e-Informatica Software Engineering Journal
Volume 20, Issue 1, pages: 1–18
DOI: 10.37190/e-Inf260102

BibTeX

A Defect Classification Framework for AI-Based Software Systems (AIODC)

Mohammed Alannsary*
*Corresponding author: alannsary@hotmail.com

Article info

Dataset link: https://doi.org/10.5281/zenodo.17843077

Keywords:
Software testing
analysis and verification
Software Quality
AI and knowledge based
software engineering

Submitted: 29 Aug. 2025
Revised: 10 Dec. 2025
Accepted: 10 Dec. 2025
Available online: 12 Dec 2025

Abstract

Context: Artificial Intelligence (AI) is increasingly integrated into critical domains, making defect analysis essential to ensure system quality and reliability. Current defect classification frameworks do not adequately address the unique properties of AI systems.
Objective: This paper proposes AIODC, a defect classification framework inspired by the Orthogonal Defect Classification (ODC) that incorporates AI-specific characteristics.
Method: The framework extends ODC by introducing three new attributes – Data, Learning, and Thinking – and adds a "Catastrophic" severity level to account for risks associated with AI. Additionally, it modifies impact mapping utilizing AI/AIP quality models. The methodology was validated through a case study that examined 42 actual Keras defects.
Results: This study demonstrated the feasibility of modifying ODC for AI systems to classify its defects. The case study indicated that defects occurring during the Learning phase are the most prevalent and were significantly linked to high severity, whereas defects in the Thinking phase primarily impact trustworthiness and accuracy.
Conclusions: The results affirm the practicality and significance of AIODC in identifying high-risk defect categories, thus facilitating more focused and effective quality assurance strategies in AI-driven software systems.

1. Introduction

The domain of software quality constitutes a specialized branch within the field of software engineering that rigorously investigates, evaluates, quantifies, and enhances the quality attributes associated with a software product [1–3]. In general terms, a software product is regarded as possessing a high level of quality if it is devoid of, or exhibits a minimal number of, issues that originate from the software itself and result in limited adverse effects on its users.

Defect analysis is the branch of software engineering that analyzes and resolves discovered defects while developing the software or using it. When it comes to software quality improvement, defect analysis is known to be a major contributor. Several defect analysis techniques and methods are available such as defect classification, where found defects are analyzed and grouped based on their characteristics. One of the main and well-known defect classification methods is the Orthogonal Defect Classification (ODC) approach [1, 4–6].

Implementing defect analysis has been successful in several types of software, such as web applications, Software as a Service (SaaS), and traditional software, yet defect analysis for Artificial Intelligence (AI) systems needs to be addressed due to the additional and different characteristics that AI systems have over traditional systems, web applications, and SaaS.

AI plays a transformative role in improving several fields of life, such as medicine, engineering, economy, social etc. The main advantage of AI is that it helps and aids in data analysis, prediction, and decision making. AI solutions are built mainly using software, therefore it is crucial to evaluate its quality to assure its correctness. One could argue that software written to produce AI does not differ from traditional software, web applications, or SaaS. However, each of the aforementioned software types – including AI systems – has its own attributes and characteristics, and needs to be catered to accordingly [7–10].

The quality of the AI system encompasses both the quality of AI as a universal yet specific entity and the quality of AI Platforms (AIP) which are the software and hardware platforms that facilitate its implementation [11]. Defining attributes and characteristics of an AI system may be troublesome, it can learn, adapt, and make decisions based on complex unstructured data and provides solutions, which fundamentally emulates human cognitive capabilities [12], while other systems are based on static rules and predetermined algorithms, works with structured data, and follows established processes. Therefore, there is a need to customize defect analysis techniques to cater for AI systems.

Despite numerous studies utilizing ODC in traditional, web, and cloud-based software, there remains a lack of a well-defined defect classification framework that addresses the unique challenges of AI systems. Current methodologies do not adequately account for the dynamic features of AI – such as data dependency, continuous learning, and autonomous decision-making – which alter the nature and propagation of defects. This research fills that gap by introducing AIODC, an adaptation of ODC that incorporates AI-specific attributes, a domain-sensitive severity scale, and quality impact mapping. The objective of the study is to assess the applicability of AIODC and to uncover defect patterns that are unique to AI-driven systems through an empirical case study.

The remainder of this work is organized as follows: the next Section presents background and related work. Section 3 discusses ODC, and how it was modified and/or adapted to different types of software. Section 4 describes the proposed framework. In Section 5 a case study is provided to show the applicability of the framework. Section 7 contains the conclusion and prospective.

2. Background and related work

Ensuring the quality of AI requires recognizing its unique attributes, which differ from those of conventional software, web applications, and SaaS. AI typically integrates novel algorithms and functions, evolves through continuous implementation, and enables developers to use diverse datasets for system training [12].

Consequently, existing methodologies and frameworks for assessing the quality of conventional, web-based software, and SaaS require adaptation to account for AI's distinctive characteristics. Reliability is particularly critical when AI decisions may affect human lives [7].

AI is now a crucial component of the software development life cycle. Its integration marks a significant shift in how software is designed, built, tested, maintained, and deployed.

The growing use of AI across diverse sectors is driving a transition away from conventional development frameworks. Developers can now leverage AI to refine processes, reduce errors, and build more sophisticated systems using machine-learning algorithms, natural-language processing, and data analytics. This transition underscores the necessity for organizations to embrace and incorporate AI into various stages of the software development life-cycle (SDLC) [13].

Defect analysis for AI software remains in its early stages, even though AI has already been applied to classify and detect defects in domains such as product inspection, image processing, and sound analysis. Moreover, several studies have explored using AI as a tool within software engineering itself–for example, in code generation, software testing, defect detection, and defect classification [14–23]. However, there has not been a defect analysis approach developed for AI systems specifically, nor was one introduced in the literature.

Vinayagasundaram and Srivatsa [24] proposed metrics for evaluating the quality of AI software and noted that its components are reusable. Their architecture comprised four layers–task specification, problem solver, domain, and adapter. The main structure is segmented into sub components in a layered manner, allowing for the addition of more layers that can alter the system's behavior. Several metrics were used to measure components quality, such as: depth of inheritance, complexity level, and source code.

Tao et al. [25] highlighted that AI software's unique characteristics introduce new challenges for validating software and system quality. Their study discussed how these challenges affect the functional aspects of AI software. In addition, they articulated the comprehension of testing AI software in relation to new features and requirements. Moreover, they presented the current classifications of AI software testing and examined different testing strategies. Furthermore, they provided an illustration of test quality evaluation and criteria analysis. Finally, a practical examination of quality validation for an image recognition system was executed through a metamorphic testing technique.

Chillarege et al. [4] proposed Orthogonal Defect Classification (ODC) – a methodology that provides in-process feedback to developers by deriving defect signatures from the development process. Generally, ODC serves as a technique for categorizing defects to identify which phase of the software development life-cycle requires focus. Defects are categorized based on several dimensions: phase of discovery, defect type, severity, source, trigger, and impact.

Ma and Tian [5] developed a method for classifying and analyzing web errors through integrating ODC into the web environment. Attributes pertinent to the web context were selected from web access logs to serve as defect types within ODC. This approach facilitated the identification of issues and offered valuable feedback to the web applications development team.

Alannsary and Tian [6] proposed a framework for defect analysis tailored for a cloud-based SaaS. The framework was influenced by the original ODC model and acknowledges the distinctive features of SaaS. It was structured into three phases: data source analysis, classification, and results analysis. The entries from the web server log file are examined to pinpoint errors based on the values of the ”Protocol status” field, the detected defects are then classified according to six defect attributes specifically designed for the cloud context. Successful implementation of one-way and two-way analyses has yielded improved insights into the identified defects and their solutions. A case study was provided to validate the effectiveness of the proposed framework.

Kharchenko et al. [11] aimed at establishing and showcasing the implementation of quality models for AI, AIP, and AIS, based on the definition and organization of characteristics. The principles underlying the development of AI quality models and their sequence are thoroughly justified. Approaches for articulating definitions of AIS characteristics, as well as methods for depicting dependencies and hierarchies of characteristics, were discussed. In addition, they proposed definitions and harmonization strategies for the hierarchical relationships among 46 characteristics of AI and AIP. Moreover, the quality models pertaining to AI, AIP, and AIS are detailed, with a focus on the most critical characteristics. Finally, two examples of AIS quality models were elaborated upon. It is worth mentioning that they pointed out that experts have the liberty of including or eliminating characteristics based on the context of the AI system.

Morovati et al. [26] examined the reproducibility and verifiability of bugs found in Machine Learning (ML) systems, which is part of AI systems, highlighting the key factors associated with each aspect. Subsequently, they delve into the difficulties of creating a benchmark for bugs in ML software systems and introduce a bug dataset to be used as a benchmark called "defect4ML". The dataset meets all the criteria of a standard benchmark, including relevance, reproducibility, fairness, verifiability, and usability. The dataset contained 100 bugs reported by ML developers on platforms such as GitHub and Stack Overflow, utilizing two of the most widely used ML frameworks: TensorFlow and Keras. Cross-platform defect reporting (for instance, GitHub compared to Stack Overflow) poses a risk of duplication, since developers frequently repost issues [27]. The process of duplication continues to be difficult in the absence of issue-ID cross-referencing.

While earlier research has focused on predictive defect detection–such as deep-learning or CodeBERT-based models that anticipate defect-prone components–these approaches differ fundamentally from the proposed AIODC framework. Unlike these approaches, AIODC does not seek to forecast future defects; rather, it classifies defects that have already been identified to ascertain their origin, severity, and impact on quality. This differentiation establishes AIODC as a supplementary tool that improves diagnostic comprehension rather than prediction, effectively connecting abstract AI quality models with practical defect analysis processes.

3. Orthogonal defect classification (ODC)

Since its first debut, ODC played a noticeable role in defect analysis for traditional software. Several researchers referenced the concept, and some adapted it to different fields. With the introduction of the Web and cloud computing, Software has changed in the way it is delivered and used. Therefore, This section describes how the concept was modified and/or adapted to other types of Software.

3.1. Original ODC

In traditional software and systems, Chillarege et al. [4] specified a cause effect relationship in ODC. There were two cause attributes: defect type and defect trigger, and several effect attributes such as severity, reliability growth, and impact areas. Table 1 shows the cause and effect relationship of the original ODC.

In addition, in the defect type attribute, defects were classified and associated with a phase in the development processes as shown below.

Function associated with the Design phase.
Interface associated with the Low Level Design phase.

Table 1. Original ODC cause-effect relationships [4], adapted as AIODC's foundation

CAUSE	⟺	EFFECT
Defect Type (Development Process)		Severity Impact areas (CUPRIMD) Reliability Growth
Defect Trigger (Verification Process)		Defect Density Rework on Fixes Etc.

Checking associated with the Low Level Design or Code phase.
Assignment associated with the Code phase.
Timing/Serialization associated with the Low Level Design phase.
Build/Package/Merge associated with the Library Tools phase.
Documentation associated with the Publications phase.
Algorithm associated with theLow Level Design phase.

Moreover, defects were classified in regards to the impact areas based on IBM's CUPRIMD [28]. Table 2 shows the attributes and metric areas associated with it.

Table 2. Traditional impact areas (IBM CUPRIMD) [4]
– replaced by AI-specific characteristics in AIODC

Attribute	Metric Areas
Capability	Functionality delivered versus requirements Volume of function to deliver
Usability	Ease of learning important tasks Ease of completing a task Intuitiveness
Performance	Transaction throughput Response time to enquiry Size of machine needed to run the product
Reliability	Mean time between failures Number of defects Severity of defects Severity/impact of failures
Installibility	Ease of making product available for use Time to complete installation Skill level needed by installer
Maintainability	Ease of problem diagnosis Ease of fixing problem correctly
Documentation	Ease of understanding Ease of finding relevant information Completeness of information

3.2. Web ODC

Ma and Tian [5] adapted the ODC concept to the web environment through identifying web error attributes. The Web ODC attributes are listed below.

1. Response code

2. File type
3. Referrer type
4. Time period

3.3. Cloud ODC

Due to the special characteristics of SaaS, such as multi-tenancy and isolation, Alannsary and Tian [6] adapted the ODC concept to a SaaS running in the Cloud. In the cloud ODC framework one attribute (Layer affected) was added to the Original ODC attributes as depicted in Figure 1 including the newly added one which is **bold**.

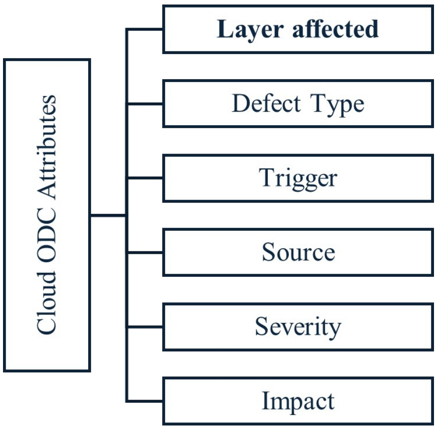

Figure 1. Cloud ODC attribute schema [6] – baseline for AIODC's extended structure

In addition, two new defect types (Isolation, IaaS/Paas) were introduced to the original ODC defect types as shown in Table 3, the newly added defect types are *italic.*

Table 3. Cloud ODC defect types [6]
– extended in AIODC with data/learning/thinking dimensions

ODC Version	Defect type
Original ODC	Function. Interface Checking Assignment Timing/serialization Build/package/merge Documentation Algorithm
Cloud ODC	*Isolation* *IaaS/PaaS*

4. Proposed framework

Defining attributes and characteristics of an AI systems is essential to allow classification of its defects. However, known software/system characteristics are not sufficient due to the special nature of AI systems. Several published literature discussed AI quality and AIP quality models such as [9, 10].

An AI system is different from other systems in the sense that it emulates human cognitive capabilities, it can learn, adapt, make decisions, and provide solutions based on complex unstructured data. To utilize the classification paradigm of the ODC model in AI system development projects, it is essential to undertake particular actions that cater to the distinctive features of AI. Therefore, the original ODC needs to be modified to be adapted to the defect analysis process of AI systems.

The AIODC classification process employs a systematic and repeatable methodology to guarantee consistency. Initially, each reported defect is scrutinized to ascertain whether it relates to AI-specific behavior by categorizing it into one of four AI attribute categories: Data (issues within training/testing datasets), Learning (errors during model training), Thinking (mistakes in inference or decision-making), or Not Related (general software defects). Each category is accompanied by explicit examples, such as mislabeled training data for Data defects, unstable convergence in Learning defects, or incorrect decision thresholds in Thinking defects. Severity ratings are allocated based on established decision rules that take into account the application domain, potential downstream effects, and the reversibility of outcomes, ensuring that "Catastrophic" severity is designated for defects with a significant risk of irreversible damage in critical domains. Ultimately, the impact mapping process utilizes Kharchenko's AI/AIP quality model, with annotators directed by a rubric that connects defect types to quality characteristics grounded in functional dependencies and observed system behavior.

First, the list of attributes is modified by introducing a new attribute called AI. The new attribute allows classifying defects based on Data, Learning, Thinking, and Not Related as explained in Table 4. The attribute allows classifying defects based on the Data used to train the AI, the AI's learning process, and the AI's training process, or not related to AI respectively. Figure 2 depicts the suggested AIODC attributes including the newly added one which is **bold**.

Table 4. AIODC's novel AI attribute
– classifying defects by data, learning, thinking, or not related phases

Classification	Description
Data	Issues with training / testing data.
Learning	Faults in the AI model training process.
Thinking	Faults in inference, logic, or decision making.
Not Related	Defects unrelated to AI logic or behavior.

Second, the severity attribute is changed from a four scale measure to a five scale measure by adding a new severity level named "Catastrophic." The severity attribute in the original ODC is measured on a scale from one to four (Critical, High, Medium, and Low), since defects in an AI systems may be catastrophic due to the fact that they may be used in numerous significant applications including ones related to human health monitoring, and illness diagnosing [29]. Therefore, the severity attribute is modified to be measured on a one to five scale (Catastrophic, Critical, High, Medium, and Low). Figure 3 depicts the suggested Severity attribute scale including the newly added level which is **bold**. To address context-sensitive risk profiles in AI systems, severity classifications encompass three key factors:

1. Application Criticality:

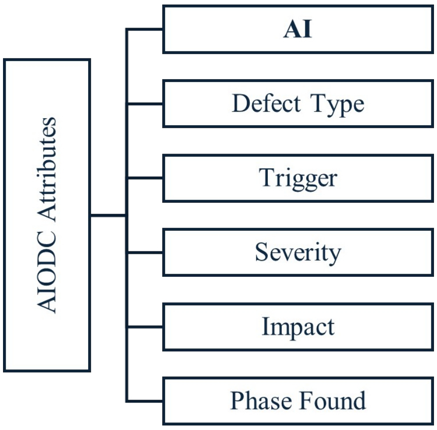

Figure 2. AIODC's attribute framework
– integrating novel AI dimension with ODC core

   - Catastrophic/Critical: Systems that affect human safety (healthcare, autonomous vehicles).
   - High/Medium: Enterprise systems (finance, security)
   - Low: Non-critical applications (entertainment, productivity)
2. Harm Reversibility:
   - Irreversible harm, such as: fatal misdiagnosis (Catastrophic)
   - Reversible errors, such as:transient data corruption (High/Medium)
3. Failure Scope:
   - Systemic failures, such as: poisoned training data (increase severity)
   - Localized errors, such as: single inference fault (allow for lower severity)

This methodology is consistent with recognized risk management standards, including ISO 14971:2019 for medical devices [30], which highlights severity levels that depend on context. Similarly, harm taxonomies specific to AI (for instance, irreversible systemic failures) are informed by fundamental safety literature [31].

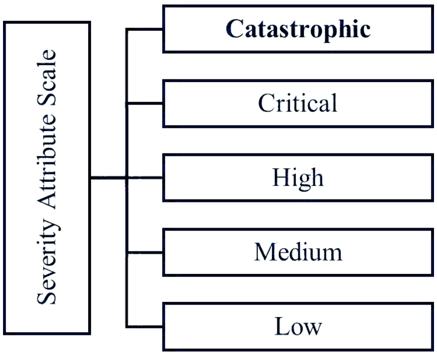

Figure 3. AIODC's 5-tier severity scale
– with domain-adaptive "catastrophic" level for AI risks

Finally, the impact areas attribute is modified to accommodate the new characteristics of AI system quality. As explained previously, Kharchenko et al. [11] developed a quality model and suggested a list of characteristics that are appropriate for AI systems, where the quality of an AI system is composed of the quality of AI and the quality of AIP. The

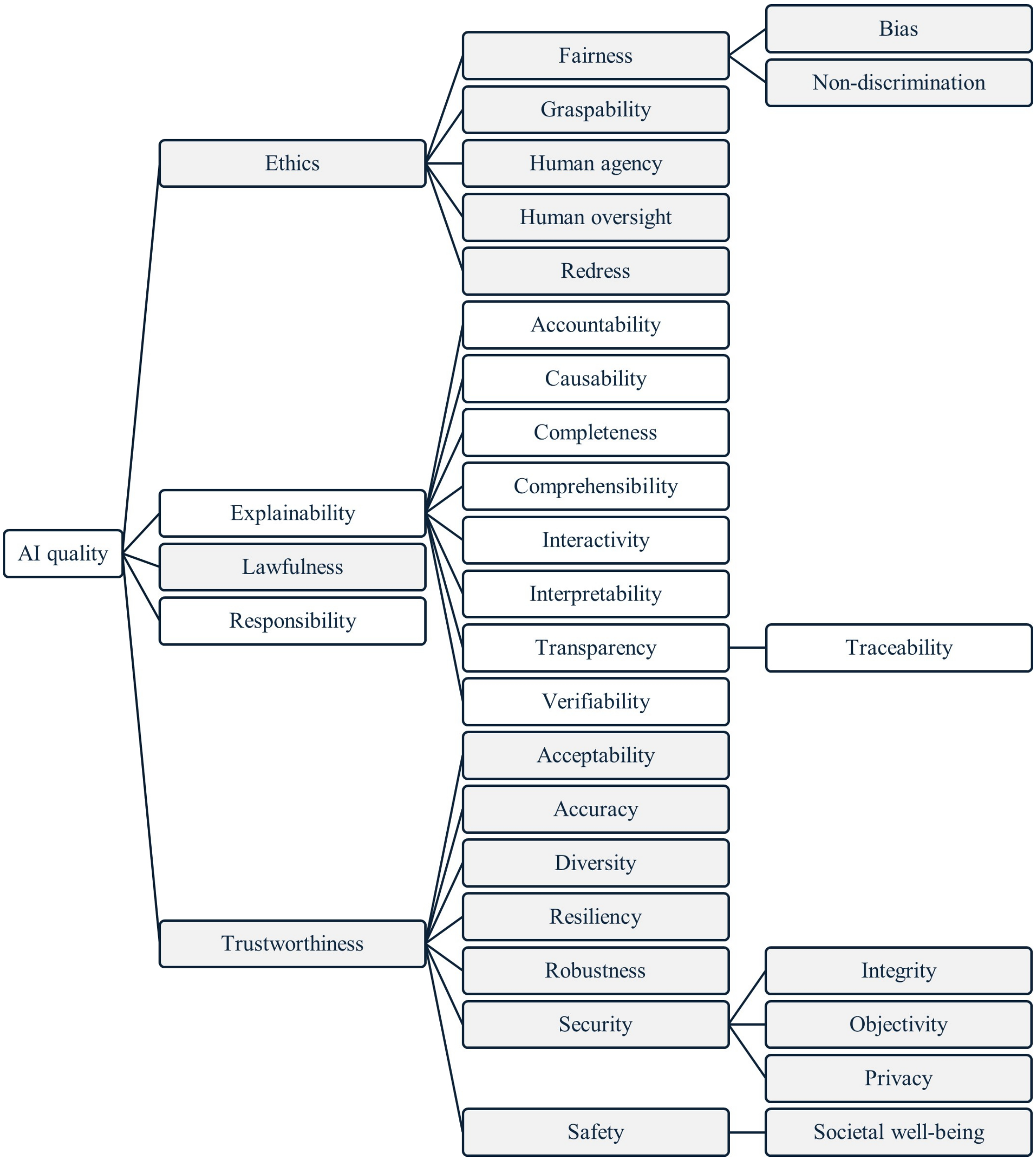

Figure 4. AI quality characteristics (Kharchenko et al. [11])
– foundation for AIODC impact mapping

characteristics were distributed among three layers, so if characteristic C1 depends on characteristic C2, then C2 will be in the next lower layer that C1 is in. In addition, some characteristics were assigned to AI quality, some assigned to AIP quality, and some were shared between both. Figure 4 depicts AI quality characteristic, while Figure 5 depicts AIP quality characteristic.

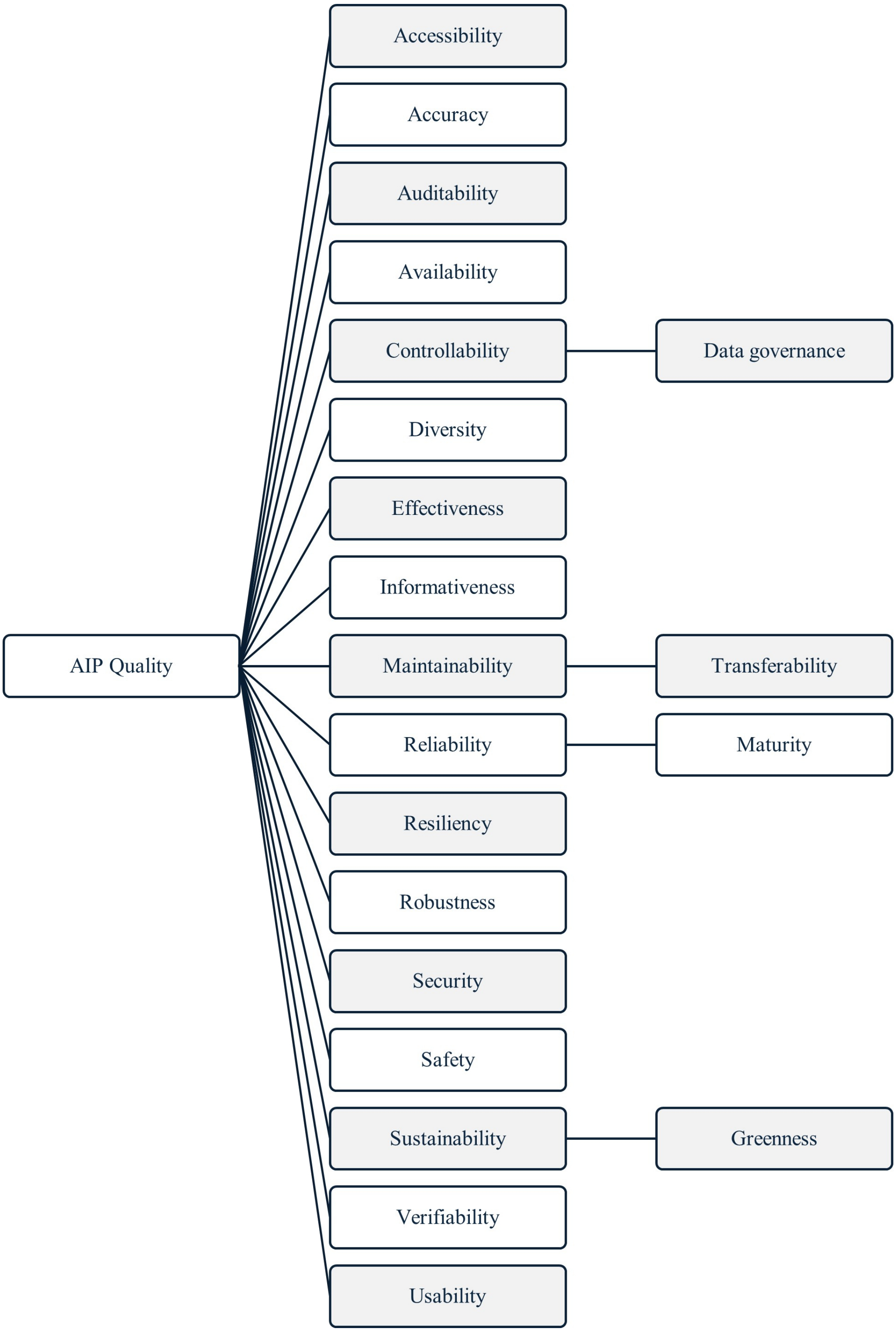

Figure 5. AIP quality characteristics (Kharchenko et al. [11])
– foundation for AIODC impact mapping

5. Case study

To test the applicability and suitability of the AIODC framework for AI systems, the benchmark dataset created by Morovati et al. [26] is utilized. The dataset comprises 100 ML defects sourced from TensorFlow/Keras on GitHub and Stack Overflow. For this study, 42 unique Keras defects from GitHub were selected for detailed analysis. This subset provided the highest count of non-duplicate, well-documented cases. The selection of Keras defects from GitHub was based on three primary criteria:
1. Highest defect count: 42 compared to TensorFlow's 20 on GitHub;
2. Uniformity of platform: to prevent duplication in cross-platform reporting;
3. Consistency within the framework: to minimize confounding variables arising from differences between frameworks.

Therefore, defects reported for TensorFlow or on Stack Overflow were excluded to prevent duplication bais.

As mentioned above, the dataset contains 100 bugs reported on two platforms: GitHub and Stack Overflow, for two widely used ML frameworks: TensorFlow and Keras. In this study, defects reported on GitHub for the Keras framework will classified based on the proposed AIODC framework. Such decision was made based on the number of defects that each framework has on each platform. Table 5 shows the number of defects for each framework on both platforms.

Table 5. Defect distribution in benchmark dataset [26]
– Keras/GitHub selected ($n = 42$) for case study due to maximal unique defects

Platform	TensorFlow	Keras
GitHub	20	42
Stack Overflow	3	35

5.1. Classification methodology

To classify defects to implement AIODC, Two separate annotators examined defects according to predefined guidelines:
1. Training: A two-hour session featuring sample defect mappings.
2. Procedure:
   - Annotators evaluated descriptions of GitHub issues;
   - Designated AI attributes (Data/Learning/Thinking/Not Applicable);
   - Assigned severity ratings (Catastrophic-Low).
3. Dispute Resolution: A third expert evaluated the conflicting labels.
4. Reliability Assessment: Cohen's Kappa = 0.82 (indicating substantial agreement).

5.2. Bug classification based on the AI attribute

The number of defects reported for the Keras framework on GitHub is 42 defects. Classifying defects based on the AI attribute resulted in having 2 defects related to Data, 18 related to the Learning process of the AI framework, 14 related to the Thinking process of the AI framework, and 8 defects not related to AI. Classification results are shown in Table 6.

Table 6. AIODC classification of unique Keras defects (GitHub subset, $n = 42$)

Defect Description	Count	AI Attribute	Total	%
Deprecated API	3	Not Related	8	19.05%
Missing API call	2	Not Related		
Missing argument scoping	1	Not Related		
Wrong API usage	2	Not Related		
Missing dense layer	1	Think	14	33.33%
Suboptimal network structure	4	Think		
Wrong size for convolutional layer	1	Think		
Wrong layer type	2	Think		
Wrong network architecture	3	Think		
Wrong type of activation function	3	Think		
Wrong tensor shape	1	Data	2	4.76%
Missing pre processing step	1	Data		
Suboptimal batch size	4	Learn	18	42.86%
Suboptimal number of epochs	4	Learn		
Wrong loss function calculation	1	Learn		
Wrong optimization function	4	Learn		
Wrong selection of loss function	5	Learn		

5.3. Bug classification based on the severity attribute

The severity of defects was categorized based on domain-adaptive thresholds outlined in Section 4, which evaluate:

1. Application Criticality: Keras (general ML development) was classified as medium-criticality (non-safety-critical).
2. Harm Reversibility: Irreversible errors (such as model collapse) heightened the severity level.
3. Failure Scope: Systemic defects (for instance, hyperparameter corruption) were given precedence over localized problems.

Therefore, the revised Severity Assignment Logic is:

1. Catastrophic (5): Defects that result in irreversible systemic damage (e.g., "incorrect loss function" causes permanent divergence of the model).
2. Critical (4): Defects that lead to recoverable yet significant outcomes (e.g., "incorrect tensor shape" causes failure in training; can be rectified through data correction).
3. High (3) to Low (1): Localized or transient defects (e.g., "suboptimal epochs" causes decrease in accuracy; can be resolved through retuning).

Severity classifications were aligned with the domain-adaptive standards outlined in Section 4, which encompass criticality levels inspired by ISO 14971 [30] and the AI failure modes detailed in [31]. In the case of Keras, categorized as medium-criticality, defects were classified as "Catastrophic" solely under the following conditions:

1. Irreversible systemic damage,
2. Equivalence of life/financial risk (according to [30] Annex C),
3. Uncontainable propagation of errors (as specified in [31] Section 3.1).

The application of these criteria to defects in Keras produced the classification shown in Table 7.

Table 7. Severity classification
– validating domain-adaptive thresholds ($n = 42$)

Severity level	Defects ($n = 42$)	%
Catastrophic	9	21.43%
Critical	10	23.81%
High	12	28.57%
Medium	11	26.19%
Low	0	0.00%

5.4. Bug classification based on the impact attribute

Expanding upon the classification of defects, we subsequently examine the manner in which AI defects adversely affect the fundamental quality characteristics of AI systems and their associated platforms. Unlike traditional defect analysis, this approach isolates AI-specific failure vectors through quality-attribute-centric mapping. Although categorization serves to pinpoint the origins of failures (Data/Learning/Thinking/Not Related), effective remediation necessitates a comprehension of the specific quality characteristic that these defects undermine. This section correlates defects identified in the case study with empirically observed degradations in the quality characteristics of AI/AIP (following Kharchenko's quality model [11] for characteristic definitions), thereby uncovering deterministic patterns between defect types and impacted attributes. These consistent failure signatures facilitate accurate localization of quality risks – thereby affirming AIODC's core proposition that the sources of defects dictate impact profiles across functionally orthogonal quality dimensions. Table 8 shows the classification of found defects based on their impact on the Kharchenko AI/AIP Characteristics.

The classification of defects in relation to AI/AIP quality characteristics shown in Table 8 was performed through a two-phase expert evaluation procedure. Annotators employed a specified rubric that correlated defect types with Kharchenko's hierarchical quality characteristics, taking into account both the functional role of the defect's location (such as model architecture and training loop) and the type of failure (for instance, robustness degradation and accuracy loss). In instances of uncertainty, annotators referred to documentation and discussions from issue threads to gain a clearer understanding of the defect's context prior to finalizing the labels. This systematic mapping process minimizes subjectivity and improves reproducibility, thereby ensuring that subsequent researchers can utilize AIODC on other datasets with similar consistency.

Analysis indicates clear failure patterns across AI defect categories: Defects originating in the Learning phase predominantly degrade operational efficiency metrics (such as computational resource usage and training duration), while those emerging from the Thinking phase primarily compromise system integrity outcomes (including decision reliability and output consistency). This illustrates how defect genesis in AI systems dictates failure modes across functionally orthogonal impact domains.

This division illustrates the architectural distinction between the training and inference subsystems. Learning defects present themselves as resource-related issues, whereas Thinking defects arise as behavioral irregularities.

Table 8. Impact mapping of Keras defects to AI/AIP quality characteristics [11]

AI Defect	Quality Model	Impacted characteristic	
		Layer1	Layer2
Deprecated API	AIP	Maintainability	=
Missing API call	AIP	Reliability	=
Missing argument scoping	AI	Security	Integrity
Wrong API usage	AIP	Accuracy	=
Missing dense layer	AI	Trustworthiness	Accuracy
	AIP	Accuracy	=
Suboptimal network structure	AI	Effectiveness	=
Wrong size for convolutional layer	AI	Trustworthiness	Robustness
	AIP	Robustness	=
Wrong layer type	AI	Trustworthiness	Accuracy
	AIP	Accuracy	=
Wrong network architecture	AI	Trustworthiness	Accuracy
	AI	Explainability	Completeness
	AIP	Accuracy	=
Wrong type of activation function	AI	Trustworthiness	Accuracy
	AIP	Accuracy	=
Wrong tensor shape	AIP	Reliability	=
Missing pre processing step	AI	Trustworthiness	Robustness
	AIP	Robustness	=
Suboptimal batch size	AI	Effectiveness	=
Suboptimal number of epochs	AI	Trustworthiness	Accuracy
	AIP	Accuracy	=
	AIP	Effectiveness	=
Wrong loss function calculation	AI	Trustworthiness	Accuracy
	AIP	Accuracy	
Wrong optimization function	AI	Effectiveness	=
Wrong selection of loss function	AI	Trustworthiness	Accuracy
	AI	Trustworthiness	Robustness
	AIP	Accuracy	=
	AIP	Robustness	=

5.5. Two-way classification

While analyzing defects classified using one attribute (one-way classification) can be insightful. However, it may not adequately inform the development team about defects and the associated high-risk areas. Implementing two-way classification will enhance the understanding of the observed defects, which may lead to more informed decisions. Therefore, by utilizing two-way classification, it becomes feasible to concentrate on rectifying high-priority defects, ultimately improving the reliability and quality of the AI system. Two way classification was implemented on the AI and Severity attributes, results are depicted in Figure 6.

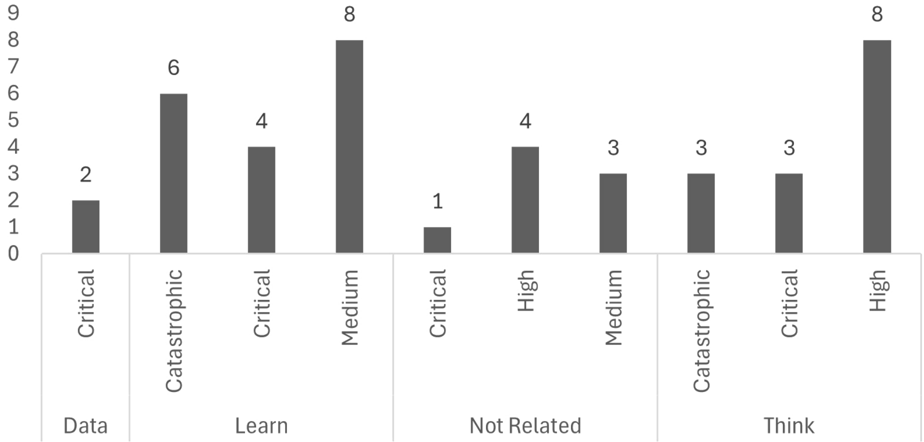

Figure 6. AI/severity correlation
– revealing high-risk defect clusters (e.g., learning + catastrophic)

6. Discussion

The findings of this research have significant implications for the practices of quality assurance in AI software. By establishing a clear connection between the origins of defects and their severity outcomes, as well as the quality characteristics affected, AIODC empowers development teams to focus on defects that present the greatest operational and safety hazards. For instance, defects categorized under Thinking that compromise trustworthiness or accuracy may require urgent intervention in healthcare AI, whereas defects arising during the Learning phase that reduce efficiency might be prioritized in large-scale industrial AI applications where computational expenses are paramount. Additionally, the organized AIODC framework could be incorporated into continuous integration workflows, facilitating real-time assessments of defect risks and proactive quality management in AI development.

From a methodological standpoint, the introduction of a new AI attribute (Data, Learning, Thinking, Not Related) alongside a five-tier severity scale offers a more comprehensive and nuanced defect profile compared to current ODC adaptations. The capacity to distinguish AI-specific challenges from general software defects enables more focused debugging approaches and supports the development of specialized test cases for data management, model training, and inference processes. The identified correlation between AI attribute categories and severity levels strengthens the hypothesis that the type of defect serves as an indicator of potential harm, which is essential in risk-based testing and quality strategy formulation.

The classification of impact utilizing Kharchenko's AI/AIP quality model has also uncovered persistent failure patterns: defects related to Learning were found to diminish both efficiency and robustness, whereas defects associated with Thinking had a disproportionate effect on trustworthiness and accuracy. This observation reinforces the architectural separation between the training and inference subsystems of AI and indicates that measures for defect prevention ought to be tailored to each subsystem. For example, tools for automated training monitoring may prove to be more effective in identifying anomalies during the Learning phase, while audits focused on explainability could more effectively address irregularities occurring in the Thinking phase.

From a theoretical perspective, this research adds to the developing domain of Software Engineering for AI by transforming quality models into a practical framework for defect analysis. Although previous studies have suggested quality metrics or general defect taxonomies, AIODC serves to connect abstract quality attributes with tangible workflows for defect classification. This linkage is crucial for enhancing quality assurance processes in AI initiatives, where the detection of defects is complicated by non-deterministic behaviors and evolving models.

6.1. Limitations

This study is constrained by a relatively small sample size and its concentration on a singular AI framework (Keras). While this approach reduces confounding variables like cross-platform duplication, it limits the generalizability of the findings. Subsequent research should assess AIODC across more frameworks (e.g., TensorFlow, PyTorch) and larger datasets to verify its scalability and robustness.

7. Conclusion and prospective

Artificial Intelligence systems are predominantly deployed as critical software solutions across a wide range of domains, making the identification and resolution of software defects essential to ensure system quality.

This study presented AIODC, a defect classification framework specifically tailored to the unique nature of AI-based systems. Through extending ODC with AI-specific attributes (Data, Learning, and Thinking), a five-level severity scale, and impact mapping based on AI/AIP quality model, AIODC offers a structured methodology for analyzing AI defects.

The analysis of 42 Keras defects substantiated the frameworks feasibility and revealed consistent correlations among defect origins, severity, and quality impacts. Defects related to Learning were found to be the most common and were linked to greater severity, while defects associated with Thinking primarily affected trustworthiness and accuracy.

Future research will aim to validate AIODC using larger and more diverse datasets, incorporate it into automated CI/CD pipelines for real-time defect classification, and investigate the application of large language models for the automation of annotation and analysis.

CRediT authorship contribution statement

The author is responsible for the conceptualization of this research work. He created the framework, prepared the figures of the manuscript, wrote the main manuscript, and reviewed the manuscript draft.

Declaration of competing interest

The author certifies that he has no affiliations or involvement in any organization or entity with any financial or non-financial interest in the subject matter or materials discussed in this manuscript.

Data and code availability

The full dataset and reproduction package used in this study – including the curated set of 42 Keras defects and all AIODC classification materials – are publicly available on Zenodo at: https://doi.org/10.5281/zenodo.17843077.

Funding

No funds, grants, or other support was received for this work.

References

[1] M.O. Alannsary, *Quality improvement of SaaS (Software as a Service) in the Cloud*, Ph.D. dissertation, Southern Methodist University, 2016.

[2] J. Tian, *Software quality engineering: Testing, quality assurance, and quantifiable improvement.* John Wiley and Sons, 2005.

[3] P.J. Denning, “What is software quality?” *Communications of the ACM*, Vol. 35, No. 1, 1992, pp. 13–15.

[4] R. Chillarege, I.S. Bhandari, J.K. Chaar, M.J. Halliday, D.S. Moebus et al., “Orthogonal defect classification – A concept for in-process measurements,” *IEEE Transactions on software Engineering*, Vol. 18, No. 11, 1992, pp. 943–956.

[5] L. Ma and J. Tian, “Web error classification and analysis for reliability improvement,” *Journal of Systems and Software*, Vol. 80, No. 6, 2007, pp. 795–804.

[6] M. Alannsary and J. Tian, “Cloud-ODC: Defect classification and analysis for the Cloud,” in *Proceedings of the International Conference on Software Engineering Research and Practice (SERP)*, 2015, p. 71.

[7] S. Martínez-Fernández, J. Bogner, X. Franch, M. Oriol, J. Siebert et al., “Software engineering for AI-based systems: A survey,” *ACM Transactions on Software Engineering and Methodology (TOSEM)*, Vol. 31, No. 2, 2022, pp. 1–59.

[8] S.C.G. Varma, “Bridging the gap between traditional software engineering and modern AI development practices,” *International Journal of Emerging Trends in Computer Science and Information Technology*, Vol. 4, No. 3, 2023, pp. 32–40.

[9] B. Gezici and A.K. Tarhan, “Systematic literature review on software quality for AI-based software,” *Empirical Software Engineering*, Vol. 27, No. 3, 2022, p. 66.

[10] M.A. Ali, N.K. Yap, A.A.A. Ghani, H. Zulzalil, N.I. Admodisastro et al., “A systematic mapping of quality models for AI systems, software and components,” *Applied Sciences*, Vol. 12, No. 17, 2022, p. 8700.

[11] V. Kharchenko, H. Fesenko, and O. Illiashenko, “Quality models for artificial intelligence systems: Characteristic-based approach, development and application,” *Sensors*, Vol. 22, No. 13, 2022, p. 4865.

[12] V. Lenarduzzi, F. Lomio, S. Moreschini, D. Taibi, and D.A. Tamburri, “Software quality for AI: Where we are now?” in *Software Quality: Future Perspectives on Software Engineering Quality: 13th International Conference, SWQD 2021, Vienna, Austria, January 19–21, 2021, Proceedings 13.* Springer, 2021, pp. 43–53.

[13] L. Bailey, *The Impact of AI on Software Development*, Ph.D. dissertation, Worcester Polytechnic Institute, 2024.

[14] H. Wang and L. Yuan, “Software engineering defect detection and classification system based on artificial intelligence,” *Nonlinear Engineering*, Vol. 11, No. 1, 2022, pp. 380–386.

[15] M. Pandit, D. Gupta, D. Anand, N. Goyal, H.M. Aljahdali et al., "Towards design and feasibility analysis of DePaaS: AI based global unified software defect prediction framework," *Applied Sciences*, Vol. 12, No. 1, 2022, p. 493.

[16] A. Tosun, A. Bener, and R. Kale, "AI-based software defect predictors: Applications and benefits in a case study," in *Proceedings of the AAAI conference on artificial intelligence*, Vol. 24, No. 2, 2010, pp. 1748–1755.

[17] J. Pachouly, S. Ahirrao, K. Kotecha, G. Selvachandran, and A. Abraham, "A systematic literature review on software defect prediction using artificial intelligence: Datasets, data validation methods, approaches, and tools," *Engineering Applications of Artificial Intelligence*, Vol. 111, 2022, p. 104773.

[18] R.G. Hussain, K.C. Yow, and M. Gori, "Leveraging an enhanced codebert-based model for multiclass software defect prediction via defect classification," *IEEE Access*, 2025.

[19] H.V. Pandhare, "Future of software test automation using AI/ML," *International Journal Of Engineering And Computer Science*, Vol. 13, No. 05, 2025.

[20] X. Dong, J. Wang, and Y. Liang, "A novel ensemble classifier selection method for software defect prediction," *IEEE Access*, 2025.

[21] S. Stradowski and L. Madeyski, "'Your AI is impressive, but my code does not have any bugs' Managing false positives in industrial contexts," *Science of Computer Programming*, 2025, p. 103320.

[22] X. Chen, X. Hu, Y. Huang, H. Jiang, W. Ji et al., "Deep learning-based software engineering: Progress, challenges, and opportunities," *Science China Information Sciences*, Vol. 68, No. 1, 2025, pp. 1–88.

[23] A. Güzel and A. Egesoy, "Development of an AI-driven model for advancing software engineering practices," *International Journal of Innovative Research in Computer Science and Technology*, Vol. 13, No. 1, 2025, pp. 1–11.

[24] B. Vinayagasundaram and S. Srivatsa, "Software quality in artificial intelligence system," *Information Technology Journal*, Vol. 6, No. 6, 2007, pp. 835–842.

[25] C. Tao, J. Gao, and T. Wang, "Testing and quality validation for AI software – perspectives, issues, and practices," *IEEE Access*, Vol. 7, 2019, pp. 120 164–120 175.

[26] M.M. Morovati, A. Nikanjam, F. Khomh, and Z.M. Jiang, "Bugs in machine learning-based systems: A faultload benchmark," *Empirical Software Engineering*, Vol. 28, No. 3, 2023, p. 62.

[27] T. Zhang, D. Han, V. Vinayakarao, I.C. Irsan, B. Xu et al., "Duplicate bug report detection: How far are we?" *ACM Transactions on Software Engineering and Methodology*, Vol. 32, No. 4, 2023, pp. 1–32.

[28] R.A. Radice and R.W. Phillips, *Software engineering: An industrial approach*, Vol. 1. Prentice-Hall, Inc., 1988.

[29] V. Lenarduzzi and M. Isomursu, "AI Living Lab: Quality assurance for AI-based health systems," in *IEEE/ACM 2nd International Conference on AI Engineering – Software Engineering for AI (CAIN)*, 2023, pp. 86–87.

[30] International Organization for Standardization, *Medical devices – Application of risk management to medical devices*, Std. ISO 14 971:2019, 2019.

[31] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman et al., "Concrete problems in AI safety," *arXiv preprint arXiv:1606.06565*, 2016.

Authors and affiliations

Mohammed Alannsary
e-mail: alannsary@hotmail.com
ORCID: https://orcid.org/0000-0003-2254-1441
Department of Digital Transformation Programs,
The Institute of Public Administration, Saudi Arabia