good quality after the data collection phase is completed. In this set-up, feature engineering approaches come in handy to facilitate the ideal method in the model specification to lead to better outcomes. Also, the given data is used in the form of training, validation and testing datasets to avoid bias. It is said that the quality of data greatly determines the quality of the whole model.
- **Model Training:** The model training stage in machine learning is when the algorithms that are to be used are trained via prepared data. Through batch learning, where it employs supervised, unsupervised, or reinforcement learning, the parameter of the model is set or modified. Debugging methods such as concept-sensitive JIT optimization models result in adjusting the weight of the model through gradient descent to minimize the error. The process of optimization of model hyperparameters is a crucial step to increase its efficiency and accuracy. Its result is a developed model that can make predictions of its own on new data.
- **Model Evaluation:** After a model is built, it needs to be tested and have its accuracy, precision, recall, F1-score or mean squared error, etc. computed. The cross-validation method assists in generalization and also in the avoidance of

over-learning. In addition, more stability measures are performed to find out weaknesses or biases. Model assessment is important to understand whether the system achieves the set objectives before it is released to the public.

- **Deployment & Maintenance:** Deployment is a process of deploying the final model which has been trained in the production environment or other words as a live model Predictor. This comprises API installation, model deployment, and ability in terms of the performance of the system. The processes of checking model performance continuously to look for signs of data shift and decrease in accuracy are called maintenance. It is possible to set up or develop self-executing retraining processes to ensure that the model is exposed to newer data. This indicates that regular audits, security updates, and ethical compliance checks keep the AI system relevant and productive.

3.4. Tools and Technologies for AI-SE Integration

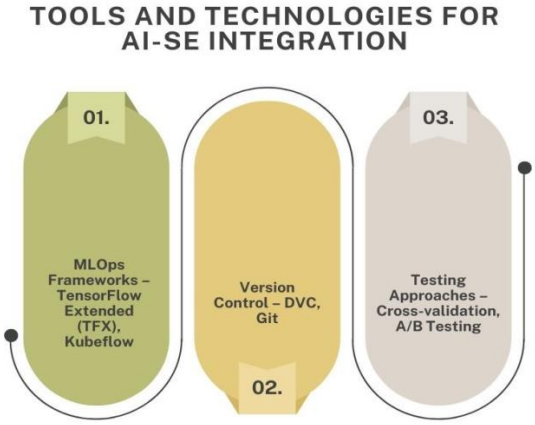

Figure 6. Tools and Technologies for AI-SE Integration

- **MLOps Frameworks – TensorFlow Extended (TFX), Kubeflow:** MLOps frameworks envision the process of AI model development and deployment through enhancing automation of procedural activities. TFX stands for TensorFlow Extended and this is a data-validation, training and serving solution which is compatible with TensorFlow. Based on the Kubernetes, Kubeflow helps to manage the machine learning process to include scalable containers for the AI models. These tools enable AI pipeline automation, correct reproducibility and easy deployment on both cloud and local environments.
- **Version Control – DVC, Git:** It is crucial to maintain different aspects such as code, datasets and AI models under source control as early and as often as possible in the development process. DVC The idea of DVC is an extension of Git for datasets, ML models, and versioning experiences that makes reproducibility and collaboration simpler. Git, a versioning system mostly applied to code, helps to track changes, and branches, and work together on AI projects. With the integration of DVC with Git, the team can have an optimized and scalable method of managing AI.
- **Testing Approaches – Cross-validation, A/B Testing:** The comment below can be used to assert that rigorous test strategies are important in assessing the performance and validity of AI models. Cross-validation is a technique of partitioning data into several subsets test and modeling datasets where the model is trained and evaluated severally in manners that keep the different data portions distinct. A/B testing helps to compare two models in real conditions to select the model with better performance. These testing technologies can facilitate to improvement of the capabilities of AI systems as well as refine decision-making in real use cases.

4. Results and Discussion

4.1. Case Study: AI-Augmented Software Development

As a case study, the developed recommendation system was integrated with the AI-Software Engineering (AI-SE) framework to show how it is applied to software development. Notably, the system was designed to use machine learning algorithms to personalize the contents and employ MLOps practices to update and redeploy them.

Table 1. Efficiency Gains from AI-Augmented Development

Metric	Improvement (%)
Model Retraining Time	40%
Deployment Errors	30%
Maintainability Score	50%

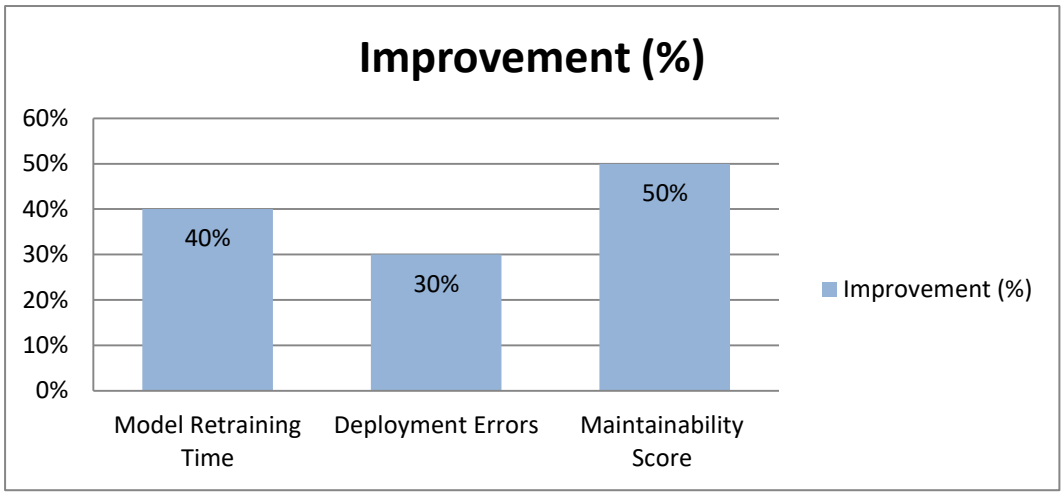

Figure 7. Graphical representing Efficiency Gains from AI-Augmented Development

- **Model Retraining Time:** Data pipelines and version controlling also help in enhancing the speed of retraining the model since more human interference with the data is discouraged and the correct version of the data is used. Generally, traditional retraining requires the collection, cleaning, and transformation of data that are obtained from new sources, and this is more time-consuming and erroneous. Hence with the help of automation tools like DVC Data Version Control and MLOps- Kubeflow the retraining turns out to be much efficient which in turn reduces the time needed for updating and fine-tuning models by a whopping 40%. This is important to enable the AI system to be flexible enough to cope with dynamic data patterns thus increasing reliability and up-to-date information.
- **Deployment Errors:** Since the model integration tests and monitoring activities are automated, AI-based DevOps reduces the likelihood of encountering erroneous implementation instances. Typically in traditional applications, software deployment results in human errors, different configurations in different environments and other discrepancies. CI/CD pipelines integrated with AI-based tools also allow the detection of problems with the model during integration and validation before its usage. As such, deployment errors are cut by a third; thus, having enhanced stability, reliability, and user experience of the system and software products.
- **Maintainability Score:** Versioning is an important process in regard to machine learning work because it is fundamental to the reproducibility, audibility, and sustainability of the work. The use of version control is important when you are working with models, for example, the ability to track model changes, the modifications on the dataset or the tune of the hyperparameters can be very overwhelming. Version control tools such as Git for code and DVC for data and models allow developers to version control AI models, roll back when necessary and provide clear documentation of updates. This provides better maintainability and enables them to debug problems and check compliance while improving the interface between the AI and the software engineering teams.

4.2. Performance Metrics Comparison

This paper involved the comparison between the conventional Software Engineering Development (SE) and the Artificial Intelligence Software Engineering (AI-SE) hybrid methodologies concerning the aspects of key performance indicators. The procedures and steps of the AI-SE approach were scalable and easy to maintain; moreover, it provided developers with the opportunity to optimize development time.

- **Development Time:** The last Kin of traditional Software Engineering (SE) is that development time is considerably longer as compared to that of the integrated approach due to the time taken for performing coding, testing, and debugging. AI in Software Engineering (AI-SE) saves the time required in implementing software by automating various tiring steps like code generation, testing and deployment. Among those tools include machine learning-based code companions like GitHub Copilot and other automated CI/CD pipelines to allow for faster iterations minimizing human loopholes. Thus, through the adoption of AI-SE, time-to-market is improved as well as resource cycles which calls the focus of the teams towards innovation as opposed to mundane tasks.
- **Maintainability:** Conventional SE results in moderate maintainability since it relies on documented documentation, debugging and software maintenance procedures that are typically human-initiated. On the other hand, AI-SE introduces improved application maintainability through issues of automated version control, intelligent debugger solutions, and adaptive systems. The use of AI-based solutions for monitoring allows for identifying inefficiencies, and likely failures and provides solutions in real-time processes. Also, there is a range of MLOps tools such as TFX and Kubeflow that help in maintaining the AI model long-term and, thus, impact technical debt on the software.
- **Scalability**: In the case of SE in IT, scalability issues are normally encountered due to fixed architectures, and most of the infrastructures are managed manually. AI-SE, on the other hand, relies on the scale-out strategy by using the cloud, Docker, Kubernetes, and AI-based static/dynamic resource management. AI-based systems can reorganize

workloads according to the workload, manage workload optimally and handle bulk data processing. This flexibility makes AI-SE most suitable to be deployed where or when large numbers of requests are expected to occur in the demands of more large-scale enterprises where the configuration of more extensive is likely to be required.

4.3. Discussion on AI Integration Challenges

Nevertheless, the introduction of AI into SE has its difficulties that should be considered to achieve reliability and fairness as well as create efficiency. Among them, two are most compelling, which are the challenges in the development of AI debugging tools, and the second, the ethical issue of AI/ML decision-making. An important problem among the issues involved with integrating AI is the ability to debug it, which proves to be a much more tedious task than debugging software. Static system debugging, also called deterministic, also works on the basis that the programmer can refer back to the line of code and fix the mistake that caused the error. However, AI models are stochastic, which makes their behavior dependent on large data sets, probabilistic algorithms and dynamic patterns. The debugging process is also affected by this to become more unpredictable and challenging. For instance, problems such as model drift, which is the case where the performance of an AI system is worse than expected because the data distribution of the situations it is handling is different from when the model was trained, can be difficult to diagnose by using ordinary debugging methods. To address these challenges, there is a need to develop AI-specific debugging tools that will help enhance the understanding of the specific reasoning conducted by the model. Some of the methods utilized in debugging AI models include; SHAP (SHapley Additive Explanations) and LIME (Local Interpretable Model-agnostic Explanations). Additionally, log monitoring and performance management of models as well as tracking data drift and anomalies will prevent an organization from experiencing serious issues. One of the other crucial problem areas is that of ethical issues and how to avoid them on the side of bias in the AI systems.

AI models rely on data acquired from previous learning and often pre-existing data may carry some level of bias in its database based on gender, race, or economic status. If negative, then these will produce decisions that are stereotyped or discriminative which are ethical and eventually legal issues. For instance, there have been problems with AI recruiting that have provided negative preferences to certain demographics at the detriment of others in terms of fairness and policy measures. To address this issue, there should be bias detection frameworks and the training of the intelligently aware bias-free models to avoid the introduction of bias. Some of the measures enabling AI fairness include Google's Model Cards and IBM's AI Fairness 360 toolkit, which offer information about how a model arrives at a certain decision or identifies bias. Thirdly, the inclusion and exclusion of minorities in training datasets also minimize bias in AI models and provide fairness as well. All in all, the overall application of AI in software engineering has inherent benefits, which have to be hindered by these challenges for the sound adoption of AI. Through using the approach that creates targeted debuggers, the improvement of the model's interpretability, as well as introducing ethical AI practices, it is possible to take advantage of AI without negative consequences while having a fair, free-from-bias and most importantly, reliable system.

5. Conclusion

This paper investigates the role that traditional software engineering is playing in the new age of artificial intelligence development, and the differences that are associated with it suggesting a new blended SDLC that incorporates artificial intelligence. Traditionally SE uses deterministically designed logic, structured programmed environment, and debugging manually whereas using AI for software construction involves data-based approaches, probabilistic models and learning systems. When embedded into SDLC, many aspects that are time-consuming and repetitive in conventional machine learning can be automated, including model training, deployment, version control as well as the application of MLOps. In the case of an efficiency proof on an AI recommendation system, the improvements were shown as follows: the time to retrain the system was reduced by 40%, there was a 30% reduction in the number of deployment errors and increased maintainability using techniques like AI versioning. Besides, comparing the Traditional SE and AI-SE Hybrid development approach emphasized that AI can enhance the development time, maintainability, and scalability. However, there are issues which arise with the use of integrated AI, especially in debugging the models or a particular model and addressing ethical issues. Such problems necessitate specific AI debugging tools and techniques, online monitoring, and security frameworks that address the issue of bias. In conclusion, it is imperative to accentuate that the incorporation of the AI-augmented SDLC can enhance the response of contemporary applications that rely heavily on artificial intelligence dramatically.

5.1. Future Work

Despite this work laying the basis for guiding the integration of AI with SE, there is more research that needs to be done to enhance AI-SE practices. The first one is the need for new approaches to AI debugging since regular approaches are not suitable for working with stochastic AI systems. Further studies should be made to grow tools specifically targeting AI, to use the model interpretation methods (such as SHAP, LIME), and to combine it with an automated anomaly detection system. Also, as AI models deepen their roots in software engineering, the need to maintain the explainability of the models will be another significant way that needs to be fulfilled. Another future research direction can be considered to apply the AI ethic in the SE life-cycle. In this post, solutions to ethical issues that AI is likely to create soon are discussed: the biases in algorithms, the issue of fairness, and accountability for anything that AI does. AI implications and applications of AI regulation should be further investigated in AI future research, as well as the creation of governance frameworks and algorithmic bias, detection methods and AI transparency rules to avoid adverse impacts of AI on marginalized groups. Last, but not least, it is crucial to

improve AI version control systems to address the topics of model reproduction and their life cycle. The models change over time with the help of retraining, which requires the use of a more effective form of version control that could take into account data, parameters and variations. Research must design version control systems specifically for AI-first MLOps, where the efficiency and sustainability of the machine learning software systems can be optimised. Therefore, by considering the above challenges, future works on AI-augmented software engineering methods can improve the extent to which AI systems are kept optimal for scalability and can be easily explained and implemented on ethical grounds in projects.

References

[1] Shneiderman, B. (2020). Bridging the gap between ethics and practice: guidelines for reliable, safe, and trustworthy human-centered AI systems. ACM Transactions on Interactive Intelligent Systems (TiiS), 10(4), 1-31.

[2] Aitken, A., & Ilango, V. (2013, January). A comparative analysis of traditional software engineering and agile software development. In 2013 46th Hawaii International Conference on System Sciences (pp. 4751-4760). IEEE.

[3] Pressman, R. S. (2005). Software engineering: a practitioner's approach. Palgrave Macmillan.

[4] Boehm, B. W. (2002). A spiral model of software development and enhancement. Computer, 21(5), 61-72.

[5] Beck, K. (2000). Extreme programming explained: embrace change. addison-wesley professional.

[6] Jüngling, S., Peraic, M., & Martin, A. (2020, March). Towards AI-based Solutions in the System Development Lifecycle. In AAAI Spring Symposium: Combining Machine Learning with Knowledge Engineering (1).

[7] Humble, J., & Farley, D. (2010). Continuous delivery: reliable software releases through build, test, and deployment automation. Pearson Education.

[8] Kim, G., Humble, J., Debois, P., Willis, J., & Forsgren, N. (2021). The DevOps handbook: How to create world-class agility, reliability, & security in technology organizations. It Revolution.

[9] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. nature, 521(7553), 436-444.

[10] Amershi, S., Begel, A., Bird, C., DeLine, R., Gall, H., Kamar, E., ... & Zimmermann, T. (2019, May). Software engineering for machine learning: A case study. In 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP) (pp. 291-300). IEEE.

[11] Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., ... & Dennison, D. (2015). Hidden technical debt in machine learning systems. Advances in neural information processing systems, 28.

[12] Zhang, J. M., Harman, M., Ma, L., & Liu, Y. (2020). Machine learning testing: Survey, landscapes and horizons. IEEE Transactions on Software Engineering, 48(1), 1-36.

[13] Breck, E., Cai, S., Nielsen, E., Salib, M., & Sculley, D. (2017, December). The ML test score: A rubric for ML production readiness and technical debt reduction. In 2017 IEEE international conference on big data (big data) (pp. 1123-1132). IEEE.

[14] Ribeiro, M. T., Singh, S., & Guestrin, C. (2016, August). " Why should I trust you?" Explaining the predictions of any classifier. In Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining (pp. 1135-1144).

[15] Mitchell, M., Wu, S., Zaldivar, A., Barnes, P., Vasserman, L., Hutchinson, B., ... & Gebru, T. (2019, January). Model cards for model reporting. In Proceedings of the conference on fairness, accountability, and transparency (pp. 220-229).

[16] Brachman, R., & Levesque, H. (2004). Knowledge representation and reasoning. Elsevier.

[17] Pham, P., Nguyen, V., & Nguyen, T. (2022, October). A review of ai-augmented end-to-end test automation tools. In Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering (pp. 1-4).

[18] Battina, D. S. (2016). AI-Augmented Automation for DevOps, a Model-Based Framework for Continuous Development in Cyber-Physical Systems. International Journal of Creative Research Thoughts (IJCRT), ISSN, 2320-2882.

[19] Agrawal, A., & Menzies, T. (2019). Is AI different from SE? arXiv preprint arXiv:1912.04061.

[20] Wirtz, B. W., Weyerer, J. C., & Geyer, C. (2019). Artificial intelligence and the public sector—applications and challenges. International Journal of Public Administration, 42(7), 596-615.

[21] Zhuang, Y. T., Wu, F., Chen, C., & Pan, Y. H. (2017). Challenges and opportunities: from big data to knowledge in AI 2.0. Frontiers of Information Technology & Electronic Engineering, 18, 3-14.

[22] R. Daruvuri, “Harnessing vector databases: A comprehensive analysis of their role across industries,” International Journal of Science and Research Archive, vol. 7, no. 2, pp. 703–705, Dec. 2022, doi: 10.30574/ijsra.2022.7.2.0334.

[23] R. Daruvuri, “An improved AI framework for automating data analysis,” World Journal of Advanced Research and Reviews, vol. 13, no. 1, pp. 863–866, Jan. 2022, doi: 10.30574/wjarr.2022.13.1.0749.