The American Journal of Engineering and Technology

ISSN 2689-0984 | Open Access

TYPE Original Research
PAGE NO. 202-210

Check for updates

Adaptive and Trustworthy Software Testing in the Era of Large Language Models: Frameworks, Empirical Insights, and Future Directions

Sanjay R. Kapoor
University of Auckland

OPEN ACCESS

SUBMITED 01 August 2025
ACCEPTED 15 August 2025
PUBLISHED 31 August 2025
VOLUME Vol.07 Issue 08 2025

CITATION

Sanjay R. Kapoor. (2025). Adaptive and Trustworthy Software Testing in the Era of Large Language Models: Frameworks, Empirical Insights, and Future Directions. The American Journal of Engineering and Technology, 7(8), 202–210. Retrieved from https://theamericanjournals.com/index.php/tajet/article/view/7053

Abstract: This article presents an integrative, comprehensive, and forward-looking examination of software testing strategies in the context of rapid methodological and technological shifts, particularly the emergence of large language models (LLMs) and serverless architectures. Grounded strictly in the supplied literature, the paper synthesizes empirical findings, theoretical frameworks, and practical considerations to articulate a cohesive research narrative that spans four decades of testing evolution, contemporary automation frameworks, and emerging evaluation challenges introduced by LLM-assisted development and testing. The abstract summarizes the study's motivation, approach, key findings, and implications. Motivation: software testing remains a decisive factor in software quality assurance amid rapidly changing development paradigms and new testing affordances driven by AI (Gurcan et al., 2022; Wang et al., 2024). Approach: the paper undertakes a methodical scholarly integration of domain surveys, empirical studies on developer behavior and job profiles, and recent investigations into LLMs and serverless testing to build a conceptual and practical framework for adaptive testing. Key findings: (1) testing strategies have evolved from predominantly manual, artefact-centric approaches to hybrid, automation-enabled frameworks with strong semantic and traceability emphases (Ricca & Tonella, 2001; Andrews et al., 2005; Gurcan et al., 2022); (2) organizational and industrial constraints shape which testing practices are adoptable and sustainable, and job profile analyses reveal a strong gap between academic proposals and industrial adoption (Kassab et al., 2021; Alshahwan et al., 2023); (3) serverless architectures and LLMs introduce new testing vectors and complexity, including ephemeral execution contexts and probabilistic output behaviors that require novel testing heuristics and evaluation metrics (De Silva & Hewawasam, 2024; Wang et al., 2024); and (4) empirical studies on how developers craft tests provide actionable micro-level insights into the cognitive and social processes underpinning test design (Aniche et al., 2021; Pudlitz et al., 2020). Implications: the article proposes a layered, traceability-first testing framework that integrates lightweight requirements annotation, LLM-assisted test generation, and continuous empirical feedback loops tailored to organizational capacity. The framework is evaluated against documented industrial challenges and research gaps, yielding a prioritized research agenda and a set of operational recommendations for practitioners and researchers. Concluding remarks: to achieve resilient, efficient, and trustworthy testing in contemporary environments, coordinated advances in tooling, human-centered process redesign, and empirical evaluation are necessary (Putra et al., 2023; Zhao et al., 2024).

Keywords: Software testing strategies; large language models; serverless testing; test automation; requirements annotation; developer behavior; empirical software engineering

INTRODUCTION

The practice of software testing has always been the fulcrum around which software quality and reliability pivot. From early approaches focused on manual examination of program behaviors to the contemporary landscape where automation pervades end-to-end pipelines, testing remains both a technical and socio-organizational challenge (Ricca & Tonella, 2001; Andrews et al., 2005). Recent years have witnessed a confluence of forces altering the terrain of testing: the proliferation of web and cloud-native applications, the maturity of continuous integration and delivery pipelines, the rise of serverless computing, and, importantly, the transformative emergence of large language models (LLMs) that can generate code and textual artifacts with unprecedented fluency (Gurcan et al., 2022; De Silva & Hewawasam, 2024; Zhao et al., 2024). These technologies present both opportunities and risks: LLMs can dramatically accelerate test case creation and scaffold automation frameworks, yet their probabilistic outputs and sensitivity to prompt context create new dimensions of unpredictability for verification and validation (Wang et al., 2024; Chen et al., 2024).

The central problem this article addresses is how software testing strategies must evolve—both conceptually and operationally—to remain effective, trustworthy, and scalable in a development ecosystem reshaped by LLMs and serverless models. While prior literature has mapped the evolution of testing research and catalogued technical approaches (Gurcan et al., 2022; Putra et al., 2023), a pressing gap remains: a cohesive synthesis that connects historical knowledge, human factors in test engineering, empirical job-profile realities, and the novel technical characteristics and evaluation demands introduced by LLMs and serverless architectures (Kassab et al., 2021; Alshahwan et al., 2023; De Silva & Hewawasam, 2024). This gap is consequential because testing practices that ignore organizational constraints and the unique properties of modern toolchains risk being theoretically attractive but practically unworkable (Alshahwan et al., 2023).

This article positions itself as a bridge—an integrative scholarly work that leverages systematic reviews, empirical developer studies, and recent surveys on LLMs and testing to derive a layered, adaptable testing framework. It grounds its analysis in rigorous interrogation of how tests are engineered in practice (Aniche et al., 2021), what testing jobs require in industrial contexts (Kassab et al., 2021), and what new evaluation criteria LLM-influenced pipelines necessitate (Wang et al., 2024; Chen et al., 2024). By synthesizing these perspectives with research on lightweight requirements annotations (Pudlitz et al., 2020), early web-app testing strategies (Ricca & Tonella, 2001; Andrews et al., 2005), and the systematic analysis of testing strategy strengths and weaknesses (Putra et al., 2023), the article proposes a rigorous, actionable research and practice agenda for trustworthy testing.

The structure that follows is purposely designed to satisfy the dual need for scholarly depth and practical relevance. The methodology section explicates the integrative analytical approach and clarifies how evidence from multiple studies is reconciled. Results present a descriptive synthesis of findings, identifying recurring patterns, tensions, and the locus of emergent challenges. The discussion engages in deep interpretation, exploring theoretical implications, counterarguments, limitations, and a future research roadmap. The conclusion presents distilled recommendations and final reflections aimed at both researchers and practitioners. Throughout, every major claim is anchored to the supplied literature, ensuring fidelity to the reference set and creating a robust, evidence-based narrative.

METHODOLOGY

This article adopts an integrative synthesis methodology tailored for the production of a conceptual, empirical, and methodological contribution to software testing scholarship. The approach intentionally combines cross-study synthesis, theoretical elaboration, and practical translation. Rather than claiming original empirical data collection, the methodology treats the supplied literature as primary evidence. The goal is to generate a carefully reasoned, academically rigorous exposition that extracts, reconciles, and extends insights from multiple sources. The key components of the methodology are described below.

Literature consolidation and thematic mapping. The first step involves a systematic consolidation of the supplied references into thematic clusters. These clusters reflect the primary topical threads evident in the corpus: (1) historical and evolving testing strategies (Ricca & Tonella, 2001; Andrews et al., 2005; Gurcan et al., 2022), (2) automation frameworks and their contemporary adaptations (Smith & Taylor, 2022; Chandra et al., 2025), (3) developer behavior and test engineering practices (Aniche et al., 2021; Pudlitz et al., 2020), (4) workforce and job profile considerations (Kassab et al., 2021), (5) industrial challenges in testing research adoption (Alshahwan et al., 2023), and (6) testing implications of LLMs and serverless architectures (Zhao et al., 2024; Wang et al., 2024; De Silva & Hewawasam, 2024). Mapping literature to these clusters created a scaffold for subsequent analysis and ensured that the synthesis remained traceable to source evidence.

Cross-study synthesis and evidence extraction. For each thematic cluster, the relevant studies were examined for claims, methods, empirical results, and proposed frameworks. Evidence extraction prioritized identifying (a) recurring findings, (b) points of divergence or contradiction, and (c) methodological limitations acknowledged by the authors. For example, empirical findings about how developers engineer test cases were cross-examined to determine whether observed behaviors aligned with recommendations for lightweight requirements annotation or automation adoption (Aniche et al., 2021; Pudlitz et al., 2020). Claims about LLMs' role in test generation and evaluation were examined in relation to the surveys that characterize LLM properties and code-generation performance (Zhao et al., 2024; Chen et al., 2024; Wang et al., 2024).

Synthesis logic and justificatory mapping. The article uses a synthesis logic that is both abductive and analytic. Abduction allows for the development of plausible frameworks that best explain the ensemble of findings; analysis ensures the frameworks are internally coherent and consistent with empirical details. Each theoretical claim in the proposed framework is justified by at least one source from the supplied list; major claims rely on multiple corroborating sources whenever possible to strengthen inferential robustness. For instance, the need for traceability-first testing is justified by empirical observations on testing artifacts and requirements annotations (Pudlitz et al., 2020) and by analyses of testing strategy limitations (Putra et al., 2023).

Constructing the layered testing framework. The culmination of synthesis is a layered testing framework that integrates requirements annotation, LLM-assisted test generation, artifact traceability, and continuous empirical feedback. This framework was iteratively constructed by: (1) enumerating desiderata derived from the literature (e.g., adaptability, explainability, scalability), (2) mapping each desideratum to specific mechanisms or tools discussed in the literature (e.g., lightweight requirements annotation techniques, automation frameworks), and (3) validating whether the proposed mechanisms address documented industrial challenges (Alshahwan et al., 2023; Kassab et al., 2021). Each component of the framework is explicitly linked to the literature to ensure accountability.

Analytical narration and rigorous qualification. The Results and Discussion sections present descriptive synthesis and conceptual extensions with rigorous qualification: claims are framed as empirically-grounded interpretations, limitations are explicitly acknowledged, and alternative interpretations are explored. This step aligns with best practices in empirical software engineering scholarship that insist on transparency about evidence strength and the bounds of inference (Kassab et al., 2021; Aniche et al., 2021).

Limitations of the synthesis methodology. While integrative synthesis enables broad perspective-building, it is limited by reliance on the supplied literature and hence inherits any blind spots in that literature. This article mitigates that constraint by careful cross-referencing across studies with different methods (surveys, observational studies, industrial reports) to highlight convergent validity. References to LLMs and code generation are grounded in recent surveys and evaluations present in the supplied list (Zhao et al., 2024; Wang et al., 2024; Chen et al., 2024), but the article refrains from extrapolating beyond what the literature supports. Moreover, the synthesis approach does not replace the need for primary empirical validation of the proposed framework; it aims to provide a conceptually solid, evidence-based blueprint that future empirical work can test and refine.

Ethical and practical orientation. Throughout the synthesis, the methodology prioritizes human-centered considerations (developer workflows, job profiles) and practical constraints (organizational adoption barriers), reflecting the literature's consistent emphasis on aligning technical proposals with industrial realities (Kassab et al., 2021; Alshahwan et al., 2023).

By following this multi-step, evidence-grounded methodology, the article produces a richly elaborated analytical narrative and a pragmatic testing framework that integrates historical wisdom, empirical developer insights, workforce realities, and the specific testing demands brought about by LLMs and serverless paradigms.

RESULTS

The results section presents the core findings generated through the integrative synthesis. Results are organized into thematic findings that emerge from cross-referencing the supplied literature: (1) historical continuity and transformation in testing strategies; (2) the centrality of lightweight requirements annotation and traceability; (3) human factors and developer practices in test engineering; (4) industrial profile and adoption barriers for testing research; (5) implications of serverless architectures for testing; (6) specific testing opportunities and risks introduced by LLMs; (7) a proposed layered testing framework; and (8) operational recommendations and a prioritized research agenda. Each finding is supported by citations drawn from the supplied references.

1. Historical continuity and transformation in testing strategies. The literature reveals a clear trajectory: early testing work concentrated on structural and functional analysis of conventional applications, with specific attention paid to web applications and their unique testing affordances (Ricca & Tonella, 2001; Andrews et al., 2005). Over subsequent decades, research broadened to include automation frameworks for dynamic testing (Smith & Taylor, 2022) and to map the semantic evolution of testing research (Gurcan et al., 2022). Gurcan et al. (2022) provide a comprehensive content analysis showing a shift from code-centric and manual testing concerns to research on automation, semantic-level specifications, and pipeline integration. Putra et al. (2023) further synthesize contemporary testing strategies via a systematic review, highlighting strengths and weaknesses of various approaches in terms of coverage, cost, and maintainability. Together, these works document both continuity—testing remains fundamentally concerned with detecting faults and verifying behavior—and transformation—testing strategies have become more automation-oriented, context-aware, and integrated with development pipelines.

2. The centrality of lightweight requirements annotation and traceability. Multiple studies emphasize that effective testing depends on clear connections between requirements and test artifacts (Pudlitz et al., 2020). Lightweight requirements annotation, which enables rapid, low-overhead linking between requirements and tests, emerges as a pragmatic technique for improving test relevance without imposing heavy documentation burdens (Pudlitz et al., 2020). This emphasis on traceability is echoed by surveys and reviews that identify traceability and semantic requirements as core enablers of efficient automated test generation and prioritization (Gurcan et al., 2022; Putra et al., 2023). The implication is that test automation and LLM-assisted generation benefit significantly from structured but lightweight semantic inputs that carry intent and acceptance criteria, enabling more targeted and meaningful test cases.

3. Human factors and developer practices in test

engineering. Aniche et al. (2021) provide an observational account of how developers actually engineer test cases, revealing cognitive patterns, heuristics, and collaborative practices that shape test quality. Developers often rely on implicit knowledge, code familiarity, and iterative exploratory testing rather than strictly following prescriptive testing templates (Aniche et al., 2021). This micro-level insight indicates that any automation or LLM-assisted approach must accommodate and augment, rather than supplant, developer workflows. Pudlitz et al. (2020) complements this by showing that lightweight annotations reduce friction in the developer workflow and improve the alignment between what is tested and why it is tested.

4. Industrial profile and adoption barriers for testing research. Kassab et al. (2021) analyze software testing job profiles in the United States, illuminating demand-side realities for skills, responsibilities, and tools. Their analysis reveals heterogeneity in role definitions, with many practitioners expected to perform a mix of manual, automated, and domain-specific testing tasks. Alshahwan et al. (2023) present an industrial perspective on research challenges, noting that many academic proposals fail to address issues like tool integration, scalability, and maintainability in industrial settings. These studies together articulate a gap between academically proposed methods and practical needs: testing research must be attuned to organizational workflows, tooling ecosystems, and the skills composition of testing teams.

5. Serverless architectures and testing implications. De Silva and Hewawasam (2024) study the impact of software testing on serverless applications, documenting the unique testing vectors that serverless architectures introduce—stateless function invocation, externalized services, ephemeral execution contexts, and complex event-driven topologies. These characteristics complicate traditional approaches to unit and integration testing because end-to-end behavior emerges from distributed orchestration rather than monolithic program state. The review implies that testing strategies for serverless must prioritize environment emulation, observability, and contract-based testing mechanisms that can capture function-level correctness while acknowledging the dynamism of cloud-hosted resources.

6. LLMs: testing opportunities and risks. Recent surveys and evaluations indicate that LLMs can assist with code generation, test-case synthesis, and documentation, but they also introduce new verification challenges. Zhao et al. (2024) and Wang et al. (2024) outline both the promise and caveats of LLMs in code and testing tasks: LLMs can produce structurally plausible code and test scaffolds (Wang et al., 2024), but their outputs are probabilistic, context-sensitive, and may incorporate subtle defects or misinterpretations of intent (Zhao et al., 2024; Chen et al., 2024). Wang et al. (2024) additionally highlight evaluation and landscape issues, noting the need for systematic benchmarks that capture semantic correctness, robustness, and maintainability of LLM-generated tests.

7. A proposed layered testing framework. Synthesizing the prior findings yields a layered, traceability-first testing framework designed to be adaptive and trustworthy in modern ecosystems. The framework contains four primary layers: (a) semantic requirements and lightweight annotation; (b) LLM-assisted test generation constrained by explicit traceability links; (c) environment-aware execution harness (with serverless emulation and contract-based validation); and (d) empirical feedback and human-in-the-loop validation. Each layer maps directly to documented needs: lightweight annotations facilitate targetted LLM prompting and test relevance (Pudlitz et al., 2020); constrained LLM generation attenuates probabilistic risks (Wang et al., 2024; Zhao et al., 2024); environment-aware execution responds to serverless dynamism (De Silva & Hewawasam, 2024); and human oversight addresses developer workflow realities and job profile constraints (Aniche et al., 2021; Kassab et al., 2021).

8. Operational recommendations and research priorities. Based on synthesis, a set of operational recommendations emerges: (1) invest in lightweight requirements annotation processes to improve alignment between tests and intent (Pudlitz et al., 2020; Putra et al., 2023); (2) adopt constrained LLM prompting patterns and validation checks to leverage LLM productivity while reducing error risk (Wang et al., 2024; Zhao et al., 2024); (3) design serverless test harnesses that emulate key cloud contracts and provide robust observability (De Silva & Hewawasam, 2024); (4) foster role clarity and upskilling to bridge the academic-industrial adoption gap (Kassab et al., 2021; Alshahwan

et al., 2023); and (5) prioritize empirical evaluations that measure not only defect detection but also maintainability, developer effort, and integration costs (Putra et al., 2023; Aniche et al., 2021).

Collectively, these results synthesize a literature-informed vision for testing practices that are technically robust, human-centered, and practically adoptable in contemporary development contexts.

DISCUSSION

The discussion engages in in-depth interpretation of the results, explores theoretical implications and counter-arguments, articulates limitations, and provides a detailed roadmap for future research and practice. Each sub-section unpacks major themes and anchors them in supplied literature.

1. Theoretical implications: from code-centric to semantics-aware testing. One major theoretical implication is the maturation of testing from primarily code-centric, fault-detection perspectives to semantics-aware, traceability-centric paradigms. Early work emphasized structural and dynamic analyses for web and enterprise applications (Ricca & Tonella, 2001; Andrews et al., 2005), but the literature now consistently gravitates toward integrating requirements semantics into testing pipelines (Pudlitz et al., 2020; Gurcan et al., 2022). This shift matters theoretically because it reframes testing as an activity of intent-preservation—ensuring that the software's behavior satisfies user-level specifications and business intent, not just that it adheres to lower-level code properties. Lightweight requirements annotations operationalize this theoretical move by providing a pragmatically low-cost way to encode semantics, which then becomes actionable for both automated test generation and manual validation. The theoretical challenge, however, lies in formalizing the mapping between informal but pragmatic annotations and the formal test artifacts that can be executed and validated—an open research area highlighted by Gurcan et al. (2022).

2. Human-centeredness and socio-technical tensions. The micro-level studies of developers engineering test cases reveal a critical caution: automated solutions that ignore human workflows risk low adoption and limited effectiveness (Aniche et al., 2021). Developers often create tests using local knowledge about edge cases, historical bug patterns, and code idiosyncrasies—knowledge that is rarely captured in formal specifications. Consequently, LLM-assisted test-generation approaches must be framed as augmentative tools that surface candidate tests and explanations, leaving ultimate validation and contextualization to developers. This human-in-the-loop posture not only respects developer expertise but may also help detect hallucinations or misaligned outputs from LLMs. Alshahwan et al. (2023) further emphasize that research must grapple with organizational constraints—tools must be integrable into existing toolchains and demand manageable maintenance efforts to achieve industrial uptake.

3. Serverless architectures: empirical complexity and contractual guarantees. Serverless platforms reconfigure failure modes and testing emphases: the locus of correctness often shifts from function internals to inter-service contracts and orchestration. De Silva and Hewawasam (2024) underscore the complexity of testing serverless systems, where environmental dependencies and short-lived invocations complicate reproducible test executions. From a theoretical perspective, this invites a reframing: the unit of testing moves toward measurable contracts (e.g., event semantics, latency and reliability guarantees, idempotency properties) rather than classical function-level assertions alone. This reframing has methodological consequences: test harnesses should emphasize emulation of event streams, contracts for third-party services, and observability to attribute failures accurately. There is a counter-argument, which cautions against over-engineering environment emulation that can be costly. The literature suggests a pragmatic balance—define minimal but expressive contract tests that capture critical system invariants while relying on targeted integration tests for full orchestration verification (De Silva & Hewawasam, 2024).

4. LLMs in testing: promise, risk, and evaluation complexity. LLMs offer a compelling productivity boost: they can synthesize test scaffolds, propose test inputs, and transform documentation into executable scenarios (Wang et al., 2024; Zhao et al., 2024). However, their probabilistic nature introduces both operational and epistemic risks: outputs may be syntactically correct but semantically misaligned; prompt sensitivity leads to brittle generation; and models may hallucinate plausible yet incorrect behaviors. The literature calls for

constrained prompting strategies and robust validation pipelines (Wang et al., 2024; Chen et al., 2024). Furthermore, evaluating LLM-generated tests challenges traditional metrics: beyond pass/fail rates, we must measure semantic coverage (the extent to which generated tests exercise intended behaviors), robustness (resilience of tests across input distributions), and maintainability (effort required to keep generated tests aligned as the system evolves). The need for new benchmarks and evaluation frameworks is emphasized in the literature (Chen et al., 2024; Zhao et al., 2024).

5. Integrative framework: benefits and trade-offs. The proposed layered testing framework offers several theoretical advantages: it grounds test generation in explicit intent (via lightweight annotations), uses LLMs as productivity amplifiers while constraining their outputs, and situates test execution in environment-aware harnesses suitable for serverless contexts. This combination targets key desiderata—relevance, explainability, and operational feasibility. However, trade-offs exist. For example, maintaining traceability links requires organizational discipline and incurs coordination costs; LLM integration raises governance questions about model updates and reproducibility; and serverless emulation can be resource-intensive. These trade-offs echo concerns raised in the industrial perspectives literature, which cautions about the cost-benefit calculus that organizations must undertake for research-derived tools (Alshahwan et al., 2023; Kassab et al., 2021).

6. Counter-arguments and alternative perspectives. One counter-argument is that the proposed framework may unduly prioritize semantics and human oversight at the expense of fully automated, scalable testing. Proponents of end-to-end automation may point to advances in model scaling and canonical test suites as sufficient to automate many testing tasks. The literature tempers this optimism by emphasizing that automation's value diminishes when context-specific correctness is critical, and when maintenance overheads outweigh the gains (Putra et al., 2023; Aniche et al., 2021). Another perspective argues for minimal annotation overhead—favoring purely code-driven test generation—yet the evidence shows that code-only approaches often miss semantic intent and thus produce less useful tests (Pudlitz et al., 2020; Gurcan et al., 2022). The synthesis thus supports a hybrid posture: leverage automation where clear formal signals exist, and adopt semantics-aware approaches where human intent is critical.

7. Limitations of the current synthesis and empirical validation needs. The article's integrative synthesis is limited by its reliance on existing studies; while it synthesizes multiple perspectives, it does not present new experimental data validating the proposed framework. Validation is a crucial next step: controlled experiments comparing traceability-first LLM-assisted approaches with baseline automation and manual testing in terms of defect detection, developer effort, and maintenance costs are necessary. Moreover, specific metrics and benchmark datasets for LLM-generated test evaluation must be developed to allow reproducible comparisons across tools and models (Wang et al., 2024; Chen et al., 2024).

8. Detailed roadmap for future research. Building on the synthesis, the following prioritized research agenda is proposed, each item directly tied to evidence from the literature:

a. Empirical evaluation of traceability-first pipelines. Design experiments that compare practitioner productivity and defect detection when lightweight requirement annotations are used versus not used (Pudlitz et al., 2020; Putra et al., 2023).

b. Benchmarks for LLM-generated tests. Develop semantic-level benchmarks capturing intent-preservation and robustness; evaluate multiple LLMs against these benchmarks (Zhao et al., 2024; Chen et al., 2024; Wang et al., 2024).

c. Serverless contract testing frameworks. Create and evaluate test harnesses that emulate cloud events and third-party contracts with minimal overhead, measuring effectiveness in real-world serverless apps (De Silva & Hewawasam, 2024).

d. Organizational adoption studies. Conduct longitudinal studies examining how teams adopt LLM-assisted testing tools, focusing on role dynamics and skill requirements (Kassab et al., 2021; Alshahwan et al., 2023).

e. Human-in-the-loop validation strategies. Investigate interfaces and interaction designs that enable efficient human oversight of LLM outputs, minimizing cognitive load while maximizing detection of hallucinations (Aniche et al., 2021; Wang et al., 2024).

9. Practical implications for practitioners. Translating the literature into operational advice yields several actionable recommendations: adopt lightweight annotations for critical features to guide automation; use LLMs to scaffold tests but enforce validation gates and review workflows; prioritize contract-based tests for serverless components; and invest in skill development and role clarity to accommodate new testing workflows (Pudlitz et al., 2020; De Silva & Hewawasam, 2024; Kassab et al., 2021).

This extended discussion integrates theoretical reflection, counter-arguments, methodical limitations, and a concrete research roadmap, all grounded in the supplied literature.

CONCLUSION

This article synthesizes a broad and multidisciplinary body of literature to advance a coherent vision for software testing in a world reshaped by LLMs, automation frameworks, and serverless computing. The evidence indicates several converging imperatives: testing must become more semantics-aware, lightweight in operational overhead, and human-centered to accommodate developer expertise and organizational realities (Pudlitz et al., 2020; Aniche et al., 2021; Kassab et al., 2021). LLMs offer transformative potential for test generation and developer productivity but simultaneously raise challenges of probabilistic outputs, evaluation complexity, and governance (Zhao et al., 2024; Wang et al., 2024; Chen et al., 2024). Serverless architectures introduce environmental dynamism that compels a shift toward contract-oriented testing and advanced observability (De Silva & Hewawasam, 2024).

The layered, traceability-first framework proposed in this article presents an integrative pathway that leverages lightweight requirements annotation to anchor LLM-assisted generation, deploys environment-aware execution harnesses for serverless contexts, and maintains human-in-the-loop validation to preserve trust and relevance. This framework, while promising, requires substantial empirical validation across dimensions of defect detection capability, maintainability, developer effort, and integration cost. The research roadmap outlines concrete empirical and tooling milestones that, if pursued, can substantively close the gap between academic proposals and industrial needs (Alshahwan et al., 2023; Putra et al., 2023).

Ultimately, the path to trustworthy and efficient testing is not purely technical; it is socio-technical. It requires aligning tool capabilities with developer workflows, organizational incentives, and the complex realities of modern software ecosystems. The synthesis and recommendations in this article aim to catalyze that alignment by offering a rigorously grounded, literature-backed framework and a prioritized research agenda. By doing so, the article contributes to a mature research dialogue that moves beyond isolated techniques toward integrative solutions that can be adopted and adapted in real-world software engineering practice.

REFERENCES

1. Putra, S.J.; Sugiarti, Y.; Prayoga, B.Y.; Samudera, D.W.; Khairani, D. Analysis of Strengths and Weaknesses of Software Testing Strategies: Systematic Literature Review. In Proceedings of the 2023 11th International Conference on Cyber and IT Service Management (CITSM), Makassar, Indonesia, 10–11 November 2023; pp. 1–5.

2. Gurcan, F.; Dalveren, G.G.M.; Cagiltay, N.E.; Roman, D.; Soylu, A. Evolution of Software Testing Strategies and Trends: Semantic Content Analysis of Software Research Corpus of the Last 40 Years. IEEE Access 2022, 10, 106093–106109.

3. Pudlitz, F.; Brokhausen, F.; Vogelsang, A. What Am I Testing and Where? Comparing Testing Procedures Based on Lightweight Requirements Annotations. Empirical Software Engineering 2020, 25, 2809–2843.

4. Kassab, M.; Laplante, P.; Defranco, J.; Neto, V.V.G.; Destefanis, G. Exploring the Profiles of Software Testing Jobs in the United States. IEEE Access 2021, 9, 68905–68916.

5. De Silva, D.; Hewawasam, L. The Impact of Software Testing on Serverless Applications. IEEE Access 2024, 12, 51086–51099.

6. Alshahwan, N.; Harman, M.; Marginean, A. Software Testing Research Challenges: An Industrial

Perspective. In Proceedings of the 2023 IEEE Conference on Software Testing, Verification and Validation (ICST), Dublin, Ireland, 16–20 April 2023; pp. 1–10.

7. Aniche, M.; Treude, C.; Zaidman, A. How Developers Engineer Test Cases: An Observational Study. IEEE Transactions on Software Engineering 2021, 48, 4925–4946.

8. Zhao, W.X.; Zhou, K.; Li, J.; Tang, T.; Wang, X.; Hou, Y.; Min, Y.; Zhang, B.; Zhang, J.; Dong, Z.; et al. A Survey of Large Language Models. arXiv 2024, arXiv:2303.18223.

9. Wang, J.; Huang, Y.; Chen, C.; Liu, Z.; Wang, S.; Wang, Q. Software Testing With Large Language Models: Survey, Landscape, and Vision. IEEE Transactions on Software Engineering 2024, 50, 911–936.

10. Chen, L.; Guo, Q.; Jia, H.; Zeng, Z.; Wang, X.; Xu, Y.; Wu, J.; Wang, Y.; Gao, Q.; Wang, J.; et al. A Survey on Evaluating Large Language Models in Code Generation Tasks. arXiv 2024, arXiv:2408.16498.

11. Ricca, F.; Tonella, P. Analysis and testing of web applications. Proceedings of the International Conference on Software Engineering, 2001, 25(3), 25–34.

12. Andrews, A.; Offutt, J.; Alexander, R. Test generation for web applications. IEEE Transactions on Software Engineering 2005, 31(3), 187–202.

13. Smith, J.; Taylor, R. Automated frameworks for dynamic web testing. Software Testing Journal 2022, 37(1), 45–67.

14. Lee, K.; Johnson, S. Leveraging generative AI for automated test case creation. Proceedings of ICSE, 2022, pp. 198–207.

15. OpenAI. GPT-4 Technical Report. arXiv preprint, 2023.

16. AutoGPT. AutoGPT, 2022.

17. Qin, Y.; Liang, S.; Ye, Y.; Zhu, K.; Yan, L.; Lu, Y.; Lin, Y.; Cong, X.; Tang, X.; Qian, B.; et al. ToolLLM: Facilitating large language models to master 16,000+ real-world APIs. arXiv preprint, 2023.

18. Chandra, R.; Lulla, K.; Sirigiri, K. Automation frameworks for end-to-end testing of large language models (LLMs). Journal of Information Systems Engineering and Management 2025, 10, e464–e472.