Acta Informatica Pragensia
2026, Volume 15, Issue 2, 514–546
https://doi.org/10.18267/j.aip.303

PRAGUE UNIVERSITY OF ECONOMICS AND BUSINESS

Review Open Access

Artificial Intelligence in Software Testing and Beyond: A Review of Current Practices and Emerging Challenges

Codrina-Victoria Lisaru, Claudiu-Vasile Kifor

Department of Industrial Engineering and Management, Faculty of Engineering, Lucian Blaga University of Sibiu, Sibiu, Romania

Corresponding author: Codrina-Victoria Lisaru (codrinavictoria.ignat@ulbsibiu.ro)

Editorial Record

First submission received:
August 29, 2025

Revisions received:
September 29, 2025
November 12, 2025
January 8, 2026

Accepted for publication:
January 19, 2026

Academic Editor:
Ahad ZareRavasan
Masaryk University, Czech Republic

This article was accepted for publication by the Academic Editor upon evaluation of the reviewers' comments.

How to cite this article:
Lisaru, C.-V., & Kifor, C.-V. (2026). Artificial Intelligence in Software Testing and Beyond: A Review of Current Practices and Emerging Challenges. *Acta Informatica Pragensia*, 15(2), 514–546. https://doi.org/10.18267/j.aip.303

Abstract

Background: Artificial intelligence (AI) is increasingly used both to test software (T1) and to assure AI-based systems (T2), with adjacent software-engineering work that shapes testing practice (T3). Prior reviews are mostly descriptive and rarely report comparable maturity or replicability signals.

Objective: To provide a PRISMA-style systematic review (2015–2025, Web of Science) that maps T1–T2–T3 within a testing-centric frame, audits evidence maturity, threats reporting, and artefact openness per paper, and adds an explicit lens of large language models or generative AI (LLMs/GenAI).

Methods: We queried the Web of Science Core Collection (2015–2025), screened via a predefined protocol, and extracted ten items (D1–D10) per study to normalize comparisons. Seventy-two papers met the criteria. Findings are organized into three themes: (T1) AI-based software testing, (T2) testing/validation of AI systems, and (T3) AI-related software engineering topics with implications for testing—T3 corresponding to the "beyond" in the paper's title.

Results: The corpus is limited in practice-oriented evidence: 31 laboratory/simulation, 3 industrial, 10 hybrid, 6 conceptual/guideline and 22 secondary studies. Only 18/72 provide public artefacts; 33/72 report no empirical metrics. By theme, T1=32, T2=15, T3=25; the LLMs/GenAI subset totals 10 papers. Openness strongly co-occurs with measurable outcomes (88.9% of artefact-sharing papers report metrics vs 42.6% without), yet "all-three credible" studies (industrial/hybrid + open artefacts + metrics) are rare (4/72 overall; 1/10 for LLMs/GenAI).

Conclusion: AI shows promise for testing, but evidence remains thin on industrial adoption and reproducibility. We recommend prioritizing hybrid/industrial validations, releasing artefacts by default, and using standardized task–metric bundles. The review presents T1 and T2 results, separates T3 for scope clarity, and provides actionable maturity and replicability signals to guide responsible, empirical adoption.

Index Terms

Software testing; Artificial intelligence; AI; AI-driven testing; Software engineering; Requirements engineering; Human-AI collaboration; Software quality; Large Language Models; LLMs.

1 INTRODUCTION

Artificial Intelligence (AI) has experienced a rapid uptake in adoption across various industries in recent years, enhancing both the quality and diversity of existing products and transforming nearly every facet of modern technology. Numerous studies highlight that AI has substantially improved efficiency in many fields and continues to grow in popularity, particularly within computing, where it offers innovative solutions to complex problems (Khaleel & Anan, 2023).

However, in the context of software testing, it remains necessary to clarify which testing activities are currently most affected by AI-based approaches and what kinds of outcomes are being reported to substantiate the claimed benefits.

Software testing plays a crucial role in software development, ensuring the technical quality and economic viability of software products (Gurcan et al., 2022). Given that software is developed by humans, errors are inevitable and can become increasingly costly to correct as software complexity grows (Gurcan et al., 2022). Effective testing demands diverse strategies, techniques, tools, and resources (Gurcan et al., 2022). As noted in (Gurcan et al., 2022), software testing permeates every phase of the software development life cycle, beginning with requirements validation and concluding with customer acceptance. In this setting, an open issue is whether AI support is concentrated in specific phases or activities, and how consistently these applications are evaluated across different testing contexts.

Across many testing processes and activities, AI has emerged as a powerful ally, driving innovation in techniques and strategies. AI offers numerous advantages, including faster task completion, reduced manual effort, efficient handling of complex or repetitive tasks, and the production of more precise results (Khaleel & Anan, 2023). It enhances testing quality, saves time, and enables the generation of intelligent and accurate test cases (Hourani et al., 2019; Khan et al., 2024). Research has demonstrated diverse applications of AI in software testing, ranging from black-box testing (Lima et al., 2020) to code generation quality assessment (Tosi, 2024), and the evaluation of Large Language Models' effectiveness in testing contexts (Li et al., 2025).

This review is motivated by two key factors:

1. The rise of powerful AI models, particularly Large Language Models (LLMs), has dramatically reshaped the software testing landscape, creating a need for a consolidated understanding of their capabilities and implications, including which testing tasks they are most applied.
2. As AI systems become increasingly integrated into critical applications, there is a growing urgency for robust testing strategies, ethical safeguards, and explainable validation methods, and for understanding whether current empirical evidence is sufficiently practice-proximal and reproducible to support deployment in such settings.

Despite the growing number of studies and reviews, the field remains fragmented, and there is limited consolidated evidence regarding where AI is applied across testing activities and how mature and reproducible the reported results are (evidence setting, artefacts, and outcome measurement).

This paper provides a PRISMA-style systematic literature review (2015–2025, Web of Science) that synthesizes the literature on AI in software testing and quality through a testing-centric lens, and evaluates research maturity using evidence setting, artefact availability, and outcome measurement. In doing so, it clarifies how the literature balances conceptual proposals versus empirical evaluation, and how often reported benefits are supported by measurable outcomes and reusable artefacts.

The goal is to map AI applications across testing activities and assess research maturity via evidence setting, artefact availability, and outcome measurement, deriving implications and open challenges for future work.

This paper is organized as follows: Section 2 presents a short overview of the related work; Section 3 discusses the methodology used for paper selection and analysis; Section 4 presents the literature analysis, exploring AI's applications in software testing and quality; Section 5 offers a discussion on connections between AI and software engineering, including challenges and open issues; Section 6 summarizes key conclusions, study limitations and suggests directions for future research; References section provides the list of references cited throughout the paper; Appendix section provides Appendix A and Appendix B, containing the supplementary tables with details that support the results and discussion.

2 RELATED WORK

2.1 AI in software testing (T1)

Prior syntheses mostly examine AI techniques mapped to classic testing tasks but differ in evidence depth and in how explicitly they assess tool support or reporting quality. A tertiary study aggregates 20 secondary reviews to show broad coverage of activities such as test case generation, prioritization, oracle design, and optimization;

however, its contribution is primarily to synthesize what the secondary studies report, rather than to provide standardized, per-primary maturity or reproducibility coding (Amalfitano et al., 2024).

Bibliometric mappings complement these syntheses by charting publication dynamics and thematic evolution in "intelligent testing"; by design, they focus on trends rather than on activity-level effectiveness evaluation or per-study artefact openness assessments (Boukhlif et al., 2023).

Focused SLRs on test suite optimization identify extensive work on optimization sub-tasks (including selection, reduction, and prioritization) and highlight gaps relevant to comparability and reuse (e.g., limited use of standard datasets in some categories, missing experimental setup details, and the need for large-scale industrial evaluation). They also flag the need for multi-objective formulations (Kiran et al., 2019).

The Reinforcement Learning-for-testing SLR narrows to the role of RL in software testing and reports that RL use has largely narrowed to two main applications; it also highlights shortages in advanced RL techniques and in multi-agent RL usage (Abo-eleneen et al., 2023).

Overall, the T1 corpus skews toward breadth, trend mapping, or narrow technique niches. Across these strands, comparatively fewer works provide standardized, per-paper coding of maturity, artefact availability, and reproducibility signals in a way that enables cross-study comparison.

2.2 Testing/validation of AI systems (T2)

Assurance-oriented reviews bring a different axis: validation strategies for AI-based systems (e.g., trials, simulation, redundancy, expert judgment), relevance levels, and the need for continuous runtime validation. They deliver strong taxonomies of validation methods and qualitative realism discussions, but they are not aligned with classic software-testing workflows or metrics and do not aim to map results to detailed testing activity taxonomies (Myllyaho et al., 2021).

A large Software Engineering for Artificial Intelligence (SE4AI) systematic mapping study reinforces that quality/validation are central concerns while noting under-studied maintenance and inconsistent reporting, including that many primary studies do not report threats to validity (Martinez-Fernandez et al., 2022).

These T2 syntheses characterize the assurance space well but stop short of a unified comparison with T1 activities or of applying a standardized, cross-paper scheme for maturity/replicability coding (Myllyaho et al., 2021; Martinez-Fernandez et al., 2022).

2.3 AI across the software engineering lifecycle (T1/T2/T3)

Lifecycle-wide reviews and surveys provide panoramic coverage of where AI appears across software engineering phases (Alenezi & Akour, 2025; Durrani et al., 2024; Navaei & Tabrizi, 2022). They highlight testing as an active phase—often the most reported—yet remain high-level for testing-specific activities, frequently lacking a taxonomy tuned to testing practice and rarely separating AI-for-testing (T1) from testing/assurance of AI systems (T2) (Navaei & Tabrizi, 2022; Alenezi & Akour, 2025).

Some apply top-cited or venue-limited sampling strategies, which risks selection bias and further constrains depth (Durrani et al., 2024). Synthetic knowledge-synthesis/bibliometric work complements this by clustering themes (e.g., defect prediction, automation) yet, by design, cannot judge technique effectiveness, reproducibility, or practice maturity; reliance on a single index also under-captures rapidly evolving GenAI work (Kokol, 2024).

Therefore, these lifecycle views situate testing in software engineering for Artificial Intelligence but don't provide methodologically comparable, per-paper maturity or replicability signals for testing research (Alenezi & Akour, 2025; Durrani et al., 2024; Navaei & Tabrizi, 2022; Kokol, 2024).

2.4 Cross-cutting deficiencies in prior syntheses

Across prior syntheses, three recurring limitations emerge. First, **industrial realism remains thin or fragmented**: many findings are inherited from laboratory/simulation benchmarks, with relatively few systematically reported deployments or hybrid evaluations, and those are fragmented (Kiran et al., 2019; Abo-eleneen et al., 2023; Martinez-Fernandez et al., 2022).

Second, **methodological comparability is weak**: incomplete experimental reporting and limited standard dataset use hinder comparability and reuse in parts of the optimization literature, while several syntheses rely on constrained indexing/sampling choices that shape coverage (Kiran et al., 2019; Durrani et al., 2024; Boukhlif et al., 2023).

Third, **LLM/GenAI coverage is mostly conceptual or absent**: many reviews either predate the rapid adoption of LLMs or discuss copilots at a high level without task-specific validity considerations such as contamination controls, oracle design, or reproducibility (Amalfitano et al., 2024; Alenezi & Akour, 2025).

2.5 How this review advances the state of the art

Motivated by these gaps, our study integrates both directions—**AI-for-testing (T1)** and **testing/assurance of AI systems (T2)**—and adds a third synthesis stream on AI-related software engineering topics (with implications for testing) **(T3)**, all within a testing-centric frame. To improve cross-study comparability, we apply a **deterministic theme assignment** and a unified data extraction schema. Specifically, studies are assigned to T1/T2/T3 using D2 (AI technique) as the primary signal, this assignment is checked against D3 (testing activity), and ties are resolved using D4 (domain/system under test), which makes the classification reproducible. We further operationalize maturity in a comparable way by coding the evidence setting per paper (D7), enabling like-for-like comparisons across laboratory, industrial, and hybrid evaluations. Replicability is assessed systematically through the reporting of threats to validity (D8) and artefact availability (D9), which serve as practical indicators of reusability and reproducibility.

Finally, we provide a task-level analysis of LLM/GenAI work by characterizing roles (copilot/oracle/generator), the evaluated tasks and benchmarks, and validity risks—details that are typically missing from prior lifecycle- and assurance-oriented syntheses (Alenezi & Akour, 2025; Durrani et al., 2024; Myllyaho et al., 2021; Martinez-Fernandez et al., 2022).

In doing so, it resolves taxonomy inconsistencies across earlier reviews, and surfaces where claims about effectiveness are unsupported by open artefacts or industrial validation. To keep the synthesis precise, the corpus is limited to peer-reviewed, English-language papers; rapid GenAI progress may outpace snapshots despite normalization. Nevertheless, the consistent D-item coding and T1/T2/T3 mapping make methodological differences, gaps, and practical maturity in AI-related testing transparent and comparable—exactly what prior syntheses have lacked. In support of this synthesis, Appendix A presents a consolidated comparison aligned to these claims.

3 METHODOLOGY

3.1 Research design

This study followed a systematic literature review (SLR) design, with the objective of identifying, classifying, and analysing current trends in the use of artificial intelligence in software testing and (adjacently) software quality. This method was chosen because it enables a structured and transparent analysis of existing research and ensures reproducibility of results. We followed PRISMA-style stages (identification – screening – eligibility – inclusion) and defined the protocol (search strings, filters, inclusion/exclusion rules, data items, and analysis plan). The time window of the papers' publication was 2015–2025, focusing on peer-reviewed journal articles, review and conference papers indexed in Clarivate Web of Science Core Collection.

We executed the review in October-November 2025, following the PRISMA-style stages (identification, screening, eligibility, inclusion) on Web of Science Core Collection (2015–2025; journal articles + reviews + conference papers; English; Open Access), then removed duplicates and screened full texts.

3.2 Search strategy and data collection

The literature search for this review was carried out in October 2025 using the Clarivate Web of Science database, with relevant papers selected during October and November 2025. Two search queries were executed to ensure comprehensive coverage of publications related to the use of Artificial Intelligence in software testing.

3.3 Inclusion and exclusion criteria

Inclusion criteria: peer-reviewed articles, published in English, studies where AI/ML (including DL, RL, LLMs) is applied to software testing or software quality activities (e.g., test generation, prioritization/minimization, fault localization, oracle construction, defect prediction, reliability/verification), conceptual/review work synthesizing AI in software testing/quality (state of the art, taxonomies, mappings).

Exclusion criteria: clear domain mismatch (e.g. AI in physics, gaming, medicine, agriculture, chemistry, accounting) without a testing/quality focus in software engineering), non-article types (editorials, notes, tutorials), theses, not open access (per filter), not in English, or outside 2015 – 2025, missing sufficient detail to understand method or contribution after full-text check.

3.4 Study selection process

The first search employed the terms "software testing" AND "artificial intelligence" in the topic fields (title, abstract, and keywords), initially retrieving 3,322 records (Figure 1). Limiting the results to English-language publications reduced this number to 3,268. Filtering by document type to include only articles, review articles and proceedings papers narrowed the set further to 3,249. Additional filters based on research areas—such as Engineering, Computer Science, Science Technology Other Topics, Telecommunications, and Automation Control Systems—reduced the pool to 1,821 articles.

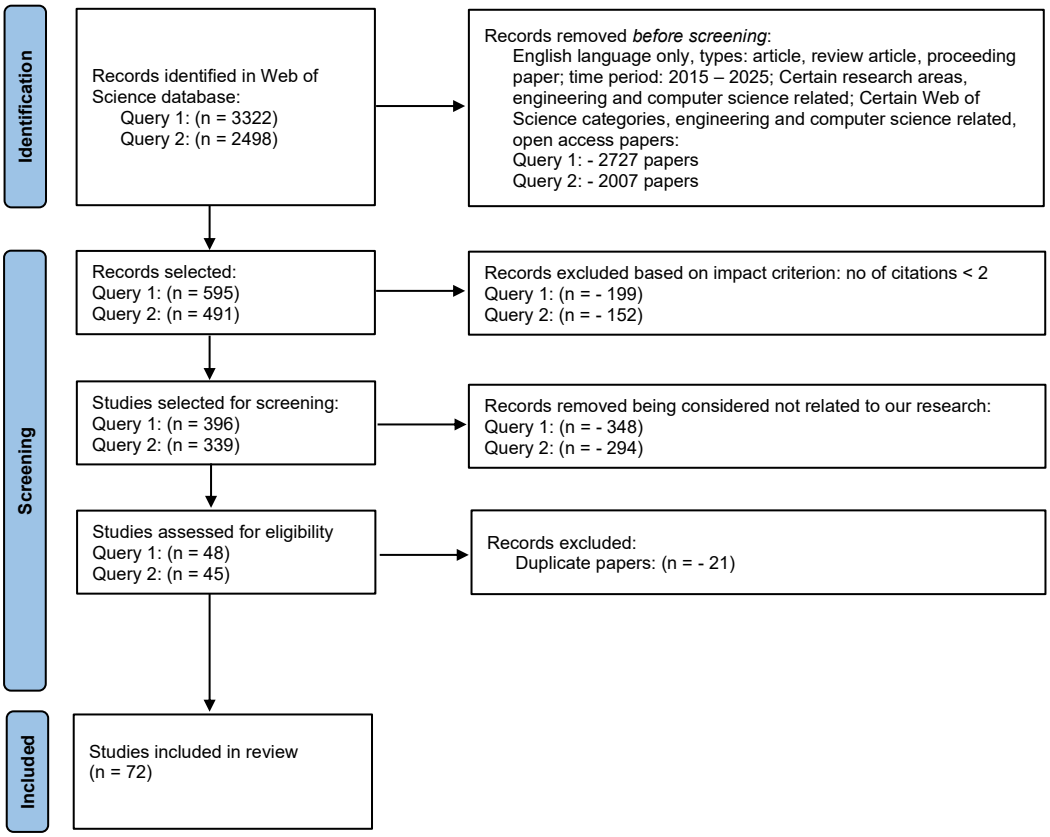

Figure 1. *PRISMA flow diagram of the study selection process.*

To further refine the selection, the search was limited to papers published between 2015 and 2025, focused on specific Web of Science categories (Engineering Electrical Electronic, Computer Science Information Systems, Computer Science Artificial Intelligence, Engineering Multidisciplinary, Telecommunications, Computer Science Software Engineering, Computer Science Interdisciplinary Applications, Multidisciplinary Sciences, Computer Science Theory Methods), and restricted to Open Access publications. This yielded 595 results. Applying a citation threshold of at least two citations further reduced the set to 396 articles. After detailed screening, 48 papers were selected for inclusion. Studies unrelated to the focus of this review—such as those in gaming, physics, medicine, agriculture, chemistry, or accounting—were excluded.

A second search using the terms "artificial intelligence" AND "software quality" retrieved 2,498 records. Applying the exact same filters resulted in 339 articles, from which 45 were considered relevant and included. Again, unrelated studies were excluded based on thematic divergence. After removing 21 duplicate entries from the combined results of both queries, the final set of articles included in this review totalled 72, comprising 46 scientific articles, 18 proceedings papers and 8 review papers.

3.5 Analysis procedure

For each included paper, the following data items were extracted: D1. study type according to Web of Science (WoS) (article/proceedings paper/review), D2. AI technique, D3. testing activity, D4. system under test/domain, D5. datasets/benchmarks and availability, D6. evaluation metrics, D7. evidence setting (laboratory/simulation, secondary evidence, industrial/practitioner, conceptual/guideline, hybrid), D8. threats to validity/limitations discussed, D9. artefacts (code/data availability), D10. main claim. Then, the articles were coded into pre-established themes. Themes were anchored in the Literature Review and in VOSviewer keyword maps.

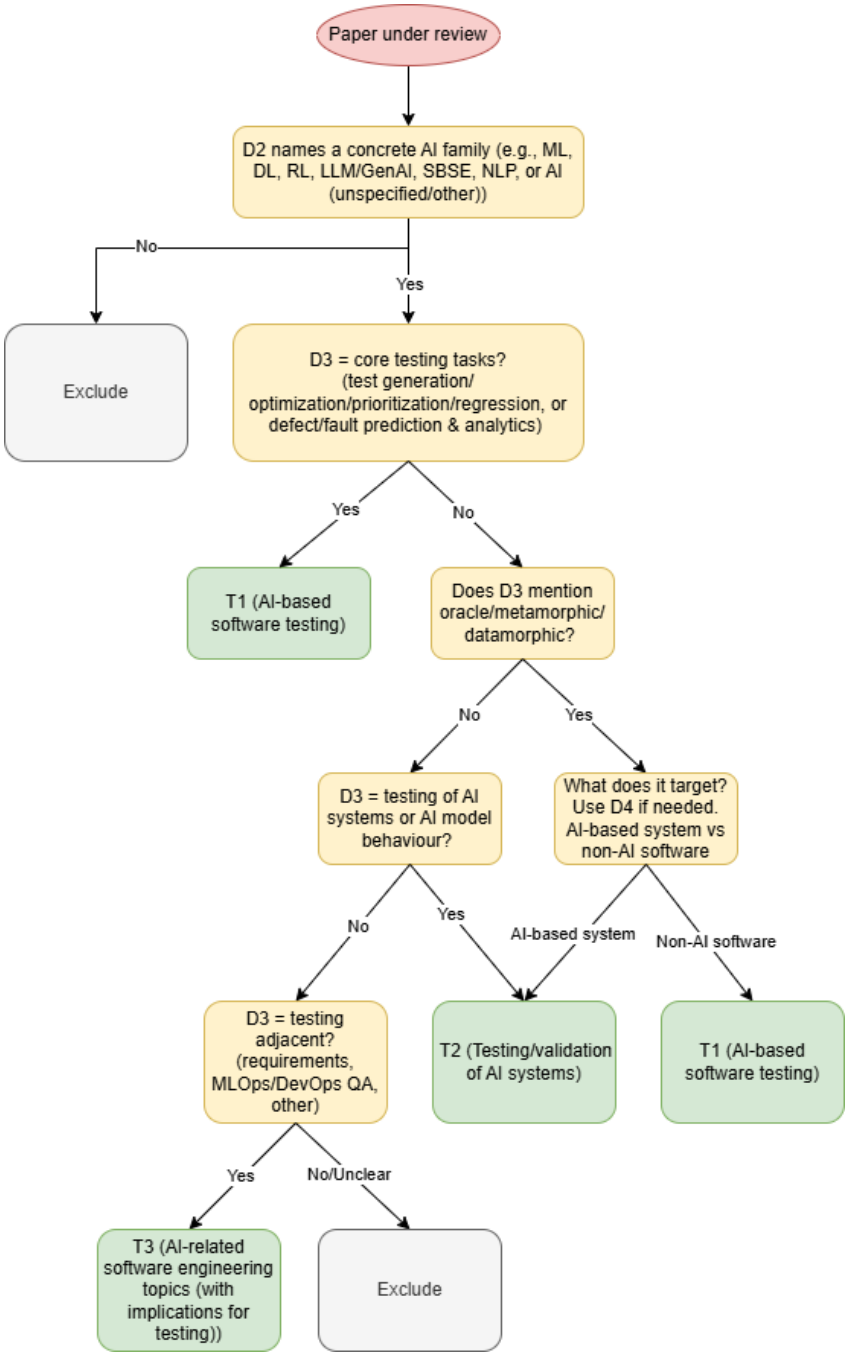

Figure 2. *Flow diagram of the papers themes assignment.*

Predefined themes: T1. AI-based software testing; T2. Testing/validation of AI systems; T3. AI-related software engineering topics (with implications for testing). Two researchers independently applied the deterministic rule set (Figure 2) to assign T1/T2/T3. Disagreements were due to interpretation of borderline cases and were resolved by consensus. The assignment of a paper into a certain theme is detailed in the following flow diagram.

We summarized the papers by data items, categories and themes, in Appendix B.

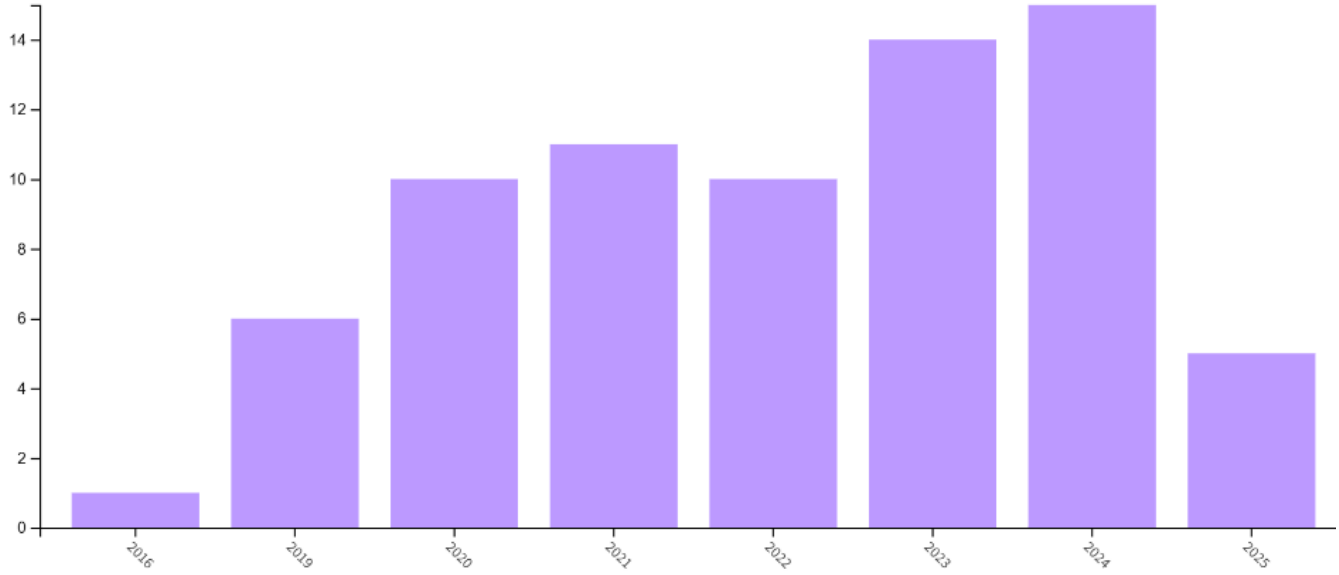

Figure 3. *Final publication years of papers according to WoS.*

Figure 4. *Web of Science categories for the selected papers.*

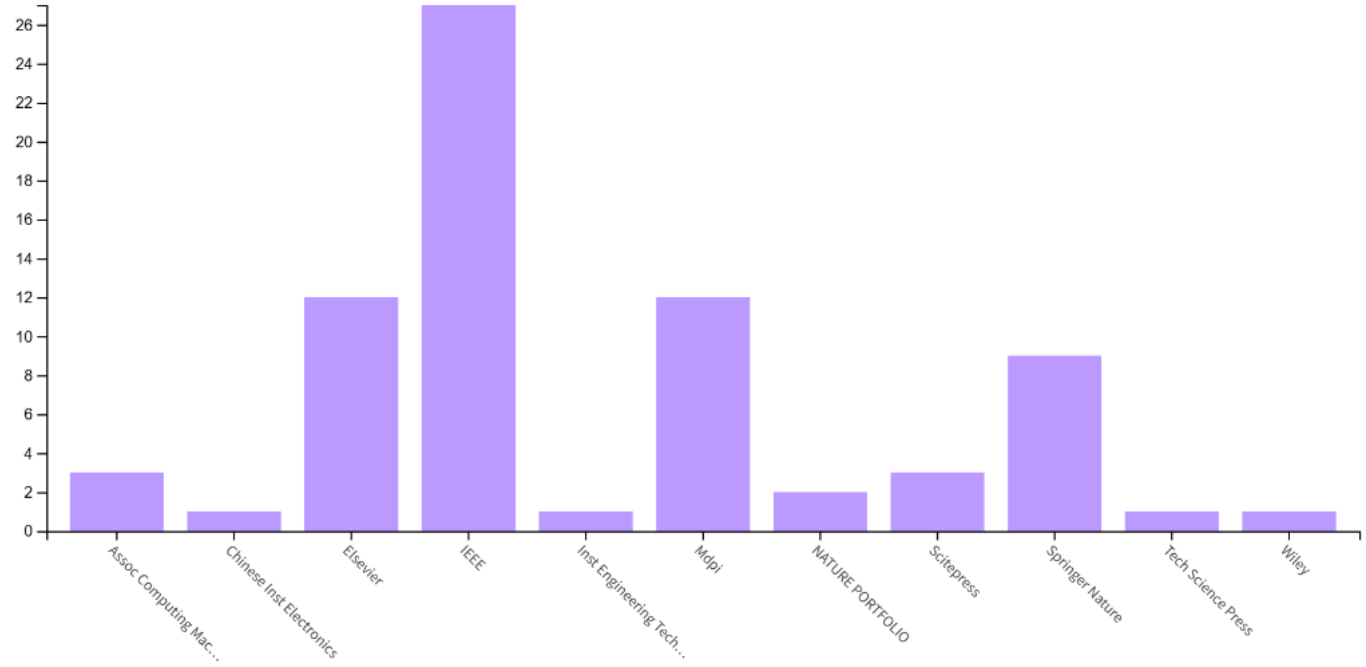

Figure 5. *Publishers' distribution for the selected papers.*

Using VOSviewer software (version 1.6.20) (van Eck & Waltman, 2023), a keyword map was generated to visualize the most significant terms across the 72 papers included in this review, highlighting the primary research themes represented in the literature.

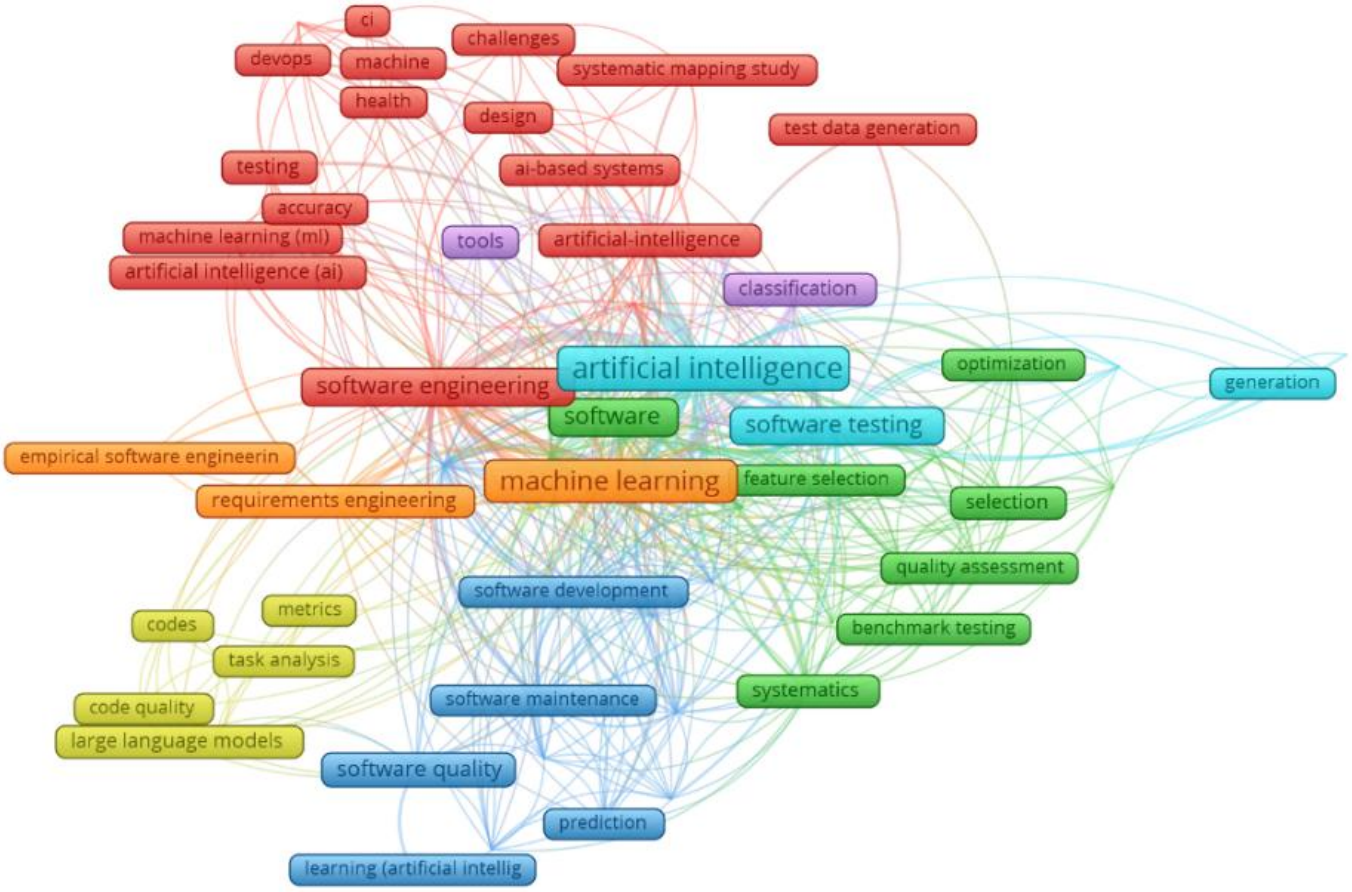

Figure 6. *Keywords map which includes the terms with an occurrence of minimum 2 (67 keywords).*

The VOSviewer keyword map over the 72 papers shows a blue core around “artificial intelligence”, “software testing”, and “generation”, indicating the central AI-for-testing theme. To the right in green sits a method/technique cluster - “optimization“, “selection”, “benchmark testing”, and “quality assessment”, “feature selection” - capturing search-based and data-driven test design and evaluation. The upper-left red cluster aggregates evidence/ML task terms such as “accuracy”, “test data generation”, “ai-based systems”, “machine learning”, “artificial intelligence” and “systematic mapping study” reflecting common ML activities and study types, plus reported challenges. The orange cluster anchors the software-engineering context with “requirements engineering” and “empirical software engineering” linking methods to processes and study design. A yellow cluster highlights emerging GenAI and measurement threads - “large language models”, “code quality”, “metrics”, “task analysis”, and “codes.” Finally, a dark-blue strand near the bottom connects “software development”, “software maintenance”, “software quality”, “prediction”, and “learning (artificial intelligence)”, pointing to quality/maintenance and defect-prediction lines.

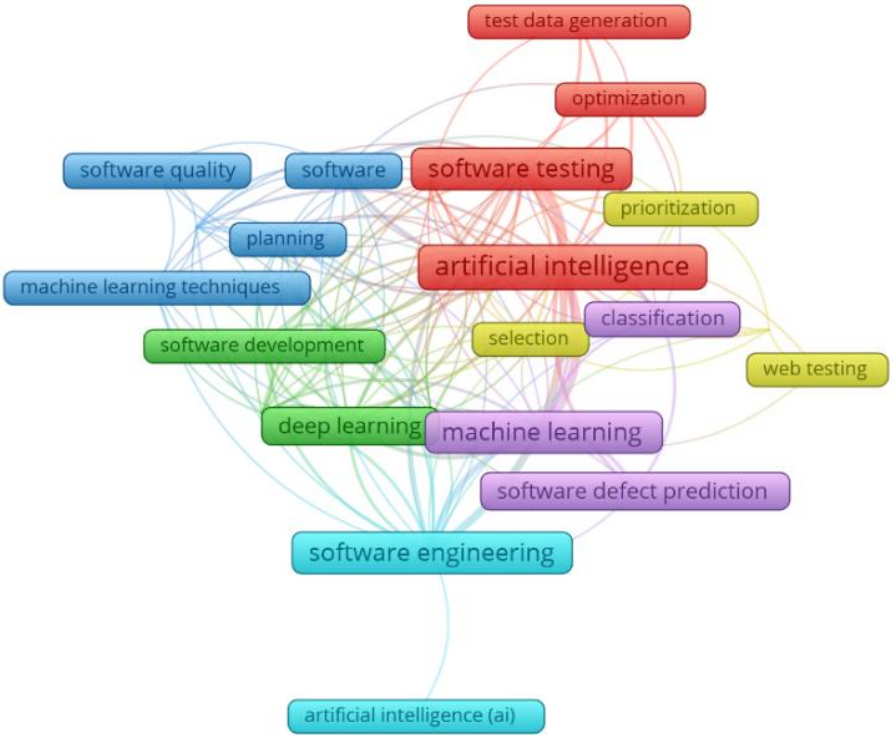

Figure 7. *Keywords map for T1 which includes the terms with an occurrence of minimum 2 (25 keywords).*

For the T1 corpus (AI-based software testing), the VOSviewer map is organized around a red core where “artificial intelligence” and “software testing” dominate, with red extensions to “test data generation” and “optimization” - evidence that automation-oriented tasks drive much of the work. To the right, a yellow cluster groups concrete testing activities, “prioritization”, “selection”, “classification”, and “web testing”, showing where AI is operationalized in practice. Beneath the core, a green cluster anchors the methodological spine - “deep learning”, and links into software development. On the left, a blue cluster, “software”, “software quality”, “planning”, and “machine learning techniques”, situates these efforts within quality objectives and practical planning/technique choices. A small purple node (“software defect prediction”, “classification”, “machine learning”) marks the mature subarea of software defect prediction, while cyan nodes (“software engineering”, “artificial intelligence (AI)”) connect the map to broader SE discourse.

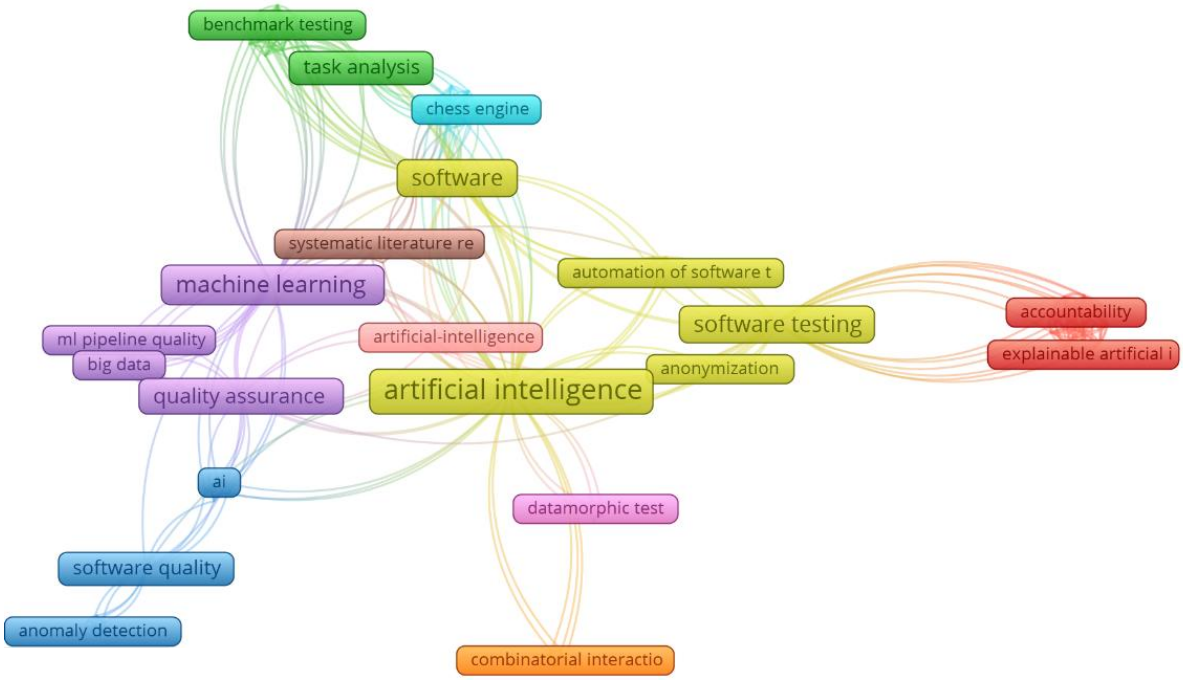

Figure 8. *Keywords map for T2 which includes the terms with an occurrence of minimum 1 (78 connected items out of 100 keywords).*

For the T2 corpus (testing/validation of AI systems), the map is organized around a central hub of "artificial intelligence" linked directly to "software testing" and "machine learning." The yellow cluster anchors practice-oriented nodes, "software testing", "automation of software testing", and "anonymization", showing the operational core of AI assurance. The green cluster groups evaluation contexts and tasks, "benchmark testing", "task analysis". To the left, the purple cluster connects "machine learning" with process and evidence terms, "quality assurance", "ML pipeline quality" and "big data", emphasizing lifecycle and quality concerns. A small blue cluster ("software quality", "anomaly detection", "AI") signals runtime monitoring themes. On the right, the red cluster ("accountability", "explainable artificial") captures governance and transparency requirements that shape validation criteria. Finally, a pink/orange methods cluster ("datamorphic test", "combinatorial interaction") highlights oracle/robustness techniques tailored to AI's data sensitivity and non-determinism. Together, the color clusters indicate that T2 research tightly couples concrete testing practice with ML-pipeline quality and responsible-AI objectives.

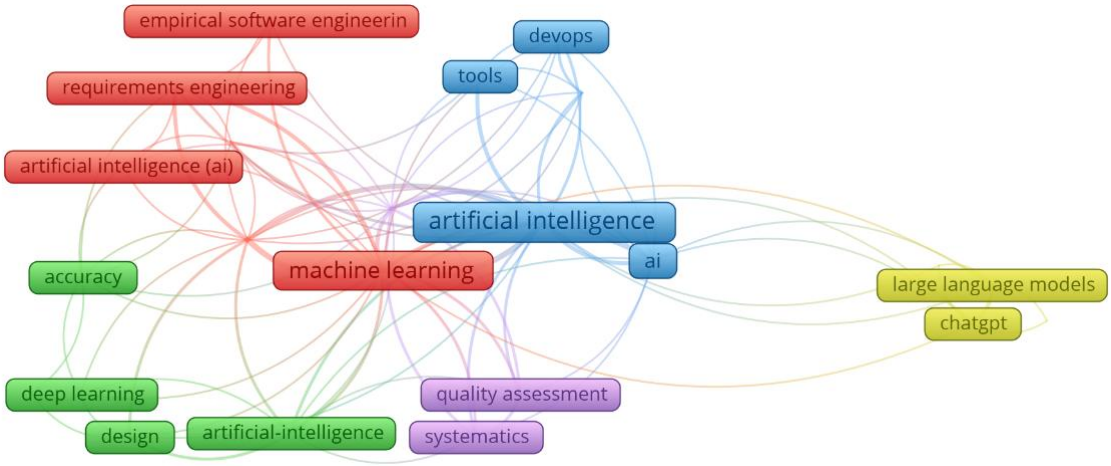

Figure 9. *Keywords map for T3 which includes the terms with an occurrence of minimum 2 (23 keywords).*

For the T3 corpus (AI-related SE topics with implications for testing), the map centers on a blue cluster linking "artificial intelligence", "ai", "tools", and "devops", indicating tooling and lifecycle integration concerns around AI

in engineering workflows. To the left, a red cluster ties "machine learning" to "requirements engineering", "empirical software engineering", and "artificial intelligence (AI)", showing that T3 work often studies ML within RE practices and empirical study designs. A green cluster ("design", "deep learning", "accuracy", "artificial-intelligence") reflects method/architecture discussions that surface measurement and performance considerations relevant to validation. At the bottom-right, a purple cluster ("quality assessment", "systematics") highlights evidence appraisal and research synthesis. On the far right, a yellow cluster ("large language models", "chatgpt") captures emerging GenAI topics that bridge back to the blue tooling/devops nodes and the red ML/RE nodes. The cross-links from the blue hub to red and purple clusters suggest that T3 papers couple AI/ML adoption in SE processes with empirical evaluation and quality assessment, while the yellow nodes signal a fast-growing stream on GenAI's role across requirements, tooling, and operations.

3.6 Quality assessment

To evaluate the rigor of the included studies, quality assessment criteria were applied, such as clarity of objectives, adequacy of methodology, relevance to the field, clear problem, context/system under test described, data/benchmark adequacy, experimental design, threats to validity, practical relevance.

3.7 Methodological limitations

It should be noted that this study has several methodological limitations: reliance on a single database (Web of Science) risks coverage gaps compared with sources like IEEE Xplore, Scopus, and the ACM Digital Library; restricting to Open Access can skew the corpus and omit impactful paywalled studies; limiting to English and applying a citation threshold (≥ 2) may bias against very recent (2024-2025) but relevant works; focusing search terms on "software testing/quality" with "artificial intelligence" may miss synonyms such as "verification & validation", "quality assurance", "QA", "test oracle", "fault localization" or "search-based testing"; relying on Web of Science research areas/categories may inadvertently exclude interdisciplinary or misclassified records; keywords mapping with VOSviewer reflects author-provided terms and is sensitive to synonym choices and threshold settings.

4 RESULTS

Following the protocol in the Methodology section (search inclusion/exclusion, and extraction of D1 – D10), we aggregated the 72 papers and assigned themes (T1 – T3) using the deterministic D2 + D3 rule with a D4 tiebreaker. Below we report the results by themes, then derive recommendations grounded in these counts.

4.1 AI-based software testing

4.1.1 Where AI is used across the testing lifecycle

Test design and automation. Text-driven approaches turn informal artefacts (e.g., natural language requirements, test specifications, bug reports) into test assets or support early QA decisions; recent tertiary/mapping syntheses show steady growth but much of the evidence remains secondary/bibliometric rather than large-scale industrial trials (Amalfitano et al., 2024; Sofian et al., 2022; Durrani et al., 2024; Kokol, 2024). Industrial NLP pipelines have also been used to classify testing constraints—specifically, to label integration tests as dependent versus independent from textual test specifications and then exploit those labels for scheduling/parallelisation (Tahvili et al., 2020). LLM-era reflections emphasise augmentation and a collaborative tester–AI workflow, with testers retaining authority and scope expanding gradually rather than full autonomous substitution (Layman & Vetter, 2024). Comparative empirical evidence on codeless/NLP end-to-end tools is beginning to quantify effort: a controlled case study finds that natural-language testing (NLT) generally yields the lowest cumulative effort across releases versus programmable testing (PT) or capture-and-replay (CRT), with one app-specific exception (Leotta et al., 2022). By contrast, Nguyen & Maag (2020) present a codeless Selenium + ML framework and argue potential cost reductions, but they do not run a head-to-head effort comparison; it is a proposal/architecture with a proof-of-concept task. Recent overviews reinforce the "growing but still maturing" picture across the SDLC and testing: a broad review synthesises AI uses (code gen, defect prediction, testing/prioritization) and stresses integration, trust, and ethics gaps alongside productivity claims (Alenezi & Akour, 2025). A lifecycle mapping likewise finds testing has the most ML publications, offering quantitative context for where AI activity clusters (Navaei & Tabrizi, 2022). **Empirical**

evaluation of LLM-generated code compares engines using **test-pass rates and static quality metrics**; results vary across engines, underscoring the need for **expert review and robust test suites** (Tosi, 2024).

Test generation, optimization, prioritization and regression. Search/metaheuristic methods remain prevalent in test-suite optimization for regression testing: an SLR of 58 studies (2016–2019) finds metaheuristics the largest block and recommends moving beyond single-objective setups toward multi-objective formulations (also mapping tools and platforms) (Kiran et al., 2019). A primary study applying the Firefly algorithm on three SIR (Software-artifact Infrastructure Repository) benchmark programs reports APFD (Average Percentage of Faults Detected) and execution-time improvements versus prior techniques (Khatibsyarbini et al., 2019). Emerging work explores learning-to-rank TCP (Test Case Prioritization) with explainability (global + local), including a preliminary experiment showing that explanations vary with predicted ranks (Ramirez et al., 2023). In industry, lean RTS (Regression Test Selection) practices with human oversight are advocated: at Volkswagen, ML assists regression-test selection while the test manager remains the decision-maker (Poth et al., 2019). For AI-assisted RTS, an ethical checklist and framing identify core risks—responsibility, bias in decision-making, and participation—and propose explicability, supervision, and diversity as mitigations, drawing on experience with an industry AI-RTS tool (Strandberg et al., 2021).

Graphical user interface (GUI) or user experience (UX) and exploratory testing. Affective testing agents can augment UX/playtesting and run alongside automated functional tests, offering continuous UX signals rather than replacing user studies; a proof-of-concept shows agents with a core-affect model (valence/arousal) traversing game levels to surface UX-relevant patterns; **evidence is conceptual/proof-of-concept rather than large-scale benchmarking** (Fernandes et al., 2021). At a broader scope, RL-based exploratory testing is a prominent RL application area in software testing—especially for GUI/mobile/web contexts—according to a recent SLR that maps RL usage across testing tasks and highlights GUI exploration as a major cluster (Abo-eleneen et al., 2023).

Security testing and penetration testing. Deep learning has advanced malware detection across static, dynamic, and image-based pipelines, with multiple experiments showing DL models outperforming classical ML baselines; the paper also proposes a scalable, hybrid, near–real-time framework (ScaleMalNet) and explicitly flags issues like dataset bias/imbalance and the need for independent evaluation and scalability (Vinayakumar et al., 2019). In connected-vehicle security, reinforcement learning (Q-learning) has been used to automate penetration-testing scenarios in a simulated VANET/CAV testbed; results are promising, showing efficiency gains (higher mean rewards) and, in at least one scenario, reduced runtime (fewer steps), though benefits vary by scenario rather than uniformly improving all coverage metrics (Garrad & Unnikrishnan, 2023).

Defect prediction and quality analytics. A 49-study SLR on FS (feature selection) for software defect prediction (SDP) reports that filter and hybrid FS approaches are most common, paired with NB/SVM/DT and ensemble learners (RF, Bagging, AdaBoost); AUC/Accuracy/F1 dominate as evaluation metrics, and WEKA/MATLAB/Python are the typical toolchains (Ali et al., 2023). A bibliometric mapping of "intelligent software testing" (2012–2022) charts rapid growth, influential venues, and collaboration patterns—useful as landscape context rather than effectiveness evidence (Boukhlif et al., 2023). Moving from handcrafted metrics toward code-semantics learning, an AST-aware LSTM outperforms baselines on several OSS projects, while noting external-validity limits from PROMISE's Java-only scope (Deng et al., 2020). For deployment, DePaaS proposes a cloud architecture that unifies SDP practice and research (use cases, components, recommended models) and reports initial practitioner feedback—vision/feasibility rather than benchmark evidence (Pandit et al., 2022). A comprehensive bibliographic survey (2011–2023) synthesises datasets, validation practices, and tools, flagging label scarcity and inadequate validation as recurring blockers and recommending multi-label formulations and stronger evaluation protocols (Mustaqeem et al., 2025). Finally, a DL + metaheuristic hybrid—Residual/Shuffle network tuned with an upgraded Fish Migration Optimization—reports 93% mean accuracy and improved Precision/Recall/F1/AUC across multiple OSS datasets, while acknowledging OSS-only threats to external validity (Liu et al., 2025). **An IDE copilot with context-based RAG reports gains in bug-detection accuracy, critical-coverage, user acceptance, and time/bug, illustrating defect-analytics–oriented assistance within testing workflows** (Wang et al., 2025).

Bug triage and maintenance. Topic-model–enhanced pipelines that improve latent Dirichlet allocation (LDA) itself (multiple-LDA with backpropagation) raise bug triage accuracy on Bugzilla and Android datasets and are designed to remain compatible with hybrid LDA methods (validated with a paired t-test) (Lee & Seo, 2020). At the process level, SQAPlanner turns defect-prediction models into actionable, rule-based quality guidance with explicit risk

thresholds, helping teams craft improvement policies; **over 32 releases from 9 OSS projects,** it reports strong coverage/confidence, stability, and positive practitioner feedback (Rajapaksha et al., 2022). Beyond triage, LLMs are being piloted to generate patch sets from code-review comments: an in-progress study compares multiple LLMs' patches to historical human patches and evaluates accuracy, relevance, and actionability, highlighting the immaturity of this niche and the need for further evidence (Rahman et al., 2024).

Oracles, mutation, and API validity. When oracle construction is costly, deep models can cut effort in adjacent cost drivers. In mutation testing of Android apps, an AST-plus-TBCNN model automatically classifies equivalent mutants, reporting 94% accuracy (recall 96%, precision 89%, F1 92%) on a MutantBench-based dataset—thereby reducing one of mutation testing's main costs before mutation-score computation (Kusharki et al., 2022). For RESTful systems, a **deep learning classifier** trained on prior requests/responses **predicts whether a request is valid** (i.e., satisfies inter-parameter dependencies) **without invoking the API;** evaluated **on five real-world APIs** with **public code and dataset**, results show **86–100% accuracy**, which can steer test generation toward valid inputs (Giuliano et al., 2021).

4.1.2 How AI does it (technique-centric view)

Search-Based Software Engineering (SBSE). Metaheuristics (e.g., GA/PSO/ACO/Firefly) remain widely used for test-suite optimization. A trend/tool survey highlights the dominance of metaheuristics and recommends moving from single objective to multi-objective formulations with consistent metrics and reusable artefacts (Kiran et al., 2019). As a representative primary study, Firefly-based TCP improves APFD and execution time on SIR benchmarks versus prior techniques (Khatibsyarbini et al., 2019). Overall, comparative gains are common, but secondary works call for stronger methodological uniformity and replication packages (Kiran et al., 2019).

Machine learning (ML) and deep learning (DL)

Defect prediction. Moving from hand-engineered metrics to code-semantics learning, LSTM models over AST/sequence features outperform baselines on several OSS projects, while noting external-validity limits (Java-heavy corpora) (Deng et al., 2020). Mapping/survey papers consolidate datasets, validation practices, and toolchains, but repeatedly flag label quality, cross-project generalisability, and inconsistent evaluation as pain points (Ali et al., 2023; Mustaqeem et al., 2025; Sofian et al., 2022). To ease adoption, a conceptual DePaaS architecture sketches how to operationalise SDP as a cloud service (Pandit et al., 2022). Hybrid approaches that tune deep models with metaheuristics (e.g., Residual/Shuffle networks optimised by an upgraded Fish Migration Optimization algorithm) report high accuracy and strong Precision/Recall/F1/AUC on multiple OSS datasets, while still being OSS-only (Liu et al., 2025).

Security and malware detection. CNN/LSTM (and hybrids) over static, dynamic, and image-based program representations improve malware detection against classical ML, with scalability prospects but also risks around dataset bias/imbalance and the need for independent validation (Vinayakumar et al., 2019).

API input validity and GUI/UX analytics. Deep classifiers can predict REST input validity from prior request/response traces and guide generation toward valid inputs (Giuliano et al., 2021). Affective, agent-based UX testing demonstrates how automated playtesting can surface UX-relevant patterns; the evidence is conceptual/proof-of-concept rather than large-scale (Fernandes et al., 2021).

Bug triage. Topic-model-informed pipelines (multiple-LDA with backpropagation) improve triage accuracy on public Bugzilla/Android corpora, remaining compatible with hybrid LDA methods (Lee & Seo, 2020).

Reinforcement Learning (RL). An SLR maps RL across GUI/exploratory testing, test generation/selection, and related validation tasks, and notes hurdles such as environment design and training cost (Abo-eleneen et al., 2023). In security-focused simulation, Q-learning increases efficiency for VANET/CAV penetration-testing scenarios (higher mean rewards, fewer steps in at least one scenario), with benefits varying by scenario (Garrad & Unnikrishnan, 2023).

Natural Language Processing (NLP) and Large Language Models (LLMs). NLP turns textual artefacts (requirements, test specs) into structured signals for testing—e.g., classifying integration tests via imbalanced-learning pipelines on proprietary corpora (Tahvili et al., 2020)—and underpins "codeless" authoring. Comparative evidence suggests NL-based testing often lowers cumulative test-maintenance effort versus programmable or

capture-and-replay, with app-specific exceptions (Leotta et al., 2022). Other codeless frameworks pair Selenium with ML for feasibility proofs rather than head-to-head baselines (Nguyen & Maag, 2020). Experience pieces position LLMs as copilots that augment—rather than replace—testers (Layman & Vetter, 2024). Prototype IDE integrations with retrieval-augmented generation report gains in bug-detection accuracy, critical-coverage, and user acceptance while coupling code-gen, bug-finding, and test generation (Wang et al., 2025). Still, empirical comparisons of code produced by different engines show meaningful variance in test pass rates and static quality metrics, reinforcing the need for expert review and robust test suites (Tosi, 2024). Early work also evaluates LLM-generated patches from review comments against historical human patches, underscoring the nascence of this niche (Rahman et al., 2024).

Cross-cutting maturity signals. Broad SE overviews agree that AI techniques permeate testing, but evidence quality is uneven, with many secondary/bibliometric syntheses and fewer large-scale industrial trials; integration, trust, and ethics (including explicability and oversight) remain active concerns (Alenezi & Akour, 2025; Durrani et al., 2024; Sofian et al., 2022; Kokol, 2024).

4.1.3 How well it works (validation-centric view)

Across the corpus, evidence spans tertiary/secondary syntheses (Amalfitano et al., 2024; Boukhlif et al., 2023; Sofian et al., 2022; Kokol, 2024; Durrani et al., 2024; Mustaqeem et al., 2025), laboratory/simulation studies (e.g., Firefly TCP on SIR: Khatibsyarbini et al., 2019; RL for VANETs: Garrad & Unnikrishnan, 2023; DL for REST input validity: Giuliano et al., 2021), hybrid/feasibility evaluations (e.g., codeless/NLP frameworks and comparative effort studies: Nguyen & Maag, 2020; Leotta et al., 2022; IDE copilot with RAG: Wang et al., 2025; quality-plan operationalisation: Rajapaksha et al., 2022), and industrial evidence from proprietary corpora (e.g., NLP-based test classification on industrial specs: Tahvili et al., 2020; practitioner-driven lean QA with AI assistance: Poth et al., 2019). Common threads across reviews include heterogeneous metrics/datasets hindering cross-study comparison, limited multi-site replications, and gaps in non-functional evidence (Amalfitano et al., 2024; Kiran et al., 2019; Sofian et al., 2022; Mustaqeem et al., 2025).

Representative comparative signals:

- **Prioritization/selection**. Firefly frequently improves APFD and runtime over baselines on SIR programs (Khatibsyarbini et al., 2019). Learning-to-rank TCP with explainability shows promise and makes prioritization decisions more interpretable (Ramirez et al., 2023).
- **Mutation testing**. AST + TBCNN achieves 94% accuracy (high recall/F1) for equivalent-mutant classification in Android, cutting mutation-analysis cost drivers (Kusharki et al., 2022).
- **Defect prediction**. LSTM models leveraging code semantics outperform classic metric-based baselines on several OSS projects (Deng et al., 2020). Recent deep + metaheuristic hybrids report 93% accuracy on selected OSS sets (Liu et al., 2025). Surveys emphasise the impact of feature selection (Ali et al., 2023) and the need for stronger validation protocols and curated corpora (Mustaqeem et al., 2025).
- **Security/malware**. DL pipelines outperform classical ML across static/dynamic/image representations, with reproducibility and data imbalance as recurring issues (Vinayakumar et al., 2019).

4.1.4 Adoption, governance, and ethics

Experience/position works advocate human-in-the-loop AI testing with clear accountability, explicability, and oversight—especially for RTS and safety-sensitive contexts (Poth et al., 2019; Strandberg et al., 2021). Comparative studies indicate cost/maintainability trade-offs between NLP/codeless and programmable approaches (Leotta et al., 2022; Nguyen & Maag, 2020). Prototype RAG-style copilots suggest tighter tester-IDE integration across code generation, bug finding, and test generation (Wang et al., 2025), while LLM patchers explore code-review workflows but remain early-stage (Rahman et al., 2024). Tertiary/secondary syntheses caution that industrial uptake lags publication growth and call for shared artefacts, replications, and transparent reporting (Amalfitano et al., 2024; Durrani et al., 2024; Sofian et al., 2022; Kokol, 2024; Boukhlif et al., 2023).

4.1.5 Overall evaluation

Evidence for T1 shows consistent gains on classic testing tasks (e.g., test generation, prioritization, oracle support) with measurable improvements on common metrics (APFD/APFDc, F1/precision–recall, runtime/CPU). The

strongest results appear in controlled or benchmarked settings; translation to large, heterogeneous systems remains uneven.

Strengths. Coverage spans the lifecycle, from requirements-/text-driven test design to regression/prioritization in CI, with mature families (SBSE/optimization, ML/DL, NLP/LLM) addressing complementary tasks. Comparable, task-specific metrics are frequently used (APFD/APFDc, fault-detection rate, mutation score, defect-prediction metrics), enabling at-a-glance comparisons within task clusters. Automation and productivity gains are reported through NLP/LLM pipelines for artefact mining (requirements, bug reports, API specs) and SBSE variants that explore large test spaces. Several studies show early toolchain fit via CI/CD integration (batch or nightly), including practical hooks for scheduling, selection, and flaky-test handling. Transparency is improving as an increasing share of works provide code/data or recipes, supporting auditability and reuse.

Weaknesses. External validity gaps persist, as many results rely on small benchmarks or single-project corpora; cross-project and cross-domain evidence is limited, and multi-site replications are rare. Metric and dataset heterogeneity (datasets, seeds, stopping criteria) hinders strict apples-to-apples comparisons and meta-analysis. Oracle and reliability issues—weak oracles, test flakiness, and non-determinism (especially with ML/LLM components)—inflate variance and threaten reproducibility. Non-functional blind spots remain, with scalability, robustness, performance overhead, and cost-of-quality impacts under-reported relative to functional improvements. Operational complexity is also a barrier, as methods with many hyperparameters or heavy compute (deep models, meta-heuristics at scale) face tuning debt and runtime constraints in production.

Scalability levers (from laboratory to industrial use). Standardize evidence and artefacts by adopting shared, versioned benchmarks per task, fixing seeds, reporting budget/compute, and releasing minimal-viable artefacts (configurations, scripts, container files) alongside threats-to-validity checklists. Engineer for CI/CD reality with incremental and budget-aware variants (time/CPU caps, warm-start, caching) to keep pipelines green, and log actionable by-products (prioritization rationale, failure clusters) to aid triage. Tame flakiness and non-determinism through deterministic seeds, quarantine policies, repeated trials with confidence intervals, and metamorphic checks for stability. Use human-in-the-loop by design by exposing explanations, safe-defaults, and reversible actions, and measure reviewer effort saved, not only APFD. Report cost and risk explicitly (runtime/infra costs, tuning effort, failure modes) and track regressions on non-functional requirements. Finally, strengthen generalization protocols with cross-project/time-split evaluations and ablations (data size, noise, domain shifts), and include governance and observability (dashboards/telemetry for drift, flaky-rate, and model/test health, plus roll-back and kill-switch procedures).

4.2 Testing/validation of AI systems

Research on validating AI systems spans complementary families—metamorphic (and datamorphic) testing, trial/simulation, model-centred checks, and expert-opinion/operational monitoring—and is often discussed alongside MLOps/CD4ML delivery. Evidence is strongest for ML-based software (especially computer vision), with growing attention to autonomous/decision-policy systems and formalized test knowledge.

4.2.1 Metamorphic and datamorphic testing

Metamorphic testing (MT) addresses the oracle problem by asserting relations that should hold when inputs are transformed. For unsupervised ML, METTLE defines 11 generic metamorphic relations (MRs) as adequacy criteria to assess clustering systems from an end-user perspective and demonstrates viability via experiments and a user study (Xie et al., 2020).

In computer vision, Tao et al. discuss validating a commercial Facial Age Recognition API using two MR groups—MR1 (mirror, rotation with size/color fixed) and MR2 (translation, cropping "tailor", scaling with color fixed)—and report robustness trends (e.g., larger rotations and non-uniform scaling degrade performance), presented as a case-style illustration rather than a public benchmark release (Tao et al., 2019).

For decision-policy software, MT has been applied to an AI chess engine, crafting MRs over piece-movement legality and over algorithmic behaviour (e.g., alpha–beta pruning/evaluation). With error seeding, the approach detected approximately 71% of injected faults and highlighted differences from the standard perft check—evidence MT can expose issues beyond simple output counts (Liaqat et al., 2020).

Datamorphic testing complements MT by systematizing input-space exploration through structured data transformations and partial oracles. Zhu & Bayley formalize strategies to discover class boundaries (e.g., near decision borders) and evaluate them on real ML models, with algorithms implemented in the Morphy tool (Zhu & Bayley, 2022).

A paper on Automation of Datamorphic Testing further details Morphy's architecture—test entities and test morphisms (seed makers, datamorphisms, metamorphisms, metrics/filters, executors, analyzers)—and shows how strategies and full processes can be composed and replayed to scale campaigns from "test intent - operators - oracles/relations", including a public examples repository (Zhu et al., 2020).

MT (and datamorphism) provides oracle-free robustness checks aligned with realistic data operations (rotation/scale/translation/crop), supplies user-centric adequacy criteria for unsupervised learning (METTLE), and enables automation at scale for boundary-seeking exploration (Morphy). Evidence across clustering, facial-age recognition, and game-AI shows strong fault-finding and actionable robustness diagnostics where ground truth is absent or costly (Xie et al., 2020; Tao et al., 2019; Liaqat et al., 2020; Zhu & Bayley, 2022; Zhu et al., 2020).

The MT/datamorphic studies specify transformations and MR rationales and quantify effects with concrete metrics where applicable (e.g., adequacy/effectiveness measures in METTLE; seeded-fault detection rates in the chess engine), while Tao et al. (2019) remain a conceptual/case-style illustration rather than a public benchmark. Internal validity is strengthened by explicit MRs and controlled manipulations; external validity is often limited to the target systems/domains. Morphy operationalizes automation with reusable morphisms, strategies (including boundary-seeking), and record/replay scripts—a path to scale—yet published evaluations are still case-study/controlled-experiment level rather than cross-domain industry trials.

4.2.2 Trial, simulation, and system-level validation

A systematic review synthesizes a four-way taxonomy—trial, simulation, model-centred validation, and expert opinion—and catalogues continuous-validation mechanisms used after deployment (failure monitors, safety channels, redundancy, voting, input/output restrictions) (Myllyaho et al., 2021). When multi-method papers are partitioned, trials occur 55 times and simulations 31 times across 90 studies, while only 14/90 papers report any form of continuous validation, indicating a runtime-assurance gap (Myllyaho et al., 2021). Simulations are especially popular in cyber-physical domains such as cars and robots (Myllyaho et al., 2021).

For multi-agent autonomous intelligent agents (AIAs), a dedicated test-and-evaluation framework proposes: hierarchical test planning from local agent to global system, combinatorial interaction testing and design-of-experiments to control interaction explosion, and continuous evaluation as agents learn and environments change—explicitly addressing emergent behaviour and finite test budgets; an illustrative satellite-constellation use case outlines the process (Lanus et al., 2021).

The review quantifies where evidence clusters (trials/simulations) and how rare continuous-validation reporting remains—constraining external validity for long-lived AI (Myllyaho et al., 2021). Simulation scales exploration for robotics/AV-like settings but risks sim-to-real gaps if models are imperfect. The AIAs framework offers conceptual guidance with an illustrative use case; broader empirical instantiation remains future work (Lanus et al., 2021).

4.2.3 Model-centred validation and expert/knowledge-driven support

Model-centred validation (e.g., benchmarks, "torture tests", stress tests) appears less frequently than trials/simulations; expert-opinion validations are rare and typically lighter-weight methodologically (Myllyaho et al., 2021).

Ontology-driven test knowledge. The AI-T ontology integrates software-testing processes with ethical/XAI constructs to guide testing activities and documentation. The paper reports an ontology of 708 terms and 706 axioms and demonstrates implementation in OWL/Protégé with FaCT++ reasoning, including examples for well-being assessment and process support (Olszewska, 2020).

Privacy-by-tests. Test-Driven Anonymization (TDA) brings privacy-by-design into AI pipelines: specify privacy constraints (e.g., k-anonymity) and functional-suitability goals as tests; iterate anonymization until both pass. The

approach is evaluated on two real datasets (healthcare, health-insurance) with trade-offs measured via standard classification metrics (Augusto et al., 2019).

Model-centred checks and expert-opinion provide early, focused feedback but need system-level evidence for external validity (Myllyaho et al., 2021). AI-T contributes conceptual and implemented knowledge-engineering artefacts with ontology metrics and reasoning demos—useful for standardization, yet not a comparative outcome study across projects (Olszewska, 2020). TDA offers replicated case-study evidence in two domains with clear procedures and metrics; cross-domain replications would further strengthen generalizability (Augusto et al., 2019).

4.2.4 Pipeline-aware validation

Validation is moving into continuous delivery for ML. A multivocal literature review of 151 sources plus nine semi-structured interviews consolidate pipeline terminology and lays out four stages—Data Handling - Model Learning - Software Development - System Operations—along with event triggers (alerts, repository changes, orchestration/schedules, manual gates) that should kick off tests and quality checks (Steidl et al., 2023). The study details trigger types and reports practice patterns from interviews, noting that many pipelines remain partly manual and feedback-looped rather than linear (Steidl et al., 2023).

Meta-testing inside the pipeline. The AIQ Meta-Testbed vision proposes running meta-experiments inside an MLOps pipeline to evaluate ML testing techniques: controlled fault injection, adequacy-driven test input generation (including search-based, GAN synthesis, metamorphic relations), a control panel to toggle methods, and dashboards for results—primarily targeting offline ML testing before deployment (Borg, 2021).

The MLR's evidence base mixes peer-reviewed and grey literature but is triangulated by interviews, yielding a defensible taxonomy and trigger map; it argues for staged (offline - online) and automated validation in practice (Steidl et al., 2023). AIQ is a position/vision with concrete components but prospective results—useful for scaling comparative evidence, pending broader instantiation (Borg, 2021).

4.2.5 Security and code-quality validation for AI software

AI applications also inherit traditional software risks from frameworks and glue code. A study of TensorFlow-based DL applications analyzes 100 CVE (Common Vulnerabilities and Exposures) instances across six CWE (Common Weakness Enumeration) categories (e.g., NULL dereference, integer overflow, improper input validation), inspects pre-/post-fix files, and evaluates three static analyzers (Cppcheck, Flawfinder, Visual Code Grepper)—finding very low effectiveness at detecting known vulnerabilities and that many issues stem from missing/incorrect checks. The authors provide a checklist of defensive practices for C/C++ numerical/DL libraries (Filus & Domanska, 2023).

Using ground-truthed CVEs and paired vulnerable/fixed files strengthens internal validity; however, generalization beyond TensorFlow needs replication on other stacks. Still, the findings underscore that validation for AI systems must cover ML code + glue code + configuration, not only models (Filus & Domanska, 2023).

4.2.6 Cross-cutting perspectives and research maturity

Synthesis work proposes quality models and challenge lists that span artefacts (data/model/system), processes (isolated versus continuous), and quality characteristics (e.g., correctness, robustness, fairness, interpretability), and explicitly flags hurdles such as oracles/metrics, non-functional properties, self-adaptation, and dynamic environments as central to validation requirements (Felderer & Ramler, 2021).

Tooling that connects literature-style evidence with model/data debugging is also emerging. MLVAL is an Eclipse plug-in that lets developers inspect training data, derived features, and model behaviour, compare model versions, and reproduce anomalies; its evaluation uses large internal document corpora and qualitative/walkthrough evidence that MLVAL helps investigate feature–label connections and detect errors early (Cheng et al., 2023).

Broader QA overviews for ML in big-data systems chart the growth of QA activity, organize techniques along the ML pipeline, and foreground the need to bring responsible/ethical qualities (e.g., fairness, privacy, explainability) into concrete testing patterns and processes (Ogrizovic et al., 2024).

4.2.7 Overall evaluation

Strengths. MT/datamorphic testing offers scalable non-oracle checks for ML behaviour where ground truth is scarce; trial/simulation taxonomies help choose validation modes across domains and lifecycles; pipeline-aware validation integrates checks where changes actually happen (data - model - software - operations); security/code-quality studies demonstrate real faults in AI stacks and the need to test framework/glue code, not just models.

Weaknesses. Many studies are single-system or domain-specific, limiting external validity; continuous validation remains under-reported relative to offline testing; expert-opinion validations tend to be methodologically light unless paired with trials/simulations; ontology/privacy-by-tests proposals need comparative outcome evidence across projects to gauge effect sizes at scale.

Scalability levers. Automate MR/datamorphic operator pipelines and boundary-seeking strategies in validation tools; couple offline adequacy-driven tests with online monitoring/experimentation (A/B, drift monitors, safety channels); adopt meta-testbeds/benchmarks to compare techniques under realistic constraints; and push security/code-quality checks into CI/CD alongside data/feature validation in pipeline stages.

4.3 AI-related software engineering topics (with implications for testing)

4.3.1 Requirements, ethics and design

Human-centered AI requirements. Two companion papers identify concrete gaps (e.g., unclear data needs, non-determinism, stakeholder understanding) and provide a structured RE framework with a worked case to improve the testability of non-functional expectations in AI systems (Ahmad et al., 2023a; 2023b). The framework organizes concerns across the lifecycle and makes room for verifiable quality attributes (requirements phrased with measurable acceptance criteria), while the practices/gaps study empirically surfaces where organizations struggle (e.g., documenting data provenance, monitoring model drift) and what to operationalize in validation (Ahmad et al., 2023a; 2023b).

Ethics as requirements. An ontology-driven perspective turns ethical qualities—such as explicability, autonomy, beneficence, and non-maleficence—into first-class, requirement-like concepts that can guide design-time decisions and corresponding test documentation, helping teams trace these concerns into validation activities (Guizzardi et al., 2023).

Design-level classification to pattern/test guidance. Supervised learning over requirement/design texts can recommend interaction patterns: in one comparative study, a linear SVM performed best among the tried classifiers (versus logistic regression, multinomial Naïve Bayes, and random forest) for the authors' dataset, suggesting a practical route to steer designs—and the tests they imply—based on predicted interaction categories (Silva-Rodriguez et al., 2020).

NLP for RE and RE - test traceability. A recent SLR maps how NLP supports RE tasks (information extraction, classification, traceability) and outlines a pipeline that can feed downstream test-case derivation and requirements-to-test links—useful scaffolding when auditable validation chains are needed (Necula et al., 2024).

Ambiguity detection. Ambiguous requirements are a major source of false positives downstream. Evidence on supervised ML for ambiguity detection indicates reductions in flagged ambiguity, offering a tangible lever to stabilize later testing (Izhar et al., 2025).

Prioritization trade-offs. A systematic review of AI-based requirements prioritization (RP) compares fuzzy/optimization/ML families: fuzzy methods tend to scale and are affordable but often lack accuracy auditing; optimization approaches can be accurate yet slow and prone to redundancy at large N; ML techniques report strong scalability and efficiency but show uneven accuracy reporting—implications that propagate to regression-test selection and maintenance (Anwar & Bashir, 2023).

Service-oriented design choices and testability. At architecture/design level, AI techniques support SOA decisions (discovery, composition, development) with attention to quality attributes; this can improve modularity and interface clarity, indirectly reducing brittleness in integration/system tests (Rodriguez et al., 2016).

Evidence and scalability—balanced view. Much of the RE-side literature is SLR/conceptual or single-case-useful for structuring validation but lighter on large-scale industrial trials (Ahmad et al., 2023a; 2023b; Necula et al., 2024). Algorithmic studies show feasibility (e.g., SVM-based classification; ML ambiguity detectors) yet leave throughput, drift handling, and CI/CD integration under-reported (Silva-Rodriguez et al., 2020; Izhar et al., 2025). For prioritization, the SLR's comparative tables clarify when each technique family tends to scale or falter, which is actionable for planning validation budgets and regression strategies (Anwar & Bashir, 2023).

4.3.2 Coding, review and maintainability

LLM-assisted refactoring—data clumps. An AI-driven refactoring pipeline leverages an LLM (ChatGPT) with a human-in-the-loop to identify and refactor data-clump smells in Git repositories; preliminary experiments report promising results and argue that reducing data-clumps can improve maintainability (with implications for testing complexity), while noting challenges and regulatory considerations (EU AI Act) (Baumgartner et al., 2024).

AI-assisted code review. AICodeReview is an IntelliJ IDEA plugin that uses GPT-3.5 to analyze code and suggest fixes; the paper details architecture and a preliminary validation, proposing that AI support can streamline review effort and surface syntax/logic issues (open-source plugin + reproducible capsule) (Almeida et al., 2024).

Code-quality comparisons of LLM tools. On 60 LeetCode problems (Python), ChatGPT and Copilot produced acceptable solutions for easier tasks but were prone to syntax/semantic errors; small, non-monotonic quality improvements across iterations and the need for human oversight were observed (Nikolaidis et al., 2024). On a 25-task synthesis benchmark (C++/Java), ChatGPT delivered 17 perfect solutions vs 13 for Copilot; static/non-functional analysis found "good quality code" from both, with characteristic smells, and human reviewers supported the comparative outcome (Sagodi et al., 2024).

Provenance detection for risk-based testing. A supervised approach combining embedding-based features with ML classifiers distinguishes ChatGPT- vs human-authored Python code with up to 98% accuracy in controlled settings; public dataset and trained models are released, and the authors note an evolving "arms-race" dynamic (Oedingen et al., 2024).

Automatic program generation survey. A broad overview maps ML techniques for code generation/completion and code knowledge mining, summarizing models (e.g., n-gram, RNN/CNN, hybrids/graphs) and application effects—useful context for framing test adequacy when evaluating generated code (Zhang & Jiang, 2020).

Evidence and scalability. Much of the comparison evidence is laboratory/benchmark-based and small-N or language-specific (Nikolaidis et al., 2024; Sagodi et al., 2024), and the review plugin paper reports preliminary validation rather than large-scale trials (Almeida et al., 2024). Provenance detection shows high accuracy in controlled corpora but must track model drift/evasion over time (Oedingen et al., 2024).

4.3.3 Quality attributes and reuse

A software reusability prediction system (FRF-ANN) over cohesion/coupling/complexity metrics reports improvements over baselines and explicitly tests the model using constraint-based testing, with reported metrics including Precision, Recall, Accuracy, F-measure, and error measures (RMSE/MAE) (Subha et al., 2023). The authors argue that better reusability decisions can reduce cost and—by concentrating reuse on higher-quality components—shrink redundant test design and focus oracles on higher-risk parts.

The evaluation is controlled with clear metrics; generalization to diverse architectures, CI environments, and live product code remains an open question to be established via multi-project replications (Subha et al., 2023).

4.3.4 System-level assurance, DevOps and terminology

Surveys and taxonomies. Broad SE4AI syntheses position validation concerns across the data–model–system stack and SWEBOK areas, emphasizing where testing is concentrated (e.g., dependability/safety), frequent data quality issues, and the role of monitoring and simulation in practice (Martinez-Fernandez et al., 2022). Complementary taxonomic work on quality assessment for intelligent software systems consolidates attributes/metrics and assessment approaches and argues for stronger tool support and remedial strategies—inputs to runtime monitoring and safety arguments (Jabborov et al., 2023). A complementary overview for ML in big-data systems charts QA

growth, pipeline-stage techniques, and the need to operationalize responsible qualities (fairness, privacy, explainability) in concrete testing patterns (Ogrizovic et al., 2024).

Quality-aware DevOps alignment. A DevOps survey maps quality gates and monitoring to CI/CD, IaC, testing/verification, and runtime management—clarifying where automated tests and production checks belong and highlighting gaps (e.g., DevOps for AI software) (Alnafessah et al., 2021).

Terminology and shared views. A common testing terminology for SE and data-science experts proposes a mapping between classical software testing and AI testing concepts to reduce misalignment and coverage gaps in ML projects (Joeckel et al., 2021).

Process frameworks that operationalize assurance. MLTRL adapts technology readiness levels to ML with stage gates tailored to data/model risks, offering practical anchors for "what to test when" across a lifecycle (Lavin et al., 2022). Responsible-AI patterns are operationalised from 21 practitioner interviews—turning ethics principles (e.g., fairness, robustness, transparency) into concrete patterns that can be realised via tests/monitors and engineering practices (Lu et al., 2022). A fairness-engineering study with 117 practitioners provides a replication package and reports that fairness remains a second-class quality compared to accuracy/security—actionable inputs to bias tests and dataset audits in CI (Ferrara et al., 2024). Broader SE experience papers underline socio-technical challenges (data/process quality, human-in-the-loop) and the need to go beyond purely functional testing when engineering AI systems (Fischer et al., 2021). An SE perspective SLR highlights limited tool maturity and immature, non-reproducible evidence in parts of the literature, inviting stronger empirical foundations for QA adoption (Giray, 2021).

These works are strong on what evidence to seek (frameworks, taxonomies, patterns) but lighter on multi-site outcome studies; more public artefacts/benchmarks and longitudinal evaluations are needed (Lavin et al., 2022; Lu et al., 2022; Martinez-Fernandez et al., 2022; Alnafessah et al., 2021).

4.3.5 Scaled-Agile assistants

A focused SLR on AI-driven assistants in scaled-Agile (SAFe) catalogs benefits and challenges and lists assistants across planning/control and development phases; the discussion notes a preponderance of advantages but explicitly calls for caution and more empirical evaluation—consistent with the need for QA guardrails around assistant outputs (Saklamaeva & Pavlic, 2024).

4.3.6 Overall evaluation

T3 contributions primarily provide scaffolding for testability—frameworks, patterns, lifecycle gates, and text-analytic aides that turn abstract qualities (fairness, explicability, robustness) and early artefacts (requirements/design text, smells) into checkable claims and actionable engineering hooks. Evidence is strongest for conceptual/process guidance and targeted laboratory studies (e.g., ambiguity detection, refactoring assistance, provenance detection). Direct, multi-site demonstrations of testing outcomes at scale (e.g., reduced flakiness, improved APFD/APFDc, lowered triage time across releases) are still scarce.

Strengths. T3 supports testability by construction: HCAI/ethics frameworks, ontologies, and MLTRL-style stage gates translate non-functional principles into verifiable acceptance criteria, traceability links, and assurance activities. It also offers shift-left enablers, as RE/NLP pipelines (classification, traceability, ambiguity detection) and LLM-assisted refactoring/review move defect prevention and evidence capture before execution testing. Governance and terminology contributions (DevOps quality-gate mappings, responsible-AI patterns, shared SE–DS testing terminology) reduce process gaps and misalignment across roles. Operational hooks such as provenance detection and quality-aware DevOps provide practical insertion points for risk-based testing, supply-chain checks, and runtime monitoring. Finally, a transparency trend is visible, as several works release artefacts (code/data/replication packages) or explicit checklists that support adoption and auditability.

Weaknesses. An outcome evidence gap persists: many papers stop at feasibility/frameworks, and few quantify downstream testing KPIs (e.g., defect leakage, APFD/APFDc, reviewer time saved, flaky rate) in multi-project settings. External validity and drift are also under-addressed, since laboratory/benchmark studies are often small-N or language/domain-specific, and adaptation under evolving models/data (drift, evasion) is rarely reported. Fragmented metrics and definitions remain a barrier, as inconsistent terminology and heterogeneous measurements

hinder comparability and meta-analysis. Operational under-specification is common, with limited reporting on CI/CD integration, budget/compute, and human-in-the-loop workload; governance is frequently described at policy level without run-book detail. Reproducibility gaps also appear, as not all studies provide runnable artefacts, seeds, or threats-to-validity discussions, and some LLM/code studies exhibit benchmark or dataset bias.

Scalability levers (laboratory–programme-level use). Qualities should be made testable early by encoding fairness/explicability/robustness as a priori acceptance criteria and tracing them into tests and monitors, supported by an evidence register that binds requirements, tests, and runtime checks. Gates and KPIs should be standardised by defining minimal quality gates (data checks, model evaluation, static/code review, deployment guardrails, runtime SLO/SLA) and reporting common KPIs (defect leakage, flaky rate, mean review time, APFD/APFDc, MTTR). Operationalisation for CI/CD should include budget-aware variants (time/CPU caps, caching, warm-starts), deterministic seeds, and repeat-with-CIs protocols, together with runbooks for rollout, rollback, and kill-switches. Governance should be paired with artefacts by linking policies (responsible-AI patterns, regulatory duties) to concrete templates such as checklists, risk logs, data-provenance/SBOM manifests, and provenance detectors in the pipeline. Human-in-the-loop should be treated by design by surfacing explanations and rationales, measuring human effort saved, and keeping reversible actions and reviewer controls. Replications and benchmarks should be prioritised through cross-project and longitudinal evaluations, releasing datasets, prompts/configurations, and containers, and aligning on shared benchmarks for RE–NLP, provenance, and refactoring tasks. Drift and adversarial readiness should be strengthened through drift monitors, periodic re-evaluation, and adversarial/evasion tests for provenance and fairness tooling, with documented re-tuning cadence and ownership. Finally, change management and skills need explicit support through training playbooks for RE/QA/dev roles, mapped responsibilities to gates, and check-in activities such as quality reviews at stage gates.

How this section relates to software testing? These contributions relate to software testing in four main ways. First, they support defining testable claims: HCAI (Human-Centered AI) and ethical-requirement artefacts translate abstract non-functional requirements (e.g., fairness, explicability, autonomy) into measurable acceptance criteria and test oracles (Ahmad et al., 2023a, 2023b; Guizzardi et al., 2023). Second, they guide where to place checks: DevOps/SE4AI mappings indicate suitable insertion points for quality gates, including data validation, model evaluation, code/static analysis, deployment checks, and runtime monitoring/alerting (Martinez-Fernandez et al., 2022; Alnafessah et al., 2021). Third, they enable shift-left practices: LLM-assisted refactoring and review, RE-text classification, and ambiguity detection reduce defect injection before execution testing, while provenance detectors inform risk-based selection of security and regression tests (Baumgartner et al., 2024; Almeida et al., 2024; Silva-Rodriguez et al., 2020; Izhar et al., 2025; Oedingen et al., 2024). Finally, they inform regression scope and adequacy: prioritization trade-offs (fuzzy/optimization/ML) and reusability prediction influence which tests to run and where to focus oracle effort, and program-generation surveys highlight evaluation pitfalls that should be reflected in test adequacy criteria (Anwar & Bashir, 2023; Subha et al., 2023; Zhang & Jiang, 2020).

5 DISCUSSION

The reviewed literature reveals numerous contributions of AI to software testing. AI techniques have been used to automate test case generation, selection, execution, and evaluation (Giray, 2021). Machine learning approaches improve defect detection, debugging support, and the identification of performance issues (Lavin et al., 2022). Large Language Models (LLMs), such as ChatGPT and Copilot, increasingly influence testing indirectly via code synthesis that still requires validation (Sagodi et al., 2024). Taken together, earlier research highlights improvements in efficiency, scalability, and adaptability of testing practice.

Key advantages recur in higher test automation, the ability to explore large test spaces, and faster execution and evaluation (Saklamaeva & Pavlic, 2024; Giray, 2021). AI-driven assistants also support collaboration in scaled Agile settings, streamlining testing workflows and reducing manual effort (Saklamaeva & Pavlic, 2024). Nonetheless, challenges persist. Risks include over-reliance on tools, inaccuracies in AI-generated test assets, and difficulty interpreting model outputs (Ahmad et al., 2023a; Anwar & Bashir, 2023). Current AI models struggle with nuanced or context-specific requirements, potentially leaving coverage gaps. Ethical and bias considerations continue to raise fairness and trustworthiness concerns (Guizzardi et al., 2023).

Despite significant advances, several gaps and opportunities for further research remain across the reviewed articles. First, further work is needed on swarm intelligence algorithms to improve test coverage efficiency (Khatibsyarbini et al., 2019). In addition, the literature calls for more sophisticated models that can address complex real-world testing scenarios, such as penetration testing in the automotive sector (Garrad & Unnikrishnan, 2023). Research also highlights the need to deepen work on test suite optimization, with greater attention to machine learning approaches and clustering methods (Kiran et al., 2019), as well as to expand research on defect prediction and how machine learning can support automated software testing (Amalfitano et al., 2024; Pandit et al., 2022).

More broadly, several studies emphasize the need to integrate AI more comprehensively across software testing activities, including unit, integration, conformance, security, usability, reliability, acceptance, combinatorial, and equivalence partitioning testing (Amalfitano et al., 2024), and to expand AI applications across software engineering phases such as design and testing (Sofian et al., 2022). Further investigations are also required into vulnerabilities in AI-based software, including TensorFlow applications and AI-based gaming systems (Filus & Domanska, 2023; Liaqat et al., 2020), alongside the development of new methods for predicting the quality of AI-based software using metrics and data from real-world AI projects (Jabborov et al., 2023).

Finally, multiple works underline persistent practical challenges in testing AI systems. Collecting sufficient and representative data remains difficult and motivates tools that enable data collection without introducing distortions (Martinez-Fernandez et al., 2022). Scalable testing is also challenging for AI and machine learning applications involving millions of parameters (Martinez-Fernandez et al., 2022). In this context, automated solutions are still needed to support AI system testing, including the generation of reliable test oracles and effective test cases (Martinez-Fernandez et al., 2022).

5.1 What the corpus says

Our coded evidence shows a strong tilt toward laboratory or secondary evidence and comparatively few industrial deployments: 43.06% laboratory/simulation versus 4.17% industrial/practitioner, 30.56% secondary evidence, 8.33% conceptual/guideline and 13.89% hybrid (laboratory + industrial/survey or laboratory + data from industry) studies. Replicability is likewise constrained: only 25% of papers provide public artefacts, while 75% either lack or do not specify them. Measurement is uneven: 45.83% of papers report no empirical metrics, with the remainder split across AI-performance metrics (22.22%), testing effectiveness (13.89%), method-specific/quality for AI metrics (11.11%) and user/survey/interview statistics (6.94%) measures.

Three quantitative signals from our dataset recur across themes T1–T3 and explain much of the field's uneven progress:

- **Industrial realism is thin**. Of 72 papers, only 3 are clearly industrial and 10 hybrid; 31 are laboratory/simulation based, 6 are conceptual/guideline and 22 are secondary. This imbalance helps explain why many successful experiments do not generalize beyond benchmarks or controlled settings.
- **Replicability is weak**. Only 18/72 papers provide public artefacts (code/data); 54 offer none or it is unspecified. Replicability correlates with stronger, more actionable claims: where artefacts exist, results tend to include clearer task definitions and evaluation setups that others can reuse.
- **Measurement is uneven**. 33 papers report no empirical metrics; where metrics appear, they split between testing effectiveness/efficiency (e.g., APFD/time; 10 papers) and AI performance metrics (e.g., accuracy/F1; 16 papers), with a smaller block using method-specific quality metrics (8) and user/survey/interview statistics (5). This heterogeneity hampers cross-study comparisons and encourages optimism based on incomparable endpoints.

By theme, T1 (AI-for-testing) dominates (32 papers), T3 (AI-related SE with testing implications) is substantial (25), and T2 (testing/assuring AI systems) is smaller (15). This distribution mirrors practice: more energy goes into applying AI to testing tasks than into assuring AI systems themselves, while process/ethics/governance work (T3) grows but rarely connects to executable checks.

5.2 An Evidence–Artefact–Outcome model

To complement the thematic synthesis, we instantiate the **Evidence–Artefact–Outcome (EAO) model** on our corpus, focusing on evidence and artefacts (D7, D9), to assess research maturity and reproducibility signals. In this operationalization, **evidence** is captured through the study's setting (industrial, hybrid, laboratory-simulation,

secondary, or conceptual), while **artefacts** are coded as either publicly available or none/unspecified. We additionally examine whether studies report measurable outcomes, operationalized here as the presence of empirical metrics.

Artefacts and measurable outcomes. Public artefact availability co-occurs strongly with empirical measurement. Among papers that provide public artefacts, 18/72 studies (25.0%), 16/18 (88.9%) report empirical metrics, whereas 2/18 (11.1%) do not. In contrast, among papers with no or unspecified artefacts (54/72, 75.0%), only 23/54 (42.6%) report metrics and 31/54 (57.4%) do not. This pattern suggests that openness is associated with more measurable and thus more verifiable claims (88.9% vs. 42.6%).

Practice-proximal evidence remains scarce and is rarely open. Only 13/72 studies (18.1%) report practice-proximal evidence, combining 3 industrial studies (4.2%) and 10 hybrid studies (13.9%). Within these 13 studies, public artefacts are available in 4/13 cases (30.8%) (Ferrara 2024; Giuliano 2021; Liu 2025; Wang 2025); notably, none of the industrial studies (0/3) provide public artefacts. Empirical metrics are nevertheless relatively common within this practice-proximal subset (10/13, 76.9%). Overall, the most compelling pathway—practice-proximal evaluation paired with openness and measurable outcomes—exists but remains rare.

Reproducibility varies by claim type. Studies that claim a model/technique outperforms baselines (D10) include 16 papers, of which 8/16 (50.0%) provide public artefacts (e.g., Kusharki 2022; Oedingen 2024; Vinayakumar 2019; Giuliano 2021; Liu 2025; Nikolaidis 2024; Tosi 2024; Wang 2025). For testing-methodology effectiveness claims (7 papers), 4/7 (57.1%) provide artefacts (Khatibsyarbini 2019; Liaqat 2020; Ramirez 2023; Zhu 2020). Governance/security-oriented contributions (8 papers) exhibit lower artefact availability, with only 2/8 (25.0%) providing artefacts (Filus & Domanska 2023; Ferrara 2024). These differences indicate that reproducibility signals are strongest where head-to-head methods are evaluated and weakest for governance-style contributions.

LLM/GenAI studies are relatively open but still laboratory heavy. The LLM/GenAI subset comprises 10 papers (13.9% of the corpus). Half provide public artefacts (5/10, 50%) (e.g., Tosi 2024; Nikolaidis 2024; Almeida 2024; Baumgartner 2024; Wang 2025), and 6/10 (60%) report empirical metrics (Sagodi 2024; Baumgartner 2024; Nikolaidis 2024; Rahman 2024; Tosi 2024; Wang 2025). However, only 1/10 (10%) meets the strongest credibility combination of being simultaneously open, measured, and practice-proximal (industrial/hybrid), namely Wang (2025). Thus, while LLM-focused studies appear more inclined to share artefacts than legacy ML studies, credible practice-proximal evaluations remain uncommon.

A strict "all-three" credibility filter selects very few studies. Defining "all-three credible" as industrial/hybrid evidence, public artefacts, and empirical metrics yields only 4/72 studies (5.6%) across the corpus (Ferrara 2024; Giuliano 2021; Liu 2025; Wang 2025). Within the LLM subset, only 1/10 (10%) satisfies the same filter (Wang 2025). This reinforces the need for more practice-proximal studies with reusable artefacts and measurable outcomes to support replication and technology transfer.

5.3 Optimism, bias, and maturity

Optimism bias and survivorship. With 33 metric-free studies and 54 without artefacts, positive narrative claims can outpace measurable progress. Publication norms that favor novelty over replication magnify this bias.

Comparator fragility. Where "outperforms baseline" appears (16 papers), baselines, datasets, and metrics vary widely. Without shared suites, performance deltas risk being artefacts of setup rather than method superiority.

LLM volatility. Rapid model updates and dataset drift mean results may become stale quickly. Regular re-evaluation and prompt/version disclosure should be treated as first-class threats-to-validity items, not afterthoughts.

5.4 Implications for researchers and practitioners

Prioritize hybrid/industrial validations. A recurring limitation in the corpus is the dominance of laboratory-based evaluations. Future studies should therefore prioritize hybrid and industrial validations by transitioning promising techniques from controlled settings into practice-proximal pilots (e.g., CI/CD trial runs, shadow deployments, or staged rollouts). In addition to effectiveness metrics, evaluations should report engineering constraints that shape real-world adoption, such as latency, flakiness, and operational cost.

Release artefacts by default. To improve reproducibility and enable cumulative research, techniques should be accompanied by publicly available artefacts whenever feasible. At minimum, this includes datasets (or data-generation procedures), prompts (for LLM-based approaches), test oracles, and execution pipelines (e.g., scripts and container images). Making artefacts available systematically strengthens the evidence–artefact–outcome (EAO) credibility profile of studies and supports reuse by both researchers and practitioners.

Standardize task–metric bundles. Comparability across studies is currently hindered by heterogeneous metrics and bespoke benchmarks. The field would benefit from curated task–metric bundles aligned with common testing activities. For example, within T1, test-case prioritization can be reported using APFD together with execution time and faults detected on predefined subject systems; oracle generation can be evaluated with precision/recall and human-acceptance rates; and GUI exploration can be reported using coverage and the number of unique states reached. Within T2, robustness and fairness evaluations can be strengthened through perturbation or stress-test suites, bias-delta reporting, and the measured effectiveness of safety channels or mitigation mechanisms.

Bring governance into CI. Governance-oriented practices (T3) should be operationalized as engineering mechanisms rather than treated as after-the-fact audits. One practical direction is to translate policy and ethics requirements into executable checks and monitors within CI/CD, including drift and fairness tests, explanation logging, incident response runbooks, and auditable trails that support traceability and accountability.

Operationalize LLM validity. For LLM/GenAI-based contributions, stronger validity reporting is needed. Studies should disclose the model and version, prompt templates, reference sets, and contamination controls, and clearly specify human-in-the-loop criteria where applicable. Importantly, evaluations should report cost and latency alongside accuracy-oriented outcomes to reflect the constraints under which such approaches are deployed.

5.5 Where the field should head next

First, the field needs shared testbeds and meta-benchmarks. Community suites for both T1 and T2 should provide fixed artefacts, seeded faults, and adversarial evaluation scenarios so that approaches can be compared under consistent conditions rather than on bespoke datasets and pipelines.

Second, researchers should bridge T1–T2–T3 by making responsible-AI requirements testable. One practical direction is to encode taxonomy terms and governance requirements as acceptance criteria and runtime monitors and then quantify their impact on defect detection and incident rates.

Third, stronger minimum reporting standards are needed. At a minimum, studies should disclose datasets, artefact links, baselines, external-validity constraints, and, for LLM-based work, model provenance and contamination controls.

6 CONCLUSIONS, LIMITATIONS AND FUTURE WORK

This review provides a testing-centric, two-axis synthesis unifying AI-for-testing (T1) with the testing of AI systems (T2) and connecting them to software engineering adjacent practices (T3). Unlike prior overviews, it normalizes evidence via D1–D10 and makes evaluation metrics (D6), maturity (D7), artefacts (D9), and LLM-specific validity risks first-class. The quantitative patterns are unambiguous: laboratory-heavy evidence, limited open artefacts, and mixed measurement constrain generalizable claims of industrial impact. The field is advancing, but responsible adoption depends on replicability and practice-proximal validation.

6.1 What this review adds

Adoption hinges on the evidence–artefacts–outcomes (EAO). Across 72 papers, credible paths to practice appear when Evidence is practice-proximal (industrial/hybrid), Artefacts are open and reusable, and Outcomes are measured with task-appropriate metrics. Only 4/72 papers meet all three, which explains why many promising techniques do not translate beyond benchmarks.

The field's center of gravity is T1, but maturity is uneven. T1 (AI-for-testing) dominates in activity, T2 (testing/assuring AI systems) is smaller but crucial for safety/robustness, and T3 (process/governance) offers requirements and policy assets that rarely become executable checks. Progress requires bridging T1–T2–T3 inside CI/CD rather than treating them as separate literatures.

Openness and metrics co-vary with credibility. Papers that release artefacts are far more likely to report measurable outcomes (88.9% vs 42.6%). Conversely, the prevalence of no-metrics (45.8%) and no/unspecified artefacts (75.0%) inflates optimism while limiting replication and comparison.

These insights are grounded in the coded patterns (D-items), not generic claims: thin industrial realism (D7), sparse artefacts (D9), and heterogeneous measurement (D6) are the binding constraints.

6.2 Concrete implications

For researchers, we recommend designing studies that satisfy at least two – and ideally all three – EAO elements: practice-relevant evidence, publicly available artefacts, and measurable outcomes. In addition to accuracy-oriented metrics, studies should report engineering constraints (e.g., cost, latency, and flakiness), clearly specify baselines, and document contamination controls when LLMs are involved. Where feasible, releasing runnable artefacts (e.g., containers or pipelines) can further enable reuse and replication.

For practitioners, we recommend prioritizing techniques supported by public artefacts, clearly defined task–metric bundles (e.g., APFD/time for prioritization, precision/recall for oracle generation, and coverage/states for GUI exploration), and prior industrial or hybrid evidence. Governance checks (e.g., fairness, drift monitoring, and explanation logging) should be integrated into CI pipelines as automated gates rather than performed as after-the-fact audits.

6.3 Prioritized, gap-aligned roadmap

Grounded in the gaps observed in the corpus (D7–D9), we propose the following agenda:

1. **Practice-proximal pilots with open artefacts.** Promising T1/T2 approaches should be evaluated in practice-proximal settings and accompanied by open artefacts to support reuse. Where feasible, authors should release data/prompts/oracles and containerized pipelines, and report both effectiveness and operational metrics (e.g., APFD and time for prioritization, defect escape, and incident rates).
2. **Shared testbeds and meta-benchmarks.** The community would benefit from shared, task-specific testbeds for TCP, GUI exploration, metamorphic robustness, and governance-oriented checks (e.g., fairness and drift). Such suites should include seeded faults and red-team scenarios to enable consistent head-to-head comparisons and reduce fragmentation across evaluation setups.
3. **Standard task–metric bundles.** To address metric heterogeneity, future work should curate reference task–metric bundles (e.g., for prioritization, oracle generation, API validity, and mutation testing). Using comparable bundles would improve interpretability and make results more comparable across methods and datasets.
4. **Threats and provenance minimums for LLM-based studies.** LLM-focused studies should adopt minimum reporting standards, including model/version disclosure, prompt templates, contamination guards, and operating constraints such as cost and latency. Because model churn can alter performance and behavior, it should be treated as a first-class external-validity risk.
5. **Governance-as-tests.** Responsible-AI requirements (e.g., fairness, privacy, transparency) should be operationalized as executable CI checks and runtime monitors. Studies should then evaluate their impact longitudinally (across multiple releases), reporting effects on defect escape and real-world incidents.

6.4 Limitations

Our scope is WoS-only, English-only, and Open-Access filtered, which may omit relevant studies and under-capture fast GenAI iterations. These are acknowledged methodological constraints of this synthesis.

ADDITIONAL INFORMATION AND DECLARATIONS

Conflict of Interests: The authors declare no conflict of interest.

Author Contributions: C.-V.L.: Conceptualization, Methodology, Data curation, Investigation, Visualization, Writing – original draft, Writing – review and editing; C.-V.K.: Conceptualization, Methodology, Data curation, Investigation, Visualization, Writing – original draft, Writing – review and editing, Supervision.

Statement on the Use of Artificial Intelligence Tools: The authors declare that they didn't use artificial intelligence tools for text or other media generation in this article.

APPENDIX A – COMPARISON OF RELATED WORK VERSUS THIS ARTICLE

Group	Paper	Scope focus	Method & evidence basis	Coverage of testing activities	Industrial realism	Rigor & threats reporting	GenAI / LLM coverage	Distinct limitations / gaps	Key limitation	How this work differs
AI in Software Testing	Amalfitano et al. (2024) Tertiary study (ACM CSUR)	AI techniques applied to testing tasks (generation, prioritization, oracle design, optimization)	Tertiary mapping of 20 secondary studies; taxonomy-driven; no new primaries.	Broad and fine-grained across classic testing tasks.	Indirect only (inherits from secondaries); no fresh industrial cases.	Systematic mapping described; maturity/replicability not deeply audited.	Surveys NLP/text mining within AI-for-testing, but it doesn't run an explicit, LLM/GenAI analysis	Strong breadth but limited appraisal of evidence maturity and reproducibility.	Tertiary level—limited granularity on primary evidence.	Adds per-study maturity (D7), replicability (D9), and explicit LLM role; unifies with testing-of-AI (T2).
	Boukhlif et al. (2023) Bibliometric (Electronics)	Trends and collaboration in intelligent software testing (2012 - 2022).	WoS bibliometrics (Bibliometrix/VOS viewer); networks & counts.	Topic-level themes (generation, prioritization, defect prediction) but not activity-level evaluation.	None (publication trends; no deployments).	Notes DB/time/language limits; not testing-method threats.	Pre-LLM framing; ML/DL trend focus.	No technique effectiveness or dataset openness; practice transfer unclear.	Bibliometric only; no study-level QA; WoS-only bias.	Goes beyond trends to compare effectiveness, artefacts, and evidence settings across T1 - T3.
	Kiran et al. (2019) SLR (IEEE Access)	Test-suite optimization (selection/reduction/prioritization).	SLR of 58 studies (2016 - 2019); catalogs tools/metaheuristics & metrics.	Narrow but deep in regression/optimization.	Mostly laboratory benchmarks; limited explicit industry validation.	Protocol + acknowledged selection/search biases; inconsistent reporting in primaries	None (pre-LLM).	Pre-dates DL/LLM prioritization; limited generalizability evidence, calls for multi-obj. approaches	Pre-LLM era; heterogenous datasets & metrics hamper comparability	Connects optimization with newer AI/LLM methods; contrasts laboratory versus industrial maturity across tasks.
	Abo-eleneen et al. (2023) SLR (IST)	Reinforcement learning in testing (exploratory/GUI, test gen, navigation)	SLR of 40 RL-in-testing studies; analyzes rewards/environments.	Focused: GUI/exploration, test generation, environment control.	Mostly simulated/laboratory settings; few real deployments.	Explicit threats section; reproducibility concerns noted.	No (agent-centric RL focus).	Fragmented setups; lack of scalable, multi-agent, real-world studies.	Narrow technique scope (RL); limited industrial validation.	Positions RL alongside other AI (LLMs, SBSE) and rates maturity/replicability consistently.
AI across the SE lifecycle	Alenezi & Akour (2025) Review (Applied Sciences)	AI/LLMs across SDLC testing covered as capability.	Narrative review + 6 industrial case summaries + practitioner survey.	High-level (test generation, prioritization) without deep categorization.	Some practice flavour via cases/survey; not systematic per testing task.	Governance/risks discussed; no formal threats-to-validity section.	Yes - copilots/LLMs discussed conceptually.	No separation of AI-for-testing vs testing-of-AI; taxonomy for testing is light.	Narrative breadth, limited method transparency vs SLRs.	Keeps testing as organizing axis; cleanly separates T1 vs T2; quantifies maturity.
	Durrani et al. (2024) SLR (IEEE Access)	AI integration in SE phases (2013 - 2023).	Systematic review - 110 papers analyzed; phase-wise mapping; breadth over depth.	Counted/mapped; limited activity-level scrutiny.	Not analyzed specifically for testing context.	Scope/coverage limitations discussed.	Mentions modern AI classes; limited LLM specificity.	Shallow on testing maturity and reproducibility.	Top-cited filter risks selection bias; modest depth on testing specifics.	Zooms in on testing with per-paper D7 - D9 maturity & artefact coding.
	Navaei & Tabrizi (2022) Review (ENASE)	ML use along SDLC; testing phase reported as most active.	Systematic review - 150 papers (2015 - 2021); the method is essentially a descriptive count, with figures and per-phase tallies	Highlights prevalence; lacks fine-grained taxonomy.	None (no separation of laboratory/industry).	Field-level limitations noted, not paper-level.	No.	Lightweight; no quality/replicability assessment.	Short proceedings length; high-level counts; limited rigor.	Adds structured activity taxonomy and per-paper rigour checks (D8/D9).
	Kokol (2024) Synthetic knowledge synthesis (Information)	AI in SE via bibliometric/thematic clusters.	Bibliometric synthesis; 15 categories / 5 themes.	Testing appears as a cluster (e.g., defect prediction) without operational detail.	None (trend-level only).	Bibliometric method limitations noted.	Minimal, trend-level.	Cannot judge effectiveness or practice maturity.	Scopus-only corpus; many preprints (incl. LLM work) excluded.	Translates clusters into actionable testing activities with maturity and artefacts lenses.
Testing/Validation of AI systems	Myllyaho et al. (2021) SLR (JSS)	Validation/V&V strategies for AI systems (trial, simulation, expert, redundancy).	SLR of 90 studies; taxonomy of validation methods; characterization by domain, task, system complexity, and malfunction impact.	Targets AI-system assurance, not classic ST activities.	Discusses realism qualitatively; advocates continuous validation.	Yes - bias/terminology/selection considered.	No (pre-LLM surge).	Not aligned with traditional test workflows; limited link to ST artefacts.	Strong taxonomy; less on SW testing metrics; pre-LLM.	Bridges assurance of AI systems (T2) with classic ST tasks (T1) in one frame.

Group	Paper	Scope focus	Method & evidence basis	Coverage of testing activities	Industrial realism	Rigor & threats reporting	GenAI / LLM coverage	Distinct limitations / gaps	Key limitation	How this work differs
	Martinez - Fernandez 2022 (TOSEM)	SE for AI-based systems (incl. quality/testing/maintenance).	SMS with a hybrid search (Scopus + backward/forward snowballing) that yielded 248 studies (2010–Mar 2020).	Treats testing/quality as key SE4AI areas; data/maintenance challenges.	Notes conceptual/laboratory prevalence; maintenance under-studied.	Yes - scope and selection limitations explicit; assessed realism/scale and whether primaries reported threats to validity	No (pre-mainstream LLM era).	Breadth without side-by-side comparison to AI-for-testing; predates LLMs.	Heterogeneous terminology; urges standard taxonomies, broader sources incl. arXiv.	Provides unified T1/T2/T3 lens with updated LLM coverage and maturity scoring.
-	This work (present review)	AI in software testing (T1) + validation of AI systems (T2) + AI-related SE topics with testing implications (T3)	PRISMA-style SLR on WoS (2015–2025), documented protocol, unified D1–D10 extraction; 72 records after screening	Activity-level mapping (e.g., generation/prioritization, defect analytics, oracles, RE - test links), reported by themes T1–T3	Quantified evidence setting (31 laboratory, 3 industrial, 10 hybrid, 22 secondary); calls out thin practice validation.	Tracks threats-to-validity D8 (61 yes) and audits artefacts D9 (18 public, 54 none/unspecified) for replicability signals.	Explicit, task-level analysis of LLM roles (copilot/oracle/generator), benchmarks, and validity risks	Heterogeneous metrics/datasets; limited industrial deployments; sparse open artefacts; pre-LLM emphasis in prior syntheses.	WoS-only, English-only, Open-Access filter; terminology sensitivity; snapshot risk in fast-moving GenAI.	Integrates both directions (AI-for-testing & testing-of-AI) with maturity + replicability metrics; updates to LLM era.

APPENDIX B – PAPERS CLASSIFICATION IN DATA ITEMS AND CATEGORIES

Data item & categories		Count	% of corpus	References
D1 - study type according to WoS	Article	46	63.89	Abo-eleneen et al., 2023; Ahmad et al., 2023a; Ahmad et al., 2023b; Ali et al., 2023; Alnafessah et al., 2021; Amalfitano et al., 2024; Deng et al., 2020; Ferrara et al., 2024; Filus & Domanska, 2023; Garrad & Unnikrishnan, 2023; Giray, 2021; Guizzardi et al., 2023; Khatibsyarbini et al., 2019; Kiran et al., 2019; Kusharki et al., 2022; Layman & Vetter, 2024; Lavin et al., 2022; Lee & Seo, 2020; Liaqat et al., 2020; Martinez-Fernandez et al., 2022; Oedingen et al., 2024; Pandit et al., 2022; Rajapaksha et al., 2022; Rodriguez et al., 2016; Sagodi et al., 2024; Saklamaeva & Pavlic, 2024; Silva-Rodriguez et al., 2020; Sofian et al., 2022; Subha et al., 2023; Steidl et al., 2023; Tahvili et al., 2020; Tao et al., 2019; Vinayakumar et al., 2019; Xie et al., 2020; Zhang & Jiang, 2020; Zhu & Bayley, 2022; Alenezi & Akour, 2025; Almeida et al., 2024; Baumgartner et al., 2024; Cheng et al., 2023; Izhar et al., 2025; Liu et al., 2025; Mustaqeem et al., 2025; Ogrizovic et al., 2024; Tosi, 2024; Wang et al., 2025
	Proceedings paper	18	25.00	Augusto et al., 2019; Borg, 2021; Nguyen & Maag, 2020; Felderer & Ramler, 2021; Fernandes et al., 2021; Giuliano et al., 2021; Joeckel et al., 2021; Lanus et al., 2021; Leotta et al., 2022; Lu et al., 2022; Navaei & Tabrizi, 2022; Nikolaidis et al., 2024; Olszewska, 2020; Poth et al., 2019; Rahman et al., 2024; Ramirez et al., 2023; Strandberg et al., 2021; Zhu et al., 2020
	Review	8	11.11	Anwar & Bashir, 2023; Boukhlif et al., 2023; Fischer et al., 2021; Jabborov et al., 2023; Kokol, 2024; Myllyaho et al., 2021; Necula et al., 2024; Durrani et al., 2024
D2 - AI technique	Deep Learning (DL)	22	30.56	Amalfitano et al., 2024; Boukhlif et al., 2023; Deng et al., 2020; Filus & Domanska, 2023; Fischer et al., 2021; Kokol, 2024; Kusharki et al., 2022; Lavin et al., 2022; Martinez-Fernandez et al., 2022; Necula et al., 2024; Oedingen et al., 2024; Sofian et al., 2022; Steidl et al., 2023; Tao et al., 2019; Vinayakumar et al., 2019; Zhang & Jiang, 2020; Alenezi & Akour, 2025; Durrani et al., 2024; Liu et al., 2025; Mustaqeem et al., 2025; Ogrizovic et al., 2024; Olszewska, 2020
	Machine Learning (ML)	37	51.39	Ahmad et al., 2023a;Ahmad et al., 2023b; Ali et al., 2023; Amalfitano et al., 2024; Anwar & Bashir, 2023; Boukhlif et al., 2023; Ferrara et al., 2024; Fischer et al., 2021; Giray, 2021; Jabborov et al., 2023; Kokol, 2024; Lavin et al., 2022; Lee & Seo, 2020; Martinez-Fernandez et al., 2022; Myllyaho et al., 2021; Necula et al., 2024; Oedingen et al., 2024; Pandit et al., 2022; Rajapaksha et al., 2022; Silva-Rodriguez et al., 2020;Sofian et al., 2022;Subha et al., 2023;Steidl et al., 2023;Tao et al., 2019;Xie et al., 2020;Zhang & Jiang, 2020;Zhu & Bayley,2022;Alenezi & Akour, 2025; Augusto et al., 2019; Cheng et al., 2023; Borg, 2021; Durrani et al., 2024; Izhar et al., 2025; Mustaqeem et al., 2025; Navaei & Tabrizi, 2022; Ogrizovic et al., 2024;Poth et al., 2019
	AI (other)	15	20.83	Alnafessah et al., 2021; Guizzardi et al., 2023; Liaqat et al., 2020; Rodriguez et al., 2016; Saklamaeva & Pavlic, 2024; Felderer & Ramler, 2021; Fernandes et al., 2021; Izhar et al., 2025; Joeckel et al., 2021; Lanus et al., 2021; Lu et al., 2022; Olszewska, 2020; Ramirez et al., 2023; Strandberg et al., 2021; Zhu et al., 2020
	LLM/GenAI	10	13.89	Layman & Vetter, 2024; Necula et al., 2024; Sagodi et al., 2024; Alenezi & Akour, 2025; Almeida et al., 2024; Baumgartner et al., 2024; Nikolaidis et al., 2024; Rahman et al., 2024; Tosi, 2024; Wang et al., 2025
	SBSE/Metaheuristics (GA/PSO/ACO/Firefly)	4	5.56	Amalfitano et al., 2024; Khatibsyarbini et al., 2019; Kiran et al., 2019; Liu et al., 2025
	NLP	9	12.50	Amalfitano et al., 2024; Lee & Seo, 2020; Necula et al., 2024; Sofian et al., 2022; Tahvili et al., 2020; Alenezi & Akour, 2025; Almeida et al., 2024; Durrani et al., 2024; Leotta et al., 2022
	DNN	2	2.78	Borg, 2021; Giuliano et al., 2021
	Reinforcement Learning (RL)	2	2.78	Abo-eleneen et al., 2023; Garrad & Unnikrishnan, 2023
	SVM	2	2.78	Nguyen & Maag, 2020; Navaei & Tabrizi, 2022

Data item & categories		Count	% of corpus	References
D3 - testing activity	Testing of AI systems	14	19.44	Filus & Domanska, 2023; Myllyaho et al., 2021; Steidl et al., 2023; Tao et al., 2019; Xie et al., 2020; Zhu & Bayley, 2022; Augusto et al., 2019; Cheng et al., 2023; Borg, 2021; Felderer & Ramler, 2021; Lanus et al., 2021; Ogrizovic et al., 2024; Olszewska, 2020; Zhu et al., 2020
	Test generation/optimization/prioritization/regression	19	26.39	Abo-eleneen et al., 2023; Amalfitano et al., 2024; Boukhlif et al., 2023; Garrad & Unnikrishnan, 2023; Khatibsyarbini et al., 2019; Kiran et al., 2019; Layman & Vetter, 2024; Tahvili et al., 2020; Alenezi & Akour, 2025; Durrani et al., 2024; Nguyen & Maag, 2020; Fernandes et al., 2021; Giuliano et al., 2021; Leotta et al., 2022; Poth et al., 2019; Rahman et al., 2024; Ramirez et al., 2023; Strandberg et al., 2021; Tosi, 2024
	Defect/fault prediction/detection & analytics	12	16.67	Ali et al., 2023; Deng et al., 2020; Kokol, 2024; Lee & Seo, 2020; Pandit et al., 2022; Rajapaksha et al., 2022; Sofian et al., 2022; Vinayakumar et al., 2019; Liu et al., 2025; Mustaqeem et al., 2025; Navaei & Tabrizi, 2022; Wang et al., 2025
	Oracles & metamorphic/data-morphic testing	2	2.78	Kusharki et al., 2022; Liaqat et al., 2020
	Requirements (adjacent to testing)	7	9.72	Ahmad et al., 2023a; Ahmad et al., 2023b; Anwar & Bashir, 2023; Guizzardi et al., 2023; Necula et al., 2024; Silva-Rodriguez et al., 2020; Izhar et al., 2025
	MLOps/DevOps QA	1	1.39	Alnafessah et al., 2021
	Other (e.g. quality topics but not software testing directly)	17	23.61	Ferrara et al., 2024; Fischer et al., 2021; Giray, 2021; Jabborov et al., 2023; Lavin et al., 2022; Martinez-Fernandez et al., 2022; Oedingen et al., 2024; Rodriguez et al., 2016; Sagodi et al., 2024; Saklamaeva & Pavlic, 2024; Subha et al., 2023; Zhang & Jiang, 2020; Almeida et al., 2024; Baumgartner et al., 2024; Joeckel et al., 2021; Lu et al., 2022; Nikolaidis et al., 2024
D4 - system under test/domain	General software/technique-driven	47	65.28	Abo-eleneen et al., 2023; Ali et al., 2023; Alnafessah et al., 2021; Amalfitano et al., 2024; Boukhlif et al., 2023; Deng et al., 2020; Garrad & Unnikrishnan, 2023; Jabborov et al., 2023; Khatibsyarbini et al., 2019; Kiran et al., 2019; Kokol, 2024; Kusharki et al., 2022; Layman & Vetter, 2024; Lee & Seo, 2020; Necula et al., 2024; Oedingen et al., 2024; Pandit et al., 2022; Rajapaksha et al., 2022; Rodriguez et al., 2016; Sagodi et al., 2024; Saklamaeva & Pavlic, 2024; Silva-Rodriguez et al., 2020; Sofian et al., 2022; Subha et al., 2023; Tahvili et al., 2020; Vinayakumar et al., 2019; Xie et al., 2020; Zhang & Jiang, 2020; Alenezi & Akour, 2025; Almeida et al., 2024; Baumgartner et al., 2024; Durrani et al., 2024; Nguyen & Maag, 2020; Fernandes et al., 2021; Giuliano et al., 2021; Izhar et al., 2025; Leotta et al., 2022; Liu et al., 2025; Mustaqeem et al., 2025; Navaei & Tabrizi, 2022; Nikolaidis et al., 2024; Poth et al., 2019; Rahman et al., 2024; Ramirez et al., 2023; Strandberg et al., 2021; Tosi, 2024; Wang et al., 2025
	AI-based software/system	25	34.72	Ahmad et al., 2023a; Ahmad et al., 2023b; Anwar & Bashir, 2023; Ferrara et al., 2024; Filus & Domanska, 2023; Fischer et al., 2021; Giray, 2021; Guizzardi et al., 2023; Lavin et al., 2022; Liaqat et al., 2020; Martinez-Fernandez et al., 2022; Myllyaho et al., 2021; Steidl et al., 2023; Tao et al., 2019; Zhu & Bayley, 2022; Augusto et al., 2019; Cheng et al., 2023; Borg, 2021; Felderer & Ramler, 2021; Joeckel et al., 2021; Lanus et al., 2021; Lu et al., 2022; Ogrizovic et al., 2024; Olszewska, 2020; Zhu et al., 2020
D5 - datasets/benchmarks and availability	Open/public datasets or code	17	23.61	Deng et al., 2020; Ferrara et al., 2024; Filus & Domanska, 2023; Khatibsyarbini et al., 2019; Kusharki et al., 2022; Lee & Seo, 2020; Liaqat et al., 2020; Oedingen et al., 2024; Almeida et al., 2024; Baumgartner et al., 2024; Giuliano et al., 2021; Liu et al., 2025; Nikolaidis et al., 2024; Ramirez et al., 2023; Tosi, 2024; Wang et al., 2025; Zhu et al., 2020
	Unclear/none specified	11	15.28	Ahmad et al., 2023a; Ahmad et al., 2023b; Garrad & Unnikrishnan, 2023; Rajapaksha et al., 2022; Sagodi et al., 2024; Silva-Rodriguez et al., 2020; Xie et al., 2020; Zhu & Bayley, 2022; Izhar et al., 2025; Leotta et al., 2022; Rahman et al., 2024
	Industrial/proprietary corpora	7	9.72	Subha et al., 2023; Tahvili et al., 2020; Augusto et al., 2019; Cheng et al., 2023; Nguyen & Maag, 2020; Lu et al., 2022; Poth et al., 2019
	Hybrid (part open/public datasets or code, part industrial/proprietary)	1	1.39	Vinayakumar et al., 2019
	Secondary/conceptual (reviews, mappings, guidelines, no runnable dataset by design)	36	50.00	Abo-eleneen et al., 2023; Ali et al., 2023; Alnafessah et al., 2021; Amalfitano et al., 2024; Anwar & Bashir, 2023; Boukhlif et al., 2023; Fischer et al., 2021; Giray, 2021; Guizzardi et al., 2023; Jabborov et al., 2023; Kiran et al., 2019; Kokol, 2024; Layman & Vetter, 2024; Lavin et al., 2022; Martinez-Fernandez et al., 2022; Myllyaho et al., 2021; Necula et al., 2024; Pandit et al., 2022; Rodriguez et al., 2016; Saklamaeva & Pavlic, 2024; Sofian et al., 2022; Steidl et al., 2023; Tao et al., 2019; Zhang & Jiang, 2020; Alenezi & Akour, 2025; Borg, 2021; Durrani et al., 2024; Felderer & Ramler, 2021; Fernandes et al., 2021; Joeckel et al., 2021; Lanus et al., 2021; Mustaqeem et al., 2025; Navaei & Tabrizi, 2022; Ogrizovic et al., 2024; Olszewska, 2020; Strandberg et al., 2021
D6 - evaluation metrics	No empirical metrics	33	45.83	Abo-eleneen et al., 2023; Alnafessah et al., 2021; Amalfitano et al., 2024; Anwar & Bashir, 2023; Boukhlif et al., 2023; Fischer et al., 2021; Giray, 2021; Jabborov et al., 2023; Kiran et al., 2019; Kokol, 2024; Layman & Vetter, 2024; Lavin et al., 2022; Martinez-Fernandez et al., 2022; Myllyaho et al., 2021; Necula et al., 2024; Rodriguez et al., 2016; Saklamaeva & Pavlic, 2024; Sofian et al., 2022; Steidl et al., 2023; Alenezi & Akour, 2025; Almeida et al., 2024; Cheng et al., 2023; Borg, 2021; Durrani et al., 2024; Felderer & Ramler, 2021; Fernandes et al., 2021; Joeckel et al., 2021; Lanus et al., 2021; Mustaqeem et al., 2025; Navaei & Tabrizi, 2022; Ogrizovic et al., 2024; Poth et al., 2019; Strandberg et al., 2021
	Testing effectiveness/efficiency	10	13.89	Khatibsyarbini et al.,2019; Liaqat et al.,2020; Tao et al.,2019; Xie et al.,2020; Zhu & Bayley,2022; Nguyen&Maag,2020; Leotta et al.,2022; Ramirez et al.,2023; Wang et al.,2025; Zhu et al., 2020
	AI performance metrics	16	22.22	Ali et al., 2023; Deng et al., 2020; Garrad & Unnikrishnan, 2023; Kusharki et al., 2022; Lee & Seo, 2020; Oedingen et al., 2024; Silva-Rodriguez et al., 2020; Subha et al., 2023; Tahvili et al., 2020; Vinayakumar et al., 2019; Zhang & Jiang, 2020; Augusto et al., 2019; Baumgartner et al., 2024; Giuliano et al., 2021; Izhar et al., 2025; Liu et al., 2025
	Method-specific/quality for AI metrics	8	11.11	Ahmad et al., 2023a; Filus & Domanska, 2023; Guizzardi et al., 2023; Sagodi et al., 2024; Nikolaidis et al., 2024; Olszewska, 2020; Rahman et al., 2024; Tosi, 2024
	User/survey/interview statistics	5	6.94	Ahmad et al., 2023b; Ferrara et al., 2024; Pandit et al., 2022; Rajapaksha et al., 2022; Lu et al., 2022
D7 - evidence setting	Laboratory/simulation	31	43.06	Ahmad et al., 2023a; Deng et al., 2020; Filus & Domanska, 2023; Fischer et al., 2021; Garrad & Unnikrishnan, 2023; Khatibsyarbini et al., 2019; Kusharki et al., 2022; Layman & Vetter, 2024; Lavin et al., 2022; Lee & Seo, 2020; Liaqat et al., 2020; Oedingen et al., 2024; Sagodi et al., 2024; Silva-Rodriguez et al., 2020; Subha et al., 2023; Vinayakumar et al., 2019; Xie et al., 2020; Zhu & Bayley, 2022; Almeida et al., 2024; Baumgartner et al., 2024; Cheng et al., 2023; Nguyen & Maag, 2020; Fernandes et al., 2021; Izhar et al., 2025; Leotta et al., 2022; Nikolaidis et al., 2024; Olszewska, 2020; Rahman et al., 2024; Ramirez et al., 2023; Tosi, 2024; Zhu et al., 2020

Data item & categories		Count	% of corpus	References
	Secondary evidence	22	30.56	Abo-eleneen et al., 2023; Ali et al., 2023; Alnafessah et al., 2021; Amalfitano et al., 2024; Anwar & Bashir, 2023; Boukhlif et al., 2023; Giray, 2021; Jabborov et al., 2023; Kiran et al., 2019; Kokol, 2024; Martinez-Fernandez et al., 2022; Myllyaho et al., 2021; Necula et al., 2024; Rodriguez et al., 2016; Saklamaeva & Pavlic, 2024; Sofian et al., 2022; Zhang & Jiang, 2020; Alenezi & Akour, 2025; Durrani et al., 2024; Mustaqeem et al., 2025; Navaei & Tabrizi, 2022; Ogrizovic et al., 2024
	Industrial/practitioner	3	4.17	Tahvili et al., 2020; Lu et al., 2022; Poth et al., 2019
	Conceptual/guideline	6	8.33	Guizzardi et al.,2023; Tao et al.,2019; Borg,2021; Felderer&Ramler,2021; Joeckel et al.,2021; Lanus et al., 2021
	Hybrid (laboratory + industrial/survey, or laboratory +data from industry)	10	13.89	Ahmad et al., 2023b; Ferrara et al.,2024; Pandit et al.,2022; Rajapaksha et al.,2022; Steidl et al.,2023; Augusto et al.,2019; Giuliano et al.,2021; Liu et al.,2025; Strandberg et al., 2021; Wang et al., 2025
D8 - threats to validity/limitations discussed	Yes	61	84.72	Abo-eleneen et al., 2023; Ahmad et al., 2023a; Ahmad et al., 2023b; Ali et al., 2023; Amalfitano et al., 2024; Anwar & Bashir, 2023; Boukhlif et al., 2023; Deng et al., 2020; Ferrara et al., 2024; Filus & Domanska, 2023; Fischer et al., 2021; Garrad & Unnikrishnan, 2023; Giray, 2021; Guizzardi et al., 2023; Jabborov et al., 2023; Khatibsyarbini et al., 2019; Kiran et al., 2019; Kokol, 2024; Lavin et al., 2022; Liaqat et al., 2020; Martinez-Fernandez et al., 2022; Myllyaho et al., 2021; Necula et al., 2024; Oedingen et al., 2024; Pandit et al., 2022; Rajapaksha et al., 2022; Rodriguez et al., 2016; Sagodi et al., 2024; Saklamaeva & Pavlic, 2024; Silva-Rodriguez et al., 2020; Sofian et al., 2022; Steidl et al., 2023; Tahvili et al., 2020; Vinayakumar et al., 2019; Xie et al., 2020; Zhu & Bayley, 2022; Alenezi & Akour, 2025; Almeida et al., 2024; Augusto et al., 2019; Baumgartner et al., 2024; Cheng et al., 2023; Durrani et al., 2024; Fernandes et al., 2021; Giuliano et al., 2021; Izhar et al., 2025; Joeckel et al., 2021; Lanus et al., 2021; Leotta et al., 2022; Liu et al., 2025; Lu et al., 2022; Mustaqeem et al., 2025; Navaei & Tabrizi, 2022; Nikolaidis et al., 2024; Ogrizovic et al., 2024; Olszewska, 2020; Poth et al., 2019; Ramirez et al., 2023; Strandberg et al., 2021; Tosi, 2024; Wang et al., 2025; Zhu et al., 2020
	No / unclear	11	15.28	Alnafessah et al., 2021; Kusharki et al., 2022; Layman & Vetter, 2024; Lee & Seo, 2020; Subha et al., 2023; Tao et al., 2019; Zhang & Jiang, 2020; Borg, 2021; Nguyen & Maag, 2020; Felderer & Ramler, 2021; Rahman et al., 2024
D9 - artefacts (code/data availability)	No or not specified	54	75.00	Abo-eleneen et al., 2023; Ahmad et al., 2023a; Ahmad et al., 2023b; Ali et al., 2023; Alnafessah et al., 2021; Amalfitano et al., 2024; Anwar & Bashir, 2023; Boukhlif et al., 2023; Deng et al., 2020; Fischer et al., 2021; Garrad & Unnikrishnan, 2023; Giray, 2021; Guizzardi et al., 2023; Jabborov et al., 2023; Kiran et al., 2019; Kokol, 2024; Layman & Vetter, 2024; Lavin et al., 2022; Lee & Seo, 2020; Martinez-Fernandez et al., 2022; Myllyaho et al., 2021; Necula et al., 2024; Pandit et al., 2022; Rajapaksha et al., 2022; Rodriguez et al., 2016; Sagodi et al., 2024; Saklamaeva & Pavlic, 2024; Silva-Rodriguez et al., 2020; Sofian et al., 2022; Subha et al., 2023; Steidl et al., 2023; Tahvili et al., 2020; Tao et al., 2019; Xie et al., 2020; Zhang & Jiang, 2020; Zhu & Bayley, 2022; Alenezi & Akour, 2025; Augusto et al., 2019; Borg, 2021; Durrani et al., 2024; Nguyen & Maag, 2020; Felderer & Ramler, 2021; Fernandes et al., 2021; Izhar et al., 2025; Joeckel et al., 2021; Lanus et al., 2021; Leotta et al., 2022; Lu et al., 2022; Mustaqeem et al., 2025; Navaei & Tabrizi, 2022; Ogrizovic et al., 2024; Poth et al., 2019; Rahman et al., 2024; Strandberg et al., 2021
	Public artefact	18	25.00	Ferrara et al., 2024; Filus & Domanska, 2023; Khatibsyarbini et al., 2019; Kusharki et al., 2022; Liaqat et al., 2020; Oedingen et al., 2024; Vinayakumar et al., 2019; Almeida et al., 2024; Baumgartner et al., 2024; Cheng et al., 2023; Giuliano et al., 2021; Liu et al., 2025; Nikolaidis et al., 2024; Olszewska, 2020; Ramirez et al., 2023; Tosi, 2024; Wang et al., 2025; Zhu et al., 2020
D10 - main claim	Evidence synthesis / mapping – SLR/tertiary/bibliometric landscapes, gaps, trends	21	29.17	Abo-eleneen et al., 2023; Ali et al., 2023; Alnafessah et al., 2021; Amalfitano et al., 2024; Anwar & Bashir, 2023; Boukhlif et al., 2023; Fischer et al., 2021; Giray, 2021; Kiran et al., 2019; Kokol, 2024; Martinez-Fernandez et al., 2022; Myllyaho et al., 2021; Necula et al., 2024; Rodriguez et al., 2016; Saklamaeva & Pavlic, 2024; Sofian et al., 2022; Zhang & Jiang, 2020; Alenezi & Akour, 2025; Durrani et al., 2024; Mustaqeem et al., 2025; Navaei & Tabrizi, 2022
	Model/technique outperforms baselines – ML/DL/SBSE methods claiming superior accuracy, F1, etc.	16	22.22	Deng et al., 2020; Garrad & Unnikrishnan, 2023; Kusharki et al., 2022; Lee & Seo, 2020; Oedingen et al., 2024; Sagodi et al., 2024; Silva-Rodriguez et al., 2020; Subha et al., 2023; Tahvili et al., 2020; Vinayakumar et al., 2019; Giuliano et al., 2021; Izhar et al., 2025; Liu et al., 2025; Nikolaidis et al., 2024; Tosi, 2024; Wang et al., 2025
	Testing methodology effectiveness – efficacy of testing strategies (metamorphic/datamorphic/TCP, etc.)	7	9.72	Khatibsyarbini et al., 2019; Liaqat et al., 2020; Tao et al., 2019; Xie et al., 2020; Zhu & Bayley, 2022; Ramirez et al., 2023; Zhu et al., 2020
	Framework / method proposal (conceptual) – new frameworks/processes with little/no empirical validation	6	8.33	Ahmad et al., 2023a; Pandit et al., 2022; Steidl et al., 2023; Borg, 2021; Fernandes et al., 2021; Lanus et al., 2021
	Tool feasibility / practical utility – prototypes/tools showing workflow gains, actionability, or usability	6	8.33	Rajapaksha et al., 2022; Almeida et al., 2024; Baumgartner et al., 2024; Cheng et al., 2023; Nguyen & Maag, 2020; Rahman et al., 2024
	Governance / ethics / responsibility / security – HCAI, fairness, TRL/governance, privacy, terminology, vulnerabilities	8	11.11	Ahmad et al., 2023b; Ferrara et al., 2024; Filus & Domanska, 2023; Lavin et al., 2022; Augusto et al., 2019; Lu et al., 2022; Ogrizovic et al., 2024; Strandberg et al., 2021
	Industrial process / cost insight – effort/cost/practice findings or practitioner implications	2	2.78	Leotta et al., 2022; Poth et al., 2019
	Perspective / position / outlook – vision pieces, future directions	2	2.78	Layman & Vetter, 2024; Felderer & Ramler, 2021
	Data/knowledge artefact contribution – taxonomies, ontologies, released replicable assets (as the claim)	4	5.56	Guizzardi et al.,2023; Jabborov et al., 2023; Joeckel et al., 2021; Olszewska, 2020
Theme (T1 - T3)	T1. AI-based software testing	32	44.44	Abo-eleneen et al., 2023; Ali et al., 2023; Amalfitano et al., 2024; Boukhlif et al., 2023; Deng et al., 2020; Garrad & Unnikrishnan, 2023; Khatibsyarbini et al., 2019; Kiran et al., 2019; Kokol, 2024; Kusharki et al., 2022; Layman & Vetter, 2024; Lee & Seo, 2020; Pandit et al., 2022; Rajapaksha et al., 2022; Sofian et al., 2022; Tahvili et al., 2020; Vinayakumar et al., 2019; Alenezi & Akour, 2025; Durrani et al., 2024; Nguyen & Maag, 2020; Fernandes et al., 2021; Giuliano et al.,2021; Leotta et al., 2022; Liu et al., 2025; Mustaqeem et al., 2025; Navaei & Tabrizi, 2022; Poth et al., 2019; Rahman et al., 2024; Ramirez et al., 2023; Strandberg et al.,2021; Tosi, 2024; Wang et al.,2025
	T2. Testing/validation of AI systems	15	20.83	Filus&Domanska,2023; Liaqat et al.,2020; Myllyaho et al.,2021; Steidl et al.,2023; Tao et al., 2019; Xie et al.,2020; Zhu & Bayley,2022; Augusto et al.,2019; Cheng et al.,2023; Borg,2021; Felderer&Ramler,2021; Lanus et al.,2021; Ogrizovic et al.2024; Olszewska,2020; Zhu et al.,2020

Data item & categories		Count	% of corpus	References
	T3. AI-related software engineering topics (with implications for testing)	25	34.72	Ahmad et al., 2023a; Ahmad et al., 2023b; Alnafessah et al., 2021; Anwar & Bashir, 2023; Ferrara et al., 2024; Fischer et al., 2021; Giray, 2021; Guizzardi et al., 2023; Jabborov et al., 2023; Lavin et al., 2022; Martinez-Fernandez et al., 2022; Necula et al., 2024; Oedingen et al., 2024; Rodriguez et al., 2016; Sagodi et al., 2024; Saklamaeva & Pavlic, 2024; Silva-Rodriguez et al.,2020;Subha et al.,2023;Zhang&Jiang,2020;Almeida et al.,2024; Baumgartner et al.,2024;Izhar et al.,2025;Joeckel et al.,2021;Lu et al.,2022;Nikolaidis et al.,2024

REFERENCES

Abo-eleneen, A., Palliyali, A., & Catal, C. (2023). The role of reinforcement learning in software testing. *Information and Software Technology*, 164, 107325. https://doi.org/10.1016/j.infsof.2023.107325

Ahmad, K., Abdelrazek, M., Arora, C., Bano, M., & Grundy, J. (2023a). Requirements engineering framework for human-centered artificial intelligence software systems. *Applied Soft Computing*, 143, 110455. https://doi.org/10.1016/j.asoc.2023.110455

Ahmad, K., Abdelrazek, M., Arora, C., Bano, M., & Grundy, J. (2023b). Requirements practices and gaps when engineering human-centered Artificial Intelligence systems. *Applied Soft Computing*, 143, 110421. https://doi.org/10.1016/j.asoc.2023.110421

Alenezi, M., & Akour, M. (2025). AI-Driven Innovations in Software Engineering: A review of current practices and future directions. *Applied Sciences*, 15(3), 1344. https://doi.org/10.3390/app15031344

Ali, M., Mazhar, T., Shahzad, T., Ghadi, Y. Y., Mohsin, S. M., Akber, S. M. A., & Ali, M. (2023). Analysis of feature selection methods in software defect prediction models. *IEEE Access*, 11, 145954–145974. https://doi.org/10.1109/access.2023.3343249

Almeida, Y., Albuquerque, D., Filho, E. D., Muniz, F., De Farias Santos, K., Perkusich, M., Almeida, H., & Perkusich, A. (2024). AICodeReview: Advancing code quality with AI-enhanced reviews. *SoftwareX*, 26, 101677. https://doi.org/10.1016/j.softx.2024.101677

Alnafessah, A., Ul Gias, A., Wang, R., Zhu, L., Casale, G., & Filieri, A. (2021). Quality-aware DevOps research: Where do we stand? *IEEE Access*, 9, 44476–44489. https://doi.org/10.1109/ACCESS.2021.3064867

Amalfitano, D., Faralli, S., Hauck, J. C. R., Matalonga, S., & Distante, D. (2024). Artificial intelligence applied to software testing: A tertiary study. *ACM Computing Surveys*, 56(3), Article 58. https://doi.org/10.1145/3616372

Anwar, R., & Bashir, M. B. (2023). A systematic literature review of AI-based software requirements prioritization techniques. *IEEE Access*, 11, 143815–143860. https://doi.org/10.1109/ACCESS.2023.3343252

Augusto, C., Moran, J., Riva, C., & Tuya, J. (2019). Test-driven Anonymization for Artificial Intelligence. In *2019 IEEE International Conference on Artificial Intelligence Testing (AITest)*, (pp. 103–110). IEEE. https://doi.org/10.1109/AITest.2019.00011

Baumgartner, N., Iyenghar, P., Schoemaker, T., & Pulvermüller, E. (2024). AI-Driven Refactoring: A pipeline for identifying and correcting data clumps in Git repositories. *Electronics*, 13(9), 1644. https://doi.org/10.3390/electronics13091644

Borg, M. (2021). The AIQ Meta-Testbed: Pragmatically Bridging Academic AI Testing and Industrial Q Needs. In *Software Quality: Future Perspectives on Software Engineering Quality*, (pp. 66–77). Springer. https://doi.org/10.1007/978-3-030-65854-0_6

Boukhlif, M., Hanine, M., & Kharmoum, N. (2023). A decade of intelligent software testing research: A bibliometric analysis. *Electronics*, 12(9), 2109. https://doi.org/10.3390/electronics12092109

Cheng, K. S., Huang, P., Ahn, T., & Song, M. (2023). Tool support for improving software quality in machine learning programs. *Information*, 14(1), 53. https://doi.org/10.3390/info14010053

Deng, J., Lu, L., & Qiu, S. (2020). Software defect prediction via LSTM. *IET Software*, 14(4), 443–450. https://doi.org/10.1049/iet-sen.2019.0149

Durrani, U. K., Akpinar, M., Adak, M. F., Kabakus, A. T., Öztürk, M. M., & Saleh, M. (2024). A Decade of Progress: A systematic literature review on the integration of AI in software engineering phases and activities (2013-2023). *IEEE Access*, 12, 171185–171204. https://doi.org/10.1109/access.2024.3488904

Felderer, M., & Ramler, R. (2021). Quality Assurance for AI-Based Systems: Overview and Challenges (Introduction to Interactive Session). In *Software Quality: Future Perspectives on Software Engineering Quality*, (pp. 33–42). Springer. https://doi.org/10.1007/978-3-030-65854-0_3

Fernandes, P., Lopes, M., & Prada, R. (2021). Agents for Automated User Experience Testing. In *2021 IEEE International Conference on Software Testing, Verification and Validation Workshops*, (pp. 247–253). IEEE. https://doi.org/10.1109/ICSTW52544.2021.00049

Ferrara, C., Sellitto, G., Ferrucci, F., Palomba, F., & De Lucia, A. (2024). Fairness-aware machine learning engineering: How far are we? *Empirical Software Engineering*, 29(1), Article 9. https://doi.org/10.1007/s10664-023-10402-y

Filus, K., & Domanska, J. (2023). Software vulnerabilities in TensorFlow-based deep learning applications. *Computers & Security*, 124, 102948. https://doi.org/10.1016/j.cose.2022.102948

Fischer, L., Ehrlinger, L., Geist, V., Ramler, R., Sobiezky, F., Zellinger, W., Brunner, D., Kumar, M., & Moser, B. (2021). AI system engineering—Key challenges and lessons learned. *Machine Learning and Knowledge Extraction*, 3(1), 56–83. https://doi.org/10.3390/make3010004

Garrad, P., & Unnikrishnan, S. (2023). Reinforcement learning in VANET penetration testing. *Results in Engineering*, 17, 100970. https://doi.org/10.1016/j.rineng.2023.100970

Giray, G. (2021). A software engineering perspective on engineering machine learning systems: State of the art and challenges. *Journal of Systems and Software*, 180, 111031. https://doi.org/10.1016/j.jss.2021.111031

Giuliano M., A., Martin-Lopez, A., Segura, S., Valencia-Cabrera, L., & Ruiz-Cortes, A. (2021). Deep Learning-Based Prediction of Test Input Validity for RESTful APIs. In *2021 IEEE/ACM Third International Workshop on Deep Learning for Testing and Testing for Deep Learning*, (pp. 9–16). IEEE. https://doi.org/10.1109/DeepTest52559.2021.00008

Guizzardi, R., Amaral, G., Guizzardi, G., & Mylopoulos, J. (2023). An ontology-based approach to engineering ethicality requirements. *Software and Systems Modeling*, 22(6), 1897–1923. https://doi.org/10.1007/s10270-023-01115-3

Gurcan, F., Dalveren, G. G. M., Cagiltay, N. E., Roman, D., & Soylu, A. (2022). Evolution of Software testing Strategies and Trends: Semantic content analysis of software research corpus of the last 40 years. *IEEE Access*, 10, 106093–106109. https://doi.org/10.1109/access.2022.3211949

Hourani, H., Hammad, A., & Lafi, M. (2019). The impact of artificial intelligence on software testing. In *2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology,* (pp. 565–570). IEEE. https://doi.org/10.1109/JEEIT.2019.8717439

Izhar, R., Bhatti, S. N., & Alharthi, S. A. (2025). Bridging Precision and Complexity: A novel machine learning approach for ambiguity detection in software requirements. *IEEE Access*, 13, 12014–12031. https://doi.org/10.1109/access.2025.3529943

Jabborov, A., Kharlamova, A., Kholmatova, Z., Kruglov, A., Kruglov, V., & Succi, G. (2023). Taxonomy of quality assessment for intelligent software systems: A systematic literature review. *IEEE Access*, 11, 130491–130507. https://doi.org/10.1109/ACCESS.2023.3333920

Joeckel, L., Bauer, T., Klaes, M., Hauer, M., & Gross, J. (2021). Towards a Common Testing Terminology for Software Engineering and Data Science Experts. In *Product-Focused Software Process Improvement*, (pp. 281–289). Springer. https://doi.org/10.1007/978-3-030-91452-3_19

Khan, M. F. I., Mahmud, F., Hossen, A., & Masum, A. (2024). A new approach of software test automation using AI. *Journal of Basic Science and Engineering*, 21, 559–570.

Khaleel, S., & Anan, R. (2023). A review paper: Optimal test cases for regression testing using artificial intelligent techniques. *International Journal of Electrical and Computer Engineering*, 13, 1803–1816, http://doi.org/10.11591/ijece.v13i2.pp1803-1816

Khatibsyarbini, M., Isa, M. A., Jawawi, D. N. A., Hamed, H. N. A., & Suffian, M. D. M. (2019). Test case prioritization using firefly algorithm for software testing. *IEEE Access*, 7, 132360–132373. https://doi.org/10.1109/ACCESS.2019.2940620

Kiran, A., Butt, W. H., Anwar, M. W., Azam, F., & Maqbool, B. (2019). A comprehensive investigation of modern test suite optimization trends, tools and techniques. *IEEE Access*, 7, 89093–89117. https://doi.org/10.1109/ACCESS.2019.2926384

Kokol, P. (2024). The use of AI in software engineering: A synthetic knowledge synthesis of the recent research literature. *Information*, 15(6), 354. https://doi.org/10.3390/info15060354

Kusharki, M. B., Misra, S., Muhammad-Bello, B., Salihu, I. A., & Suri, B. (2022). Automatic classification of equivalent mutants in mutation testing of Android applications. *Symmetry*, 14(4), 820. https://doi.org/10.3390/sym14040820

Lavin, A., Gilligan-Lee, C. M., Visnjic, A., Ganju, S., Newman, D., Ganguly, S., Lange, D., Baydin, A. G., Sharma, A., Gibson, A., Zheng, S., Xing, E. P., Mattmann, C., Parr, J., & Gal, Y. (2022). Technology readiness levels for machine learning systems. *Nature Communications*, 13(1), 6039. https://doi.org/10.1038/s41467-022-33128-9

Layman, L., & Vetter, R. (2024). Generative artificial intelligence and the future of software testing. *Computer*, 57(1), 27–32. https://doi.org/10.1109/MC.2023.3306998

Lee, D.-G., & Seo, Y.-S. (2020). Improving bug report triage performance using artificial intelligence based document generation model. *Human-centric Computing and Information Sciences*, 10(1), 26. https://doi.org/10.1186/s13673-020-00229-7

Leotta, M., Ricca, F., Stoppa, S., & Marchetto, A. (2022). Is NLP-based Test Automation Cheaper Than Programmable and Capture & Replay?. In *Quality of Information and Communications Technology*, (pp. 77–92). Springer. https://doi.org/10.1007/978-3-031-14179-9_6

Li, Y., Liu, P., Wang, H., Chu, J., & Wong, W. E. (2025). Evaluating large language models for software testing. *Computer Standards & Interfaces*, 93, 103942. https://doi.org/10.1016/j.csi.2024.103942

Liaqat, A., Sindhu, M. A., & Siddiqui, G. F. (2020). Metamorphic testing of an artificially intelligent chess game. *IEEE Access*, 8, 174179–174190. https://doi.org/10.1109/ACCESS.2020.3024929

Lima, R., Rosado da Cruz, A. M., & Ribeiro, J. (2020). Artificial intelligence applied to software testing: A literature review. In *15th Iberian Conference on Information Systems and Technologies*, (pp. 1–6). IEEE. https://doi.org/10.23919/CISTI49556.2020.9141124

Liu, Z., Su, T., Zakharov, M. A., Wei, G., & Lee, S. (2025). Software defect prediction based on residual/shuffle network optimized by upgraded fish migration optimization algorithm. *Scientific Reports*, 15(1), 7201. https://doi.org/10.1038/s41598-025-91784-5

Lu, Q., Zhu, L., Xu, X., Whittle, J., Douglas, D., & Sanderson, C. (2022). Software engineering for Responsible AI: An empirical study and operationalised patterns. In *2022 IEEE/ACM 44th International Conference on Software Engineering: Software Engineering in Practice*, (pp. 241–242). ACM. https://doi.org/10.1145/3510457.3513063

Martinez-Fernandez, S., Bogner, J., Franch, X., Oriol, M., Siebert, J., Trendowicz, A., Vollmer, A.M. & Wagner, S. (2022). Software engineering for AI-based systems: A survey. *ACM Transactions on Software Engineering and Methodology*, 31(2), 37e. https://doi.org/10.1145/3487043

Mustaqeem, M., Alam, M., Mustajab, S., Alshanketi, F., Alam, S., & Shuaib, M. (2025). Comprehensive Bibliographic Survey and Forward-Looking Recommendations for Software Defect Prediction: Datasets, Validation Methodologies, Prediction Approaches, and Tools. *IEEE Access*, 13, 866–903., https://doi.org/10.1109/ACCESS.2024.3517419

Myllyaho, L., Raatikainen, M., Mannisto, T., Mikkonen, T., & Nurminen, J. K. (2021). Systematic literature review of validation methods for AI systems. *Journal of Systems and Software*, 181, 111050. https://doi.org/10.1016/j.jss.2021.111050

Navaei, M., & Tabrizi, N. (2022). Machine Learning in Software Development Life Cycle: A Comprehensive Review. In *Proceedings of the 17th International Conference on Evaluation of Novel Approaches to Software Engineering*, (pp. 344–354). ScitePress. https://doi.org/10.5220/0011040600003176

Necula, S.-C., Dumitriu, F., & Greavu-Serban, V. (2024). A systematic literature review on using natural language processing in software requirements engineering. *Electronics*, 13(11), 2055. https://doi.org/10.3390/electronics13112055

Nguyen, D. P., & Maag, S. (2020). Codeless Web Testing using Selenium and Machine Learning. In *Proceedings of the 15th International Conference on Software Technologies*, (pp. 51–60). ScitePress. https://doi.org/10.5220/0009885400510060

Nikolaidis, N., Flamos, K., Gulati, K., Feitosa, D., Ampatzoglou, A., & Chatzigeorgiou, A. (2024). A Comparison of the Effectiveness of ChatGPT and Co-Pilot for Generating Quality Python Code Solutions. In *2024 IEEE International Conference on Software Analysis, Evolution and Reengineering*, (pp. 93–101). IEEE. https://doi.org/10.1109/SANER-C62648.2024.00018

Oedingen, M., Engelhardt, R. C., Denz, R., Hammer, M., & Konen, W. (2024). ChatGPT code detection: Techniques for uncovering the source of code. *AI*, 5(3), 1066–1094. https://doi.org/10.3390/ai5030053

Ogrizović, M., Drašković, D., & Bojić, D. (2024). Quality assurance strategies for machine learning applications in big data analytics: an overview. *Journal of Big Data*, 11(1), 156. https://doi.org/10.1186/s40537-024-01028-y

Olszewska, J. (2020). AI-T: Software Testing Ontology for AI-based Systems. In *Proceedings of the 12th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*, (pp. 291–298). ScitePress. https://doi.org/10.5220/0010147902910298

Pandit, M., Gupta D., Anand D., Goyal N., Aljahdali H. M., Mansilla A. O., Kadry S., Kumar A. (2022). Towards design and feasibility analysis of DePaaS: AI based global unified software defect prediction framework. *Applied Sciences*, 12(1), 493. https://doi.org/10.3390/app12010493

Poth, A., Beck, Q., & Riel, A. (2019). Artificial Intelligence Helps Making Quality Assurance Processes Leaner. In *Systems, Software and Services Process Improvement,* (pp. 722–730). Springer. https://doi.org/10.1007/978-3-030-28005-5_56

Rahman, T., Singh, R., & Sultan, M. (2024). Automating Patch Set Generation from Code Review Comments Using Large Language Models. In *Proceedings of the IEEE/ACM 3rd International Conference on AI Engineering*, (pp. 273–274). ACM. https://doi.org/10.1145/3644815.3644981

Rajapaksha, D., Tantithamthavorn, C., Jiarpakdee, J., Bergmeir, C., Grundy, J., & Buntine, W. (2022). SQAPlanner: Generating data-informed software quality improvement plans. *IEEE Transactions on Software Engineering*, 48(8), 2814–2835. https://doi.org/10.1109/TSE.2021.3070559

Ramirez, A., Berrios, M., Raul Romero, J., & Feldt, R. (2023). Towards Explainable Test Case Prioritisation with Learning-to-Rank Models. In *2023 IEEE International Conference on Software Testing, Verification and Validation Workshops*, (pp. 66–69). IEEE. https://doi.org/10.1109/ICSTW58534.2023.00023

Rodriguez, G., Soria, A., & Campo, M. (2016). Artificial intelligence in service-oriented software design. *Engineering Applications of Artificial Intelligence*, 53, 86–104. https://doi.org/10.1016/j.engappai.2016.03.009

Sagodi, Z., Siket, I., & Ferenc, R. (2024). Methodology for code synthesis evaluation of LLMs presented by a case study of ChatGPT and Copilot. *IEEE Access*, 12, 72303–72316. https://doi.org/10.1109/ACCESS.2024.3403858

Saklamaeva, V., & Pavlic, L. (2024). The potential of AI-driven assistants in scaled agile software development. *Applied Sciences*, 14(1), 319. https://doi.org/10.3390/app14010319

Silva-Rodriguez, V., Nava-Munoz, S. E., Castro, L. A., Martinez-Perez, F. E., Perez-Gonzalez, H. G., & Torres-Reyes, F. (2020). Classifying design-level requirements using machine learning for a recommender of interaction design patterns. *IET Software*, 14(5), 544–552. https://doi.org/10.1049/iet-sen.2019.0291

Sofian, H., Yunus, N. A. M., & Ahmad, R. (2022). Systematic mapping: Artificial intelligence techniques in software engineering. *IEEE Access*, 10, 51021–51040. https://doi.org/10.1109/ACCESS.2022.3174115

Steidl, M., Felderer, M., & Ramler, R. (2023). The pipeline for the continuous development of artificial intelligence models—Current state of research and practice. *Journal of Systems and Software*, 199, 111615. https://doi.org/10.1016/j.jss.2023.111615

Strandberg, P., Frasheri, M., & Enoiu, E. (2021). Ethical AI-Powered Regression Test Selection. In 2021 IEEE International Conference on Artificial Intelligence Testing (AITest), (pp. 83–84). IEEE. https://doi.org/10.1109/AITEST52744.2021.00025

Subha, R., Haldorai, A., & Ramu, A. (2023). Artificial intelligence model for software reusability prediction system. Intelligent Automation & Soft Computing, 35(3), 2639–2654. https://doi.org/10.32604/iasc.2023.028153

Tahvili, S., Hatvani, L., Ramentol, E., Pimentel, R., Afzal, W., & Herrera, F. (2020). A novel methodology to classify test cases using natural language processing and imbalanced learning. *Engineering Applications of Artificial Intelligence*, 95, 103878. https://doi.org/10.1016/j.engappai.2020.103878

Tao, C., Gao, J., & Wang, T. (2019). Testing and quality validation for AI software—Perspectives, issues, and practices. *IEEE Access*, 7, 120164–120175. https://doi.org/10.1109/ACCESS.2019.2937107

Tosi, D. (2024). Studying the quality of source code generated by different AI generative engines: An empirical evaluation. *Future Internet*, 16(6), 188. https://doi.org/10.3390/fi16060188

Van Eck, N. J., & Waltman, L. (2023). VOSviewer (Version 1.6.20) [Software]. *Centre for Science and Technology Studies*. https://www.vosviewer.com/download

Vinayakumar, R., Alazab, M., Soman, K. P., Poornachandran, P., & Venkatraman, S. (2019). Robust intelligent malware detection using deep learning. *IEEE Access*, 7, 46717–46738. https://doi.org/10.1109/ACCESS.2019.2906934

Wang, Y., Guo, S., & Tan, C. W. (2025). From code generation to software testing: AI Copilot with Context-Based Retrieval-Augmented Generation. *IEEE Software*, 42(4), 34–42. https://doi.org/10.1109/ms.2025.3549628

Xie, X., Zhang, Z., Chen, T. Y., Liu, Y., Poon, P.-L., & Xu, B. (2020). METTLE: A METamorphic testing approach to assessing and validating unsupervised machine learning systems. *IEEE Transactions on Reliability*, 69(4), 1293–1322. https://doi.org/10.1109/TR.2020.2972266

Zhang, X., & Jiang, Y. (2020). Research and application of machine learning in automatic program generation. *Chinese Journal of Electronics*, 29(6), 1001–1015. https://doi.org/10.1049/cje.2020.10.006

Zhu, H., Bayley, I., Liu, D., & Zheng, X. (2020). Automation of Datamorphic Testing. In 2020 IEEE International Conference On Artificial Intelligence Testing (AITest), (pp. 64–72). IEEE. https://doi.org/10.1109/AITEST49225.2020.00017

Zhu, H., & Bayley, I. (2022). Discovering boundary values of feature-based machine learning classifiers through exploratory data-morphic testing. *Journal of Systems and Software*, 187, 111231. https://doi.org/10.1016/j.jss.2022.111231

Acta Informatica Pragensia is published by the Prague University of Economics and Business, Czech Republic | eISSN: 1805-4951