Journal of Soft Computing Paradigm (ISSN: 2582-2640)
www.irojournals.com/jscp/

Building Trust in AI -A Simplified Guide to Ensure Software Quality

Sahithi Devalla[1], Manas Kumar Yogi[2]

[1]B.Tech III Year Student, CSE-AI&ML Department, Pragati Engineering College (A), Surampalem, A.P., India

[2]Assistant Professor, Computer Science and Engineering Department, Pragati Engineering College (A), Surampalem, A.P., India

E-mail: [1]sahithidsnl@gmail.com, [2]manas.yogi@gmail.com

Abstract

In recent years, Artificial Intelligence (AI) has emerged as an innovative technology in a variety of areas, including software development. The demand for high-quality software has grown in tandem with the increasing complexity of applications and user expectations.AI-driven approaches are revolutionizing traditional software development methodologies by automating and augmenting various stages of the development life cycle, leading to improved efficiency, reduced costs, and enhanced software quality. This research explores the crucial role of AI in developing high-quality software and its impact on the software development process. Firstly, it discusses how AI technologies like machine learning, natural language processing, and deep learning can facilitate requirements gathering, analysis, and validation, leading to better understanding and refinement of user needs. Next, it delves into the significance of AI in automating the coding process, such as generating code snippets, fixing bugs, and optimizing performance, thus accelerating development and reducing human errors. Moreover, the paper highlights the pivotal role of AI in software testing and quality assurance. AI-powered testing tools can execute comprehensive tests more efficiently, detect defects, and predict potential software vulnerabilities, thereby enhancing the overall reliability and robustness of the software product. Additionally, AI techniques can enable real-time monitoring and analytics, allowing developers to identify and address issues promptly during the software's operational phase. Furthermore, the paper addresses the ethical considerations and challenges associated with AI in software development, including bias in training data, interpretability of AI-driven decisions, and potential job displacement for software developers.

Keywords: AI, Quality, Defect, Improvement, Process

DOI: https://doi.org/10.36548/jscp.2023.3.001
Received:15.06.2023, received in revised form: 17.07.2023, accepted: 03.08.2023, published: 18.08.2023

1. Introduction

The rise of Artificial Intelligence (AI) has brought about a revolutionary transformation, reshaping the interaction with technology, processing of data, and making crucial decisions. As AI continues its rapid expansion across diverse industries, the vital role of software quality in AI implementation becomes increasingly apparent.

Software quality refers to the ability of software to work as per user requirement. In general, it evaluates the effectiveness of the software created, and the way the software sticks to that design. It is evaluated based on several criteria, including functionality, reliability, usability, performance, maintenance, flexibility, and safety.

A good software must provide quality assurance. As nowadays developers and companies are more focused on deploying new codes without any proper software testing. Without quality, speed has very little value. Due of its distinct features, ensuring software quality in AI is a significant challenge. Achieving software quality in AI requires a broader perspective beyond mere functionality and user experience. Aspects such as reliability, transparency, fairness, and ethical considerations take big role in gaining trust and confidence in AI.

Integrating AI into software development necessitates a combination of traditional software engineering abilities, domain experience, and an understanding of AI techniques. It is a dynamic industry that necessitates staying up-to-date with the newest breakthroughs and standard practices in order to create successful AI-powered software solutions. Some metrics for creating high-quality software are listed in table.1 below [1].

The utilization of the machine learning algorithms and data processing, applications based on AI principles have spawned numerous sectors and brought about numerous technical improvements.

The Current AI software landscape is distinguished by a wide range of advanced quality practises that have evolved to satisfy the needs of this dynamic industry. These practises are intended to ensure that artificial intelligence systems are accurate, dependable, secure, and ethical. [2] The main software quality practises in AI are:

1. Data Quality and Pre-processing: The foundation of good AI systems is high-quality data. To remove noise, biases, and inconsistencies that might affect AI performance, rigorous data collecting, cleaning, and pre-processing approaches are used.

2. Model Training and Validation: AI models are developed using powerful algorithms on massive datasets. Cross-validation, hyperparameter tuning, and regularisation approaches are used to assure model accuracy and generalisation.

3. Understanding and Interpretability: As AI models become more complicated, it becomes critical to understand their decision-making processes. To provide insights into model predictions, techniques such as feature importance analysis and model visualisation are used.

4. Ethical and Fair AI: It is vital to ensure that AI systems are unbiased and fair. To address potential ethical problems about AI algorithms, fairness assessments, bias detection, and mitigation strategies are applied.

5. Security and privacy: AI systems frequently handle sensitive data. Encryption, access controls, and secure deployment practises are used to preserve data privacy and avoid potential risks.

The current level of AI software quality reflects a synthesis of complex practises aimed at ensuring accuracy, dependability, and ethical use. The research study discusses the problems in the AI software quality and the approaches to ensure the AI software quality.

2. Inducing Quality in AI Software

The motivation for inducing quality in AI software is multifaceted and extends beyond technical considerations. Quality ensures the accuracy, reliability, and safety of AI applications while promoting user trust, innovation, and societal impact. By prioritizing quality, The full potential of AI, is unlocked and can make it a transformative change that helps various domains.

[3] There are standards that outline the required properties of the system in order to ensure the desired quality of a software system. ISO/IEC, for example, specifies quality models for software & systems, i.e., a hierarchy of quality features of importance and the ways to determine and analyse them. These quality models cannot be implemented directly due to the heterogeneous nature of data-driven software components. Some must be redefined, while others must be introduced (e.g., trustworthiness, fairness). It is important to be very specific about which quality is relevant for which component of the overall system.

A study proposes to use the risk-based testing (RBT) paradigm from traditional Software engineering to validate data in ML-based software systems. RBT is a practical and extensively used approach that uses software product risks as a guiding element to guide all phases of the testing process [4]. RBT, as a sort of software testing, uses software system risks to address decision problems throughout the testing process. Risk is defined as a factor that has the potential to cause future negative effects and is expressed by the probability of occurrence and subsequent impact.

Masuda [5] conducted a review to discover approaches for assessing and improving the software quality of machine learning systems. They examined 101 publications and concluded that the topic is still in its infancy, particularly in terms of quality criteria other than operational accuracy and safety.

Lorenzoni [6] conducted a systematic literature study on the subject of ML system development. During the years 2010 and 2020, 33 papers were examined and categorized into 10 challenges and 13 solutions under seven Software Engineering techniques.

Using the snowballing method with the papers identified during a database search serving as the starting point set. Backward and forward snowballing are ways of systematically analysing the references and citations of a group of papers. Almost 30% of papers were found during snowballing [7].

3. Metrics for High Quality Software

The Creating high-quality AI software involves a number of interconnected factors that must be carefully considered throughout the development lifecycle.

Data Quality*:* Foundation of AI systems is data. For good model training, high-quality, relevant, and diverse data is required. It is important to ensure data cleanliness, correctness, and accurate labelling in order to reduce biases and inaccuracies flowing through the model.

Model Selection**:** It is critical to select the best model architecture and method for your unique situation. Take into consideration variables such as complexity, and the availability of pre-trained models. A carefully chosen model can have a better impact on the performance of an AI program.

Training*:* Proper training involves arranging the training pipeline, defining loss functions. Balancing underfitting and overfitting is important while creating a model that responds well to new data.

Validation: It means evaluating model's performance on unseen or new data. Techniques like cross-validation helps to ensure that the model isn't just memorizing the training data but is actually learning.

Ethical Considerations*:* Addressing bias, fairness, and ethical concerns is crucial. AI systems should not reinforce existing biases or discriminate against certain groups. Fairness audits and ongoing monitoring are needed to identify and rectify biases.

Continuous Improvement*:* AI models degrade over time as data distributions shifts. Regularly updating and retraining models using new data ensures that the software remains accurate and relevant. This might also involve deploying mechanisms to collect user feedback and iteratively improve the software.

[8] Some metric ranges required in solving the problem and improving the software quality is shown below in table.1.

Table 1. Software Metrics

Metric	Explanation	Range
Lines of code	The number of statements that are executable per element	1 to 200
Cyclomatic Complexity	This is a measure of the work required for understanding and evaluate the component	1 to15
No. of paths	Returns the total amount of non-cyclic pathways per component	1 to 80
Frequency of Vocabulary	The total of the distinct number of operands and operators needed for the definition of a component	1 to 4 3 to 500

Length of the program	Overall length of program	3 to 7
Size on average	The component's average statement size	

4. Challenges in Software Quality of AI

When it comes to data that is trained there are certain specifications that the data must meet

compatibility with the context data that is trained to match with real world. [10]

(i) Incorruptibility – Data have to be free form errors.

(ii) Completeness – Data should be complete.

This macro area has four unresolved problems: code understandability, code quality guidelines, training concerns, and a lack of tools for software quality improvement.

AI and software engineering commonly use the same phrases to describe distinct objectives. Machine learning professionals view technical terms quite differently than software programmers (for example, "testing," "regression," "validation," and "model"). This problem frequently leads to misunderstandings amongst developers from various backgrounds [1].

There are also a few issues in the coding. According to a survey, Jupyter notebooks are saturated with poor quality code. Some of the challenges that respondents encountered in their AI/ML code are as follows:

The most frequently expressed concern is data management. It was highlighted as the most critical challenge in AI/ML code development; it includes improper labelling, poor dataset construction, data administration, data preparation, and pipeline [6].

The second concern was the frameworks' stability and the results they produced. For certain businesses, it is tied to security issues, and any library they use must be checked by the IT security department before being utilised in a project.

Miscommunication was another major concern. Participants defined miscommunication as failure of data scientists to communicate effectively with software

engineers, customers, or business experts. Invisible faults or flaws that reduce performance were also identified as major concerns.

(iii) Study to determine best Software Engineering for ML Applications:

To determine and evaluate SE best practises for ML applications, Alex, et al. [9] undertook a study of 21 relevant papers (including white and grey publications). They gathered 29 finest practises and conducted an additional survey with 313 experts in software to determine the level of adoption and effect of these practises.

Teams of practitioners employing ML elements in the project were the audience that was targeted. Particular initial queries were included to allow teams to be separated into those who create and deploy ML apps, those that utilise ML but do not create an application, and those that don't utilise ML at all.

Adoption of practises classified by demographic characteristics. All plots display the proportion of correct responses.

1)Adoption of practises dependent on the type of practise

- Traditional 15.6% adoption
- Modified 11.3% adoption
- New 16.9% adoption

2)Adoption of practises dependent on the kind of data

- Tabular Data, Text, Images, Videos, Audio, Time Series, Graphs

3)Linear regression models that describe the reliance of effects on practises that impact the software quality.

- Agility - The team can immediately test new algorithms and data, as well as review and implement new models.
- Software Reliability - The software created is of excellent quality (both technically and functionally).
- Team Efficiency Experts - with diverse skill sets (for example, data science, software development, and operations) work effectively together.

- Traceability Production - model outcomes may be readily tracked back to model settings and input data.

5. Future Implications

In relation to a Machine Learning-based technique, The data set is stated as full when it can be tuned to create the aimed output without the need for fine-tuning, Due to a lack of software development skills, AI code examples frequently provided online by AI professionals and professors are of low quality [6]. The explanation might be the complexity of executing tests, as well as the issue of un predictable AI-algorithm outcomes. Metamorphic testing can be a suitable alternative for unit testing.

Software testing usually involves techniques such as: Unit Testing, Integration Testing, Regression Testing, Functional Testing, White Box Testing, Black Box Testing and Automated Testing. The application of these tests in software testing makes the testing process leaner and more efficient.

In an AI software deep learning process, data quality validation and services have three aspects.

- Checking the quality of raw data
- Validating the quality of training data
- Evaluating the quality of test data

A testing framework for Machine Learning (ML) is provided, which introduces the concept of regression testing as well as a mechanism for ranking the accuracy of new ML algorithm versions [7]. To cope with large-scale multidimensional input data sets (various image graphs and movies) [8], each model must be trained with new testing methods. The figure.1 shows the various quality variables for image identification software.

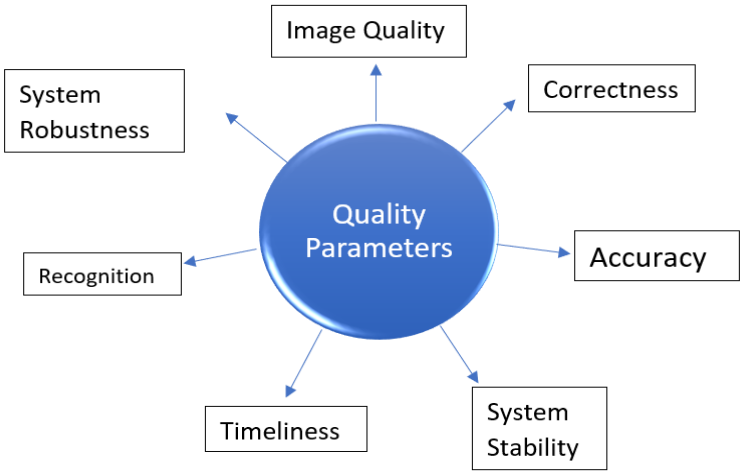

Figure 1. Various Quality Variables for Image Identification Software

5.1. Software Quality for AI Systems

The key aspects ensuring the software quality of the AI system is shown in table.2 below

Table 2. Key Aspects

Key Aspects	Ensuring Quality Data Inputs
	Comprehensive Test Data Selection
	Validating Knowledge-Based Decisions
	Enhancing Robustness and Reliability
	Efficient Learning and Adaptation
	Robust Architecture and Design
	Optimizing Real-Time Performance

The software testing step in developing the software is time consuming. As a result, it is critical to complete testing in the shortest amount of time and with the fewest test cases possible. The Bug Reporting System (BRS) is essential during software development life cycle (SDLC) for monitoring all serious problem reports [9].

A researcher has suggested an approach for continuous validation of the quality of Machine Learning (ML) models, employing black box techniques and continuous monitoring of model performance after deployment. This validation process encompasses both traditional test datasets and real-world data from production scenarios. Instead of relying on internal or

manual testing methods, the researcher recommends exposing the models to RESTful API services to facilitate the testing process [9]

A variety of testing methodologies have been developed, which may be classified as specification driven and structure driven (which are also called as Blackbox and Whitebox) [10], or experience-based, it's important to note that the field of AI evaluation is still evolving, and new methodologies continue to emerge.

ML algorithms are utilized in various testing scenarios, with each type serving specific purposes. In black-box testing, clustering techniques benefit from all forms of ML algorithms: supervised, semi supervised, and unsupervised Learnings [11]. Additionally, regression testing heavily relies on artificial intelligence and neural networks to enhance its effectiveness. In terms of ML Model code quality, the ML developers' code review approaches can be established based on methods that have been in use for a while in software Engineering.

Regression testing is a critical component of every modern software development effort. However, aside from safety, security and moral concerns may limit the use of testing in production, necessitating additional constraints & monitors. The table.3 below is the AI Software Evaluation Approaches.

Table 3. AI Software Evaluation Approaches

Approach	Explanation
Classification-based AI software checking	A method that classifies and evaluates software code based on predetermined criteria. It assists in identifying code quality concerns, security vulnerabilities, and compliance to coding standards, providing developers with efficient and consistent feedback[12].
Testing based on Model	Intelligent learning models have been enhanced to generate detectable and testable AI test models. This provides efficient AI software evaluation and analyses of training data and test data quality, ensuring AI system reliability and efficiency.
Metamorphic testing	Metamorphic testing is a technique for confirming algorithm correctness and robustness

	by examining consistent output changes after applying specific input transformations [13].
AI software test based on Learning with a crowdsourced method	Uses the collective expertise of a wide group of individuals to check and analyse the software's behaviour, enhancing test coverage and finding potential faults more effectively.
AI software testing based on rules	Involves creating a set of specified rules or conditions that the software must meet. The testing process ensures that the programme follows these principles, guaranteeing that it meets the specified criteria and behaves as planned.

6. Discussions

The study emphasises the ethical issues that affect AI software quality. The accountability, fairness, and openness are essential characteristics that reflects the social effect of AI systems [14]. This realisation highlights the importance of ethically based software quality practises in AI development.

A communication gap was discovered between the AI and software engineering areas. This discovery underlines a key difficulty that requires bridging the terminological and viewpoint gap between these areas. Effective cooperation is required to ensure that the quality of AI software fulfils both technological and user-centric criteria.

It reveals a curious interaction between artificial intelligence and testing approaches. The use of metamorphic testing and AI algorithms into testing practises opens up new opportunities for thorough evaluation. This finding suggests a possible paradigm shift on the methods the AI systems are evaluated and verified.

Stakeholders may make informed decisions, adjust practises, and contribute to the responsible and effective adoption of AI technology across multiple domains by identifying these insights [15]. The insights culminate in a demand for dynamic, multifaceted, and ethically based ways to ensuring AI software quality corresponds with the revolutionary potential of this technology.

7. Conclusion

In conclusion, this study emphasises the crucial importance of software quality in the field of Artificial Intelligence (AI). AI technology's fast growth has resulted in dramatic transformations, changing human relationships with technology and transforming decision-making processes. It has been clearly evident during the investigation that software quality is the key in assuring the effective deployment, operation, and ethical usage of AI systems.

The study looks into the various aspects of AI software quality, from data quality and pre-processing to the critical function of model training and validation. To establish confidence and credibility in AI systems, ethical concerns, fairness, openness, and continuous progress emerge as critical components.

The necessity for smooth data interoperability and the communication gap between the AI and software engineering sectors are recognised as challenges on this trip. The introduction of testing approaches such as metamorphic testing, as well as the incorporation of AI algorithms into the testing process, provide new opportunities for improving software quality.

In future, this study would serves as a guiding light, highlighting the importance of collaboration, innovation, and constant research in the domain of AI software quality. The methodologies and practices elucidated within these pages offer a roadmap for developers, researchers, and practitioners to navigate the intricacies of AI implementation while upholding the highest benchmarks of software quality.

References

[1] Golendukhina, V., Lenarduzzi, V., & Felderer, M. (2022, May). What is software quality for AI engineers? Towards a thinning of the fog. In Proceedings of the 1st International Conference on AI Engineering: Software Engineering for AI (pp. 1-9).

[2] Siebert, J., Joeckel, L., Heidrich, J., Trendowicz, A., Nakamichi, K., Ohashi, K., ... & Aoyama, M. (2022). Construction of a quality model for machine learning systems. *Software Quality Journal*, *30*(2), 307-335.

[3] Foidl, H.; Felderer, M. Risk-based data validation in machine learning-based software systems. In Proceedings of the 3rd ACM SIGSOFT International Workshop on Machine Learning Techniques for Software Quality Evaluation, ACM, Tallinn, Estonia, 27 August 2019; pp. 13–18.

[4] Satoshi Masuda, Kohichi Ono, Toshiaki Yasue, and Nobuhiro Hosokawa. 2018. A survey of software quality for machine learning applications. In 2018 IEEE International conference on software testing, verification and validation workshops (ICSTW). IEEE, 279–284.

[5] Alex Serban, Koen van der Blom, Holger Hoos, and Joost Visser. 2020. Adoption and effects of software engineering best practices in machine learning. In Proceedings of the 14th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM). 1–12.

[6] Perkusich, M.; e Silva, L.C.; Costa, A.; Ramos, F.; Saraiva, R.; Freire, A.; Perkusich, A. Intelligent software engineering in the context of agile software development: A systematic literature review. Inf. Softw. Technol. **2020**, 119, 106241.

[7] Vinayagasundaram, B., and S. K. Srivatsa. "Software quality in artificial intelligence system." *Information Technology Journal* 6.6 (2007): 835-842.

[8] Serban, Alex, et al. "Adoption and effects of software engineering best practices in machine learning." *Proceedings of the 14th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. 2020.

[9] Cernau, L. D., Dioşan, L. S., & Șerban, C. (2022, November). A pedagogical approach in interleaving software quality concerns at an artificial intelligence course. In Proceedings of the 4th International Workshop on Education through Advanced Software Engineering and Artificial Intelligence (pp. 18-24).

[10] Shaikh, S. and Alam, I., 2022. Role of artificial intelligence in software quality assurance. In Intelligent Systems and Applications: Proceedings of the 2021 Intelligent Systems Conference (IntelliSys) Volume 2 (pp. 125-136). Springer International Publishing.

[11] Balasubramaniam, Nagadivya, et al. "Transparency and explainability of AI systems: ethical guidelines in practice." International Working Conference on Requirements Engineering: Foundation for Software Quality. Cham: Springer International Publishing, 2022.

[12] Tantithamthavorn, Chakkrit, et al. "Explainable ai for se: Challenges and future directions." IEEE Software 40.3 (2023): 29-33.

[13] Tosun, Ayse, Ayse Bener, and Resat Kale. "Ai-based software defect predictors: Applications and benefits in a case study." Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 24. No. 2. 2010.

[14] Harman, Mark. "The role of artificial intelligence in software engineering." 2012 First International Workshop on Realizing AI Synergies in Software Engineering (RAISE). IEEE, 2012.

[15] Martínez-Fernández, Silverio, et al. "Software engineering for AI-based systems: a survey." ACM Transactions on Software Engineering and Methodology (TOSEM) 31.2 (2022): 1-59.