International Journal of Engineering Trends and Technology
ISSN: 2231–5381 / https://doi.org/10.14445/22315381/IJETT-V73I9P121

Volume 73 Issue 9, 229-235, September 2025

Review Article

Machine Learning Software Component Quality: Current Status, Challenges, and Future Directions

Mohamed Abdullahi Ali[1*,2], Ng Keng Yap[1], Hazura Zulzalil[1], Novia Indriaty Admodisastro[1], Amin Arab Najafabadi[1], Jamal Abdullahi Nuh[3]

[1]*Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, 43400 Serdang, Selangor, Malaysia.*
[2]*Faculty of Engineering and Technology, Salaam University, Mogadishu, Somalia*
[3]*Faculty of Information Sciences and Engineering, Management and Science University, Shah Alam, Malaysia.*

[*]*Corresponding Author : moha.abdalla315@gmail.com*

Received: 15 April 2025 Revised: 15 August 2025 Accepted: 22 September 2025 Published: 30 September 2025

Abstract - *Traditional software is developed by writing code. As big data analytics and Artificial Intelligence (AI) technologies advanced, many Machine Learning (ML) based software and applications became widely accepted and used in people's daily lives. Such software is developed from trained data, and this behaviour differs from traditional software development. At this moment, building ML software consumes time and effort and requires knowledge of statistics and ML model training. To overcome this, several recent studies proposed building ML software through an ML software component-based method. Consequently, this approach will increase reusability and reduce development effort in ML software. Presently, there is a high demand for creating a quality model for ML software components, as traditional software component quality models cannot support specific quality aspects of ML software components. For instance, ML software component behaviour differs from conventional software components because they are built from trained data rather than being written in programming code. Thus, the ML software component quality model became essential due to their unique nature. This study offers an outline and insights for researchers to better understand the present condition of machine learning software component quality models, related challenges, future directions, and the advantages of adopting a component-based software development approach for machine learning software (i.e., machine learning software components).*

Keywords - *ML software, Quality model, ML software component.*

1. Introduction

With the rise of Big Data, Cloud Computing, and Machine Learning (ML), the field of Software Engineering (SE) has evolved from a simple calculating engine. Each time one of these transitions occurs, SE goals are modified to fit these trends, which encourages the investigation of new and reliable techniques [1]. AI and big data technologies have advanced rapidly, leading to widespread adoption of AI-based applications in daily life [2]. ML models now form the core of modern software development, with 7 million developers already using them and 9.5 million more expected to adopt them soon [4, 5]. This study defines ML software as systems built using ML models (data and algorithms). Such software increasingly impacts critical decisions like medical diagnoses and financial approvals, underscoring its societal importance [6]. Traditional software quality assurance is well-established, with proven industry practices [7]. However, these approaches often fail for ML systems, where behavior emerges from training data rather than explicit programming [8]. Unlike deductively coded traditional systems with predefined rules, ML systems learn patterns inductively, creating unique quality challenges that require new validation paradigms beyond conventional software engineering methods [9]. Significant efforts are advancing industrial AI and ML applications, yet quality evaluation and assurance remain key challenges [10]. While development support for ML systems has grown, their unique data-driven nature creates novel quality concerns. The rapid evolution of AI technologies has intensified demand for high-quality AI software, but traditional quality approaches often prove inadequate for these distinctive systems [2]. ML software development is complex and expertise-intensive [11]. Component-based approaches help address challenges, but data-driven ML components employing various techniques introduce unique quality concerns beyond traditional software [12-16]. ML components possess unique quality traits like accuracy and fairness, requiring distinct measurement methods from traditional software. Different development paradigms demand tailored quality models to address their specific characteristics [17, 18]. Nonetheless, there exist scholarly studies on the quality of AI software [19], surveys about the quality of ML software [20], and Systematic Mapping Studies (SMS) pertaining to the quality of software for AI-based systems, components, and software. Nevertheless, none of the existing literature examines models addressing the quality of machine learning software components, the challenges they present, or the benefits of employing a component-based approach for machine learning software. The primary aim of this work is to elucidate the constraints established by the prior studies. Section 2 presents the relevant contextual information of the study. Section 3 provides a comprehensive explanation of the research methodology employed in this study. Section 4 of the study contains a summary of the results and supplementary analysis. Section 5 of the study examines the conclusions drawn from the evidence and contemplates alternatives for future research.

2. Literature Review

2.1. Component-Based Software Engineering (CBSE)

According to Crnkovic [22], Component-based Software engineering involves developing systems as an assembly of parts (components), developing parts as reusable entities, and customising and replacing parts to maintain and upgrade systems. Component-Based Development (CBD), as defined by Lau and Wang [23], utilises pre-built software elements or components to assemble systems. Instead of being created as a single entity, a system is composed of smaller components. Such a method reduces production costs by constructing a system from pre-existing parts rather than building it from scratch. The ability to reuse components across different systems also facilitates software reuse. As a result, CBD promises the advantages of more reuse, cost-effectiveness, and shorter time to market.

2.2. Software Quality

According to the IEEE [24], software quality is the level to which a system, part, or process meets certain standards. Software engineering is a subject that places a strong emphasis on developing high-quality software products. Hence, the significance of software quality is obvious. Over the last three decades, software engineering researchers have paid close attention to software quality [25, 26], focusing on the market value of software products [27, 28]. It takes a lot of work to create high-quality products since product developers must also cope with difficulties, including competitors, quality problems, and client satisfaction [29]. As noted in [27, 30], an increasing number of firms and organizations are imposing requirements on both the quality of the processes utilized in software development and the quality of the products they acquire or create.

2.3. Software Quality Model

Software quality models are defined by ISO/IEC IS 9126-1 [31] and consist of a collection of qualities and the relationships between them. These attributes form the basis for quality evaluations and the establishment of quality standards. Quality models have been established to delineate the essential components, referred to as characteristics, and their corresponding sub-factors, known as sub-characteristics, for the assessment of software quality. Each sub-factor is assigned a specific set of measurements for the evaluation. Software quality models are primarily categorized into basic and customized quality models.

Because of their hierarchical structure, the Basic Models are open to review and improvement and can be used for any type of software product. Look at these six: Various models and standards have been developed for international software, including McCall et al., 1977; Boehm et al., 1978; FURPS Model, 1992; Dromey Model, 1995 [32]; and ISO 9126-1 model, 2001 [31]. In 2003, ISO/IEC 9126-2 was issued for external metrics, and in 2004, for internal metrics and quality in use, ISO/IEC 9126-4 was issued. Taking into account feedback from previous models, the ISO-9126 model specifies criteria for assessing software quality. The 2007 revision of the ISO 25010 concept, as published in ISO/IEC CD 25010 [33], included certain changes. ISO 25010, which stands for "Software Engineering—Software Product Quality Requirements and Evaluation," is an acronym for software engineering.

Tailored Quality Models were first demonstrated by the Bertoa, Alvaro, and Rawashdesh models [34-36]. Their primary characteristic is that they are customized for a particular application area, and feature adjustments can differ when compared to a universal model. These models were created in response to the need for high-quality models to evaluate specific components in organizations and the software industry. The latest software development, ISO 9126, is one example of how they are built utilizing sub-factors that are either added to or modified from Basic Models to satisfy the requirements of specific domains or specialized applications.

2.4. Intersection between AI and SE

The "AI spells" dominate SE research and communities [37]. While AI is defined as the process of giving machines intelligence, SE is a practical engineering topic and is defined as the process of defining, developing, and deploying systems. Software and engineering are two terms that SE made [38]. Engineering pertains to the methodologies employed in design and building to ascertain the cost of effective solutions, whereas software denotes programs that integrate instructions to provide required functionality. A systematic methodology for the design, development, implementation, and maintenance of a software system constitutes another definition of Software Engineering (SE) [39].

The SE community has adopted and tailored numerous valuable AI-related approaches, methodologies, and procedures. These AI algorithms and methodologies influence nearly every software engineering action. SE for ML involves the development, design, and upkeep of software systems that incorporate machine learning capabilities. Academic researchers are currently engaged in the examination of differences between ML software and conventional software. Additionally, they are putting forth novel methodologies and instruments to address these differences.

ML for SE involves the use or customisation of AI techniques to various software engineering activities [40]. Software defect prediction [41], code smell detection [42], reusability metric prediction, and project cost estimation [43] represent a subset of these tasks; however, they are not exhaustive. Software engineers can enhance the velocity and efficacy of program development by utilising machine learning models derived from software engineering data, encompassing source code, requirement specifications, and test cases.

2.5. Related Works

This part discusses past studies on the quality of AI software and systems, comparing the study's goals, scope, and conclusions.

Gezici and Tarhan conducted a comprehensive assessment of 29 papers concerning the quality of AI software. To identify contemporary quality models for AI-driven software quality, they examined quality attributes, their assurance, challenges, and solutions from 1988 to 2020. Researchers aim to evaluate the efficacy of AI software comprehensively.

A total of seventy-two research papers pertaining to the creation, maintenance, issues, and solutions of ML-based software systems were reviewed by Lwakatare et al. [21]. Since the software under review is ML-based, this study does not place an emphasis on product quality.

Riccio et al. [54] mapped out all 70 papers that were written on functional testing for Machine Learning Systems (MLS). Reference [54] talks about ML software testing, problems, and solutions, while this work is mostly about AI-based software, systems, and parts. Even though the research subjects are testing-related, this study is mostly about the quality of ML software components. Because of this, [54]'s contributions and consequences are very different from those of this work.

Habibullah et al. [55] present the most comprehensive synthesis of quality indicators among all the evaluated publications. The collection of QAs was established from conversations with professionals in the domain of designing ML-enabled solutions. The authors gathered 37 quality attributes (non-functional needs of the system) pertinent to product operation, revision, and transition. Indykov et al. [56] proposed a quality model for machine learning-based systems. The analysis showed 11 prominent quality attributes, 16 applicable architectural techniques, and 85 probable quality trade-offs. The outcomes organize current research in the building of ML-enabled system architectures.

Prior research has enhanced our understanding of AI-based software scenarios and instances; nevertheless, it has not elucidated the quality of machine learning software components in academic, industrial, and experimental contexts.

3. Research Methodology

To accomplish the study objective, it is essential to develop a well-structured plan that outlines the specific sequence of tasks to be undertaken. This section outlines the sequential procedures undertaken to conduct the research.

A comprehensive literature review on machine learning software component quality was carried out in the IEEE Xplore, Scopus, Google Scholar, and ACM Digital Library databases. The search included both conference and journal publications written in English. The literature search was conducted with three research aims in consideration. Firstly, to determine the benefits of using component-based software development in the machine learning software context. Secondly, to assess the status of machine learning software component quality. And finally, to examine the problems associated with it.

4. Results and Discussion

4.1. Benefits of the Adoption Component-based Approach for ML Software Development

Traditional SE practices, such as encapsulation and modular design, have demonstrated the value of clear abstraction boundaries [44]. However, several difficulties have been encountered when developing ML software from scratch, such as the need for experts, time consumption, complexity, and cost, rather than reusing a model that addresses these difficulties. Implementing ML from scratch into applications is difficult, time-consuming, and necessitates expert knowledge [11]. Furthermore, ML has seen widespread adoption in a wide range of real-world problem domains, from business to healthcare to agriculture [45]. However, developing effective ML solutions necessitates highly specialized experts who are proficient in both statistics and programming. Furthermore, starting ML from scratch requires more training cost and time than utilizing an existing model [11].

In order to overcome these challenges, a number of earlier studies have employed Component-Based Development (CBD) strategies; these studies centre on the idea of componentizing Machine Learning (ML) models and Artificial Intelligence (AI) neural networks through the reuse of component-based approaches; the goal of this strategy is to shorten the time it takes to develop ML software and do away with the need for software engineers to have deep knowledge of ML algorithms and models [12-14]. Additionally, it aims to reduce the complexity associated with software maintenance. By assembling a system from pre-built software units or components rather than from scratch, the CBD approach aims to reduce production costs [11]. The software industry has long wished for increased reuse, lower production costs, and quicker time to market, which is why CBD makes these promises. These components are easily reusable across several applications [12]. The benefits of an ML software component are as follows: The concept of reusability in software engineering refers to the ability of a software component to be reused without requiring knowledge of its underlying implementation. Reusability: This is achieved by the creation of a coherent and loosely linked module that can be easily integrated into different systems. Plug-ability: The ability of the software components to be quickly replaced. It offers pluggability both when running and when not.

4.2. Current Status and Challenges for ML Software Component Quality

There are enormous efforts to improve the quality of ML software, such as [10, 47, 48]. These studies only contain a quality characteristic from the ML software perspective. Nevertheless, as ML software components own both ML software and software component characteristics, these studies are irrelevant and inappropriate for assessing ML software components due to the lack of software component quality characteristics.

On the other hand, many studies on software component quality have been conducted [35, 49-51], but dealing with ML software components differs from dealing with conventional software components due to the involvement of training data [3]. Furthermore, because their functionality is derived from data, AI-based software components, particularly ML-based software components, present significant issues for quality assurance [16]. In fact, there are presently multiple software development paradigms [17]. Due to each software paradigm's uniqueness, a specific software quality model must be developed for it. According to this, a study by [18] stated that new characteristics might be added, and existing definitions may be modified when considering the nature of the product itself. So far, no study has been presented on the current quality status and problems with ML software components. As a quality standard, this ideal model should be used for both software parts and machine learning apps. Therefore, Gharib et al. [52] stressed how important it is to make a quality model for machine learning software components that includes a quality feature of these components. The product's overall quality and functionality will diminish if these components are unsuccessful [53]. Traditional systems and software's dependability is judged by several quality indicators. However, it is hard to use or modify these criteria to evaluate the quality of AI-based pieces because they are not usually good enough for directly analysing ML software and systems [3]. Special features of AI software systems mean that new quality models and measures are needed for software that uses AI [19, 48].

5. Conclusion

Conventional software is constructed by writing code. Many ML-based software and applications have received much interest and use in people's daily lives. However, developing such ML software requires time, effort, and training in both statistics and ML. To address this, numerous studies already in progress have been inspired to create ML software using ML software components. The behaviour of ML software components is distinct from that of conventional software components. Instead of being explicitly programmed, such components are constructed from trained data. However, because of their unique characteristics, ML software components do not adhere to conventional software component quality models and practices.

This study analysed both the shortcomings and advantages of employing a component-based methodology for the development of machine learning software, along with the present state of the quality of machine learning software components. This review revealed only papers focused on the standpoint of machine learning software quality. The functionality of ML-based software components is contingent upon data, presenting novel issues for quality assurance [16]. At present, various software development paradigms are in use. Each software paradigm's distinctiveness necessitates the creation of a tailored software quality model. Similarly, another study [18] asserted that, depending on the product's characteristics, new attributes may be incorporated, and existing definitions may be modified. To the best of our knowledge, no quality model exists for machine learning software at the component level. Consequently, a quality model for machine learning software components must be established by examining quality characteristics and metrics pertinent to their nature and functionality. Finally, the practicality of the established quality model must be assessed via expert evaluation and case study.

References

[1] Saleema Amershi et al., "Software Engineering for Machine Learning: A Case Study," *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, Montreal, QC, Canada, pp. 291-300, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[2] Chuanqi Tao, Jerry Gao, and Tiexin Wang, "Testing and Quality Validation for AI Software-Perspectives, Issues, and Practices," *IEEE Access*, vol. 7, pp. 120164-120175, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[3] Fuyuki Ishikawa, "Concepts in Quality Assessment for Machine Learning-From Test Data to Arguments," *International Conference on Conceptual Modeling*, Xi'an, China, pp. 536-544, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[4] Tammo Rukat et al., "Towards Automated ML Model Monitoring: Measure, Improve and Quantify Data Quality," *Amazon Science: MLSys 2020 Workshop on MLOps Systems*, pp. 1-3, 2020. [Google Scholar] [Publisher Link]

[5] Worldwide Developer Population Report, Global Developer Population Numbers and Statistics: 2025 through 2030, Evans Data Corporation, 2025. [Online]. Available: https://evansdata.com/reports/viewRelease.php?reportID=9/

[6] Jennifer Horkoff, "Non-Functional Requirements for Machine Learning: Challenges and New Directions," *2019 IEEE 27th International Requirements Engineering Conference (RE)*, Jeju, Korea (South), pp. 386-391, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[7] Nayan B. Ruparelia, "Software Development Lifecycle Models," *ACM SIGSOFT Software Engineering Notes*, vol. 35, no. 3, pp. 8-13, 2010. [CrossRef] [Google Scholar] [Publisher Link]

[8] Hiroshi Kuwajima, and Fuyuki Ishikawa, "Adapting Square for Quality Assessment of Artificial Intelligence Systems," *2019 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, Berlin, Germany, pp. 13-18, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[9] Foutse Khomh et al., "Software Engineering for Machine-Learning Applications: The Road Ahead," *IEEE Software*, vol. 35, no. 5, pp. 81-84, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[10] Gaku Fujii et al., "Guidelines for Quality Assurance of Machine Learning-Based Artificial Intelligence," *International Journal of Software Engineering and Knowledge Engineering*, vol. 30, no. 11n12, pp. 1589-1606, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[11] Marc-Oliver Pahl, and Markus Loipfinger, "Machine Learning as a Reusable Microservice," *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, Taipei, Taiwan, pp. 1-7, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[12] Uzair Ahmad et al., "CompoNet: Programmatically Embedding Neural Networks into AI Applications as Software Component," *19th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2007)*, Patras, Greece, vol. 1, pp. 194-201, 2007. [CrossRef] [Google Scholar] [Publisher Link]

[13] Sundaravelpandian Singaravel, Philipp Geyer, and Johan Suykens, "Component-Based Machine Learning Modelling Approach for Design Stage Building Energy Prediction: Weather Conditions and Size," *Proceedings of Building Simulation 2017: 15th Conference of IBPSA*, San Francisco, CA, USA, pp. 2617-2626, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[14] Sundaravelpandian Singaravel, Johan Suykens, and Philipp Geyer, "Deep-Learning Neural-Network Architectures and Methods: Using Component-Based Models in Building-Design Energy Prediction," *Advanced Engineering Informatics*, vol. 38, pp. 81-90, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[15] Julien Siebert et al., "Towards Guidelines for Assessing Qualities of Machine Learning Systems," *International Conference on the Quality of Information and Communications Technology*, Faro, Portugal, pp. 17-31, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[16] Michael Kläs et al., "Using Complementary Risk Acceptance Criteria to Structure Assurance Cases for Safety-Critical AI Components," *Proceedings of the Workshop on Artificial Intelligence Safety: Co-Located with the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI 2021)*, pp. 1-7, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[17] Toufik Marir et al., "QM4MAS: A Quality Model for Multi-Agent Systems," *International Journal of Computer Applications in Technology*, vol. 54, no. 4, pp. 297-310, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[18] Pere Botella et al., *Towards a Quality Model for the Selection of ERP Systems*, Component-Based Software Quality, pp. 225-245, Springer, Berlin, Heidelberg, 2003. [CrossRef] [Google Scholar] [Publisher Link]

[19] Bahar Gezici, and Ayça Kolukısa Tarhan, "Systematic Literature Review on Software Quality for AI-Based Software," *Empirical Software Engineering*, vol. 27, no. 3 2022. [CrossRef] [Google Scholar] [Publisher Link]

[20] Satoshi Masuda et al., "A Survey of Software Quality for Machine Learning Applications," *2018 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, Västerås, Sweden, pp. 279-284, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[21] Lucy Ellen Lwakatare et al., "Large-Scale Machine Learning Systems in Real-World Industrial Settings: A Review of Challenges and Solutions," *Information and Software Technology*, vol. 127, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[22] Ivica Crnkovic, "Component-Based Software Engineering-New Challenges in Software Development," *Software Focus*, vol. 2, no. 4, pp. 127-133, 2001. [CrossRef] [Google Scholar] [Publisher Link]

[23] Kung-Kiu Lau, "Software Component Models," *Proceedings of the 28th International Conference on Software Engineering*, Shanghai, China, pp. 1081-1082, 2006. [CrossRef] [Google Scholar] [Publisher Link]

[24] "IEEE Standard Glossary of Software Engineering Terminology," *IEEE Std 610.12-1990*, pp. 1-84, 1990. [CrossRef] [Publisher Link]

[25] Dag I.K. Sjoberg, Tore Dyba, and Magne Jorgensen, "The Future of Empirical Methods in Software Engineering Research," *Future of Software Engineering (FOSE '07)*, Minneapolis, MN, USA, pp. 358-378, 2007. [CrossRef] [Google Scholar] [Publisher Link]

[26] Niklaus Wirth, "A Brief History of Software Engineering," *IEEE Annals of the History of Computing*, vol. 30, no. 3, pp. 32-39, 2008. [CrossRef] [Google Scholar] [Publisher Link]

[27] Kumi Jinzenji et al., "An Experience Report for Software Quality Evaluation in Highly Iterative Development Methodology Using Traditional Metrics," *2013 IEEE 24th International Symposium on Software Reliability Engineering (ISSRE)*, Pasadena, CA, USA, pp. 310-319, 2013. [CrossRef] [Google Scholar] [Publisher Link]
[28] Aiman M. Solyman, Osama A. Ibrahim, and Arafat Abdulgader Mohammed Elhag, "Project Management and Software Quality Control Method for Small and Medium Enterprise," *2015 International Conference on Computing, Control, Networking, Electronics and Embedded Systems Engineering (ICCNEEE)*, Khartoum, Sudan, pp. 123-128, 2015. [CrossRef] [Google Scholar] [Publisher Link]
[29] Wilder Perdomo, Julia Prior, and John Leaney, "How do Colombian Software Companies Evaluate Software Product Quality?," *Proceedings of the 30th International Workshop on Software Measurement and the 15th International Conference on Software Process and Product Measurement (IWSM Mensura 2020)*, Mexico City, Mexico, pp. 1-17, 2020. [Google Scholar] [Publisher Link]
[30] Karina Curcio et al., "An Analysis of the Factors Determining Software Product Quality: A Comparative Study," *Computer Standards & Interfaces*, vol. 48, pp. 10-18, 2016. [CrossRef] [Google Scholar] [Publisher Link]
[31] "ISO/IEC 9126-1:2001, Software Engineering-Product Quality," *International Organization for Standardization*, 2001. [Publisher Link]
[32] R.G. Dromey, "A Model for Software Product Quality," *IEEE Transactions on Software Engineering*, vol. 21, no. 2, pp. 146-162, 1995. [CrossRef] [Google Scholar] [Publisher Link]
[33] "ISO/IEC 25010: 2011: Systems and Software Engineering-Systems and Software Quality Requirements and Evaluation (SQuaRE)-System and Software Quality Models," *International Organization for Standardization*, 2011. [Google Scholar] [Publisher Link]
[34] Manuel F. Bertoa, and Antonio Vallecillo, "Quality Attributes for COTS Components," *I+ D Computacion*, vol. 1, no. 2, pp. 128-143, 2002. [Google Scholar]
[35] Alexandre Alvaro, Eduardo Santana De Almeida, and Silvio Lemos Meira, "A Software Component Quality Model: A Preliminary Evaluation," *32nd EUROMICRO Conference on Software Engineering and Advanced Applications (EUROMICRO'06)*, Cavtat, Croatia, pp. 28-37, 2006. [CrossRef] [Google Scholar] [Publisher Link]
[36] Adnan Rawashdeh, and Bassem Matalkah, "A New Software Quality Model for Evaluating COTS Components," *Journal of Computer Science*, vol. 2, no. 4, pp. 373-381, 2006. [CrossRef] [Google Scholar] [Publisher Link]
[37] Mark Harman, "The Role of Artificial Intelligence in Software Engineering," *2012 First International Workshop on Realizing AI Synergies in Software Engineering (RAISE)*, Zurich, Switzerland, pp. 1-6, 2012. [CrossRef] [Google Scholar] [Publisher Link]
[38] Mohammad Shehab et al., "(AIAM2019) Artificial Intelligence in Software Engineering and Inverse," *International Journal of Computer Integrated Manufacturing*, vol. 33, no. 10-11, pp. 1129-1144, 2020. [CrossRef] [Google Scholar] [Publisher Link]
[39] Emily Winter, Steve Forshaw, and Maria Angela Ferrario, "Measuring Human Values in Software Engineering," *ESEM '18: Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, Oulu, Finland, pp. 1-4. 2018. [CrossRef] [Google Scholar] [Publisher Link]
[40] Xie, Tao. "Intelligent Software Engineering: Synergy between AI and Software Engineering," *ISEC '18: Proceedings of the 11th Innovations in Software Engineering Conference*, Hyderabad, India, 2018. [CrossRef] [Google Scholar] [Publisher Link]
[41] Ruchika Malhotra, "A Systematic Review of Machine Learning Techniques for Software Fault Prediction," *Applied Soft Computing*, vol. 27, pp. 504-518, 2015. [CrossRef] [Google Scholar] [Publisher Link]
[42] Muhammad Ilyas Azeem et al., "Machine Learning Techniques for Code Smell Detection: A Systematic Literature Review and Meta-Analysis," *Information and Software Technology*, vol. 108, pp. 115-138, 2019. [CrossRef] [Google Scholar] [Publisher Link]
[43] Rasmita Panigrahi et al., "Software Reusability Metrics Prediction and Cost Estimation by Using Machine Learning Algorithms," *International Journal of Knowledge-based and Intelligent Engineering Systems*, vol. 23, no. 4, pp. 317-328, 2019. [CrossRef] [Google Scholar] [Publisher Link]
[44] Anders Arpteg et al., "Software Engineering Challenges of Deep Learning," *2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, Prague, Czech Republic, pp. 50-59, 2018. [CrossRef] [Google Scholar] [Publisher Link]
[45] Doris Jung-Lin Lee, and Stephen Macke, "A Human-in-the-Loop Perspective on AutoML: Milestones and the Road Ahead," *IEEE Data Engineering Bulletin*, pp. 59-70, 2020. [Google Scholar] [Publisher Link]
[46] Valentina Lenarduzzi et al., "Software Quality for AI: Where We are Now?," *International Conference on Software Quality*, Vienna, Austria, vol. 1, pp. 43-53, 2021. [CrossRef] [Google Scholar] [Publisher Link]
[47] Shin Nakajima, "Quality Evaluation Assurance Levels for Deep Neural Networks Software," *2019 International Conference on Technologies and Applications of Artificial Intelligence (TAAI)*, Kaohsiung, Taiwan, pp. 1-6, 2019. [CrossRef] [Google Scholar] [Publisher Link]
[48] Koji Nakamichi et al., "Requirements-Driven Method to Determine Quality Characteristics and Measurements for Machine Learning Software and its Evaluation," *2020 IEEE 28th International Requirements Engineering Conference (RE)*, Zurich, Switzerland, pp. 260-270, 2020. [CrossRef] [Google Scholar] [Publisher Link]
[49] Manuel F. Bertoa, José M. Troya, and Antonio Vallecillo, "Measuring the Usability of Software Components," *Journal of Systems and Software*, vol. 79, no. 3, pp. 427-439, 2006. [CrossRef] [Google Scholar] [Publisher Link]

[50] Rikard Land, Alexandre Alvaro, and Ivica Crnkovic, "Towards Efficient Software Component Evaluation: An Examination of Component Selection and Certification," *2008 34th Euromicro Conference Software Engineering and Advanced Applications*, Parma, Italy, pp. 274-281, 2008. [CrossRef] [Google Scholar] [Publisher Link]

[51] Sharma, Arun, Rajesh Kumar, and P.S. Grover, "Estimation of Quality for Software Components: An Empirical Approach," *ACM SIGSOFT Software Engineering Notes*, vol. 33, no. 6, pp. 1-10, 2008. [CrossRef] [Google Scholar] [Publisher Link]

[52] Mohamad Gharib et al., "On the Safety of Automotive Systems Incorporating Machine Learning Based Components: A Position Paper," *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, Luxembourg, Luxembourg, pp. 271-274, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[53] Hasan Kahtan, Nordin Abu Bakar, and Rosmawati Nordin, "Reviewing the Challenges of Security Features in Component Based Software Development Models," *2012 IEEE Symposium on E-Learning, E-Management and E-Services*, Kuala Lumpur, Malaysia, pp. 1-6, 2012. [CrossRef] [Google Scholar] [Publisher Link]

[54] Vincenzo Riccio et al., "Testing Machine Learning Based Systems: A Systematic Mapping," *Empirical Software Engineering*, vol. 25, no. 6, pp. 5193-5254, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[55] Khan Mohammad Habibullah, Gregory Gay, and Jennifer Horkoff, "Non-Functional Requirements for Machine Learning: Understanding Current Use and Challenges Among Practitioners," *Requirements Engineering*, vol. 28, no. 2, pp. 283-316, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[56] Vladislav Indykov, Daniel Strüber, and Rebekka Wohlrab, "Architectural Tactics to Achieve Quality Attributes of Machine-Learning-Enabled Systems: A Systematic Literature Review," *Journal of Systems and Software*, vol. 223, pp. 1-20, 2025. [CrossRef] [Google Scholar] [Publisher Link]