International Science Journal of Engineering & Agriculture
2026; 5(2): 21-27
https://isg-journal.com/isjea
doi: 10.46299/j.isjea.20260502.03
ISSN: 2720-6319

INTERNATIONAL
SCIENCE GROUP

AI agents in software testing: a human-in-the-loop assurance model

Serhiy Kopovskyi
Lviv Polytechnic National University, Lviv, Ukraine
ORCID: 0009-0001-8969-2925

Abstract: AI agents are increasingly integrated into software testing workflows, including test case generation, regression prioritization, execution analysis, and defect triage. While these capabilities can improve throughput and expand exploratory coverage, their adoption in release-relevant contexts remains methodologically undergoverned. Current studies predominantly evaluate task-level utility of model outputs, but provide limited guidance on how to define delegation boundaries, approval authority, evidence requirements, and escalation rules when AI-generated artifacts may affect merge or release decisions. The aim of this study is to develop a governance-oriented framework for the controlled use of AI agents in software testing. The research applies a design-oriented conceptual methodology that synthesizes evidence from software engineering, software testing, and trustworthy artificial intelligence governance literature. As a result, two linked methodological artifacts are proposed: a human-in-the-loop assurance model and an activity-level risk/control matrix. The model distinguishes assistive, supervised, and conditional autonomous modes, while the matrix relates testing activities, artifact criticality, and autonomy level to required human controls, traceability obligations, and escalation triggers. Risk is operationalized through a heuristic composite formulation ($S = C + I + U + L + V$), used to calibrate governance intensity rather than to support statistical prediction. The practical value of the study lies in providing a structured baseline for integrating AI-agent-supported testing into continuous integration and continuous delivery workflows. The main limitation is the absence of empirical cross-context validation of the proposed governance parameters.

Keywords: artificial intelligence agents, software testing, human-in-the-loop supervision, risk governance, quality assurance, continuous integration and continuous delivery, trustworthy artificial intelligence.

1. Introduction

Software testing is being reconfigured by agent-assisted workflows in activities such as test case generation, prioritization, oracle suggestion, and defect triage. In contemporary development pipelines, these capabilities can improve the speed and coverage of testing tasks that were previously constrained by manual effort.

Despite these opportunities, current adoption practices often emphasize task-level performance of agent outputs while insufficiently specifying human oversight and governance constraints. When artificial-intelligence-generated artifacts influence release-critical decisions, unclear responsibility boundaries can lead to inconsistent quality control and weak accountability.

The central problem addressed in this paper is therefore not whether AI agents can support testing, but how they can be integrated in a controlled and auditable manner. This challenge is especially relevant for teams seeking practical adoption without introducing avoidable organizational or technical risk.

This study contributes two methodological artifacts for governance-aware integration: (i) a human-in-the-loop integration model that specifies delegation boundaries across testing activities, and (ii) a risk/control matrix that links artifact criticality and autonomy level to mandatory oversight actions and escalation triggers.

2. Object and subject of research

The object of this study is AI-agent-supported software testing workflows in contemporary software delivery environments, particularly in situations where generated or prioritized testing artifacts may influence merge and release decisions.
The subject of the study is the governance mechanism required for such workflows, including delegation boundaries, human approval requirements, evidence obligations, and escalation rules. Analysis of current practice and literature indicates three persistent deficiencies: an autonomy-to-authority mismatch, missing artifact-level control logic, and weak escalation formalization.

3. Aim and objectives of research

The aim of this research is to develop a human-in-the-loop assurance model and an activity-level risk/control matrix that formalize delegation boundaries, evidence requirements, and escalation rules for AI-agent-supported software testing workflows.
To achieve this aim, the following research tasks were defined:

- identify the principal risk categories associated with AI-agent-generated or AI-agent-modified testing artifacts;
- map risk level and autonomy level to concrete human control requirements within testing workflows;
- determine how such governance logic can be integrated into practical delivery pipelines without removing the efficiency gains of AI assistance.
- Literature analysis

This section synthesizes evidence from peer-reviewed and preprint studies on large-language-model-enabled software engineering and software testing (2021–2024), together with governance standards used in safety- and compliance-sensitive artificial intelligence deployment contexts. The synthesis is organized into four clusters to separate capability evidence from assurance evidence. This distinction clarifies the governance gap addressed below.
The first cluster covers general software engineering use cases. Benchmark-oriented studies on code-focused language models and repository-level issue resolution report improvements on tasks such as code completion and patch drafting, while the same evidence base documents variability, context sensitivity, and non-trivial failure rates in realistic settings [1, 2, 3]. This cluster establishes capability potential, not governance sufficiency.

The second cluster covers testing-specific applications. Published studies, including a peer-reviewed empirical evaluation in IEEE Transactions on Software Engineering, report use of large language models for unit-test generation, assertion suggestion, and test-suite expansion, with utility in repetitive testing tasks and baseline scaffolding [4, 5, 14]. Broader survey evidence on software testing practice also indicates that industrial testing decisions remain strongly conditioned by process discipline, context, and resource constraints [15]. At the same time, the testing-focused studies identify limits directly relevant to release governance: prompt-sensitive output quality, brittle assertions, and weak provenance links between generated artifacts and domain acceptance criteria [4, 5, 14].

The third cluster addresses agentic workflows, where models execute multi-step plans across tools rather than returning a single response. Evidence from agent frameworks and evaluation benchmarks shows that tool-using agents increase automation scope, but also increase coordination and error-propagation risk across pipeline stages [6, 7, 8]. These findings motivate explicit control handoffs when agent output can affect release decisions.
The fourth cluster covers trustworthy artificial intelligence governance standards. NIST AI RMF and its generative-AI profile, together with ISO/IEC management and risk guidance, define transparency, accountability, risk-tiering, and documentation expectations for consequential artificial intelligence

use [9, 10, 11, 13]. Yet these frameworks are not testing-activity-specific and do not define who approves which artifact at each quality-assurance gate.

Across the four clusters, the key methodological gap is the absence of an operational mechanism that jointly maps (i) autonomy level, (ii) testing activity, and (iii) artifact risk to concrete human controls, required evidence, and escalation rules. The subsequent sections formalize this gap into a structured human-in-the-loop model and a risk/control matrix.

Table 1. Analytical synthesis of empirical capability evidence and governance coverage gaps.

Study focus	Testing phase	Agent role	Human oversight type	Governance coverage	Limitation
Code-generation and repository benchmarks [1, 3]	Design, maintenance	Assistive	Practitioner review	Limited	Task success varies by context complexity
Testing-focused survey and unit-test evaluation [4, 5, 14]	Test design/execution	Assistive recommender	Reviewer confirmation	Partial	Prompt sensitivity and weak artifact provenance
Agent frameworks and agent benchmarks [6, 7]	Multi-phase orchestration	Semiautonomous	Role-based supervision	Partial	Error propagation across tool chains
NIST/ISO governance standards [9, 11, 13]	Cross-domain governance	Policy baseline	Human-in-command principle	Strong at policy level	Not operationalized for quality-assurance gate decisions

The synthesis in Table 1 highlights a consistent mismatch between capability-oriented evidence and testing-specific governance operationalization.

5. Research methods

This article uses a design-oriented conceptual methodology in which the primary research artifact is a governance instrument rather than a predictive model.

The methodological protocol includes three explicit stages.

Stage 1: Source selection and delimitation. We prioritize peer-reviewed evidence on large-language-model-enabled software engineering and testing, using preprints as supplementary sources where peer-reviewed coverage remains limited. Normative governance and software-testing process standards are used for construct definition [1, 3, 4, 9, 11, 12].

Stage 2: Construct coding and synthesis. Each source is coded by testing activity, agent role, autonomy mode, artifact type, uncertainty signal, control mechanism, and accountability requirement. These constructs are grouped as context, autonomy, and control dimensions to support consistent

comparison. The coding output is then normalized into a common vocabulary used in the model and matrix.

Stage 3: Artifact specification and analytical consistency checking. Decision rules, approval gates, and escalation conditions are specified and then checked analytically across scenarios of test generation, defect triage, and regression gating. Assessment criteria are rule completeness, role clarity, and audit-trace sufficiency.

5.1 Operationalization procedure for quality-assurance-gate integration

To support practical adoption without changing the conceptual scope of the study, the proposed artifacts are operationalized through a structured implementation sequence. First, organizations define the testing activities where AI-generated artifacts may influence merge or release outcomes. Second, each activity is mapped to an autonomy mode (assistive, supervised, or conditional autonomous) and to the responsible human authority.

Third, artifact classes are scored using the five-factor rubric shown in equation (1), and the resulting risk tier is linked to mandatory controls: review depth, approval role, evidence logging, and reproducibility checks. Fourth, escalation triggers are encoded at quality-assurance gates, including low-confidence outputs, conflicting evidence, reproducibility failure, and policy-boundary violations.

Fifth, audit traceability is ensured by linking each decision to a minimal evidence package comprising prompt/output trace, review rationale, and override record where applicable. This procedure does not introduce empirical performance claims; it provides a replicable governance workflow for applying the conceptual model and matrix in operational continuous integration and continuous delivery contexts.

Risk is operationalized as follows:

$$S = C + I + U + L + V, \tag{1}$$

where C denotes artifact criticality, I expected failure impact, U output uncertainty, L compliance sensitivity, and V environmental volatility. Each component is rated on an ordinal three-point scale. The resulting score functions as an ordinal governance rubric for control assignment rather than as a statistically validated quantitative risk model.

6. Research results

The main results of the study are presented through the proposed human-in-the-loop integration model, the risk and control matrix, and the interpretation of their practical application conditions.

6.1 Proposed human-in-the-loop integration model

Using the constructs defined in Stage 2, the proposed model contains three coordinated layers: agent execution layer, human supervision layer, and governance layer. The agent layer performs bounded tasks such as generating candidate tests or ranking defect reports. The supervision layer validates outputs, resolves ambiguity, and applies contextual judgment. The governance layer defines policy constraints, approval responsibilities, and accountability records.

Integration is specified across core testing activities: test design, test generation, execution interpretation, defect triage, and regression maintenance. For each activity, the model defines acceptable agent actions and non-delegable human decisions.

The model distinguishes three autonomy modes: assistive mode (human decides), supervised mode (agent proposes and humans approve), and conditional autonomous mode (agent acts within policy constraints and triggers mandatory review under predefined conditions). This structure avoids binary automation assumptions.

Escalation logic is formalized through four policy rules. R1. If artifact risk class is high, autonomy may not exceed supervised mode. R2. Any artifact that affects merge or release gates requires explicit human approval and a linked evidence record. R3. Conditional autonomous execution is allowed only when policy constraints, tool provenance, and rollback path are pre-validated. R4. Escalation is mandatory when confidence is below threshold, conflicting evidence is detected, or output reproducibility fails.

6.2 Risk and control matrix for AI-agent-supported testing activities

To operationalize governance decisions in a transparent and auditable manner, risk is represented as the heuristic composite score in equation (1). The resulting score is mapped to operational control tiers: low, medium, and high.

Control actions include mandatory review depth, approval authority, evidence logging, reproducibility checks, and escalation routing. This aligns testing governance with risk-tiered principles used in current artificial intelligence assurance guidance while translating them to quality-assurance-gate decisions [9, 10, 13].

Practical integration occurs at pre-merge, pre-release, and post-release checkpoints, mapped to actions such as test selection, release approval, and incident-driven reassessment. The matrix operationalizes governance decisions at these delivery-workflow checkpoints.

Table 2. Risk and control matrix for AI-agent-supported testing.

Testing activity	AI output type	Risk class	Autonomy level	Mandatory human control	Required evidence	Escalation rule
Test design	Candidate test scenarios	Medium	Supervised	Senior quality-assurance review and approval	Prompt/output trace and rationale note	Escalate if coverage rationale is insufficient for critical paths
Regression selection	Prioritized suite	Medium	Conditional autonomous	Test lead confirmation for release branches	Selection log with risk tags	Escalate if critical paths are excluded
Defect triage	Severity recommendation	High	Supervised	Dual review (quality assurance and developer)	Decision record and conflict annotation	Escalate on disagreement or low confidence
Assertion generation	Auto-generated checks	High	Assistive or supervised	Manual validation before merge	Reproducible execution evidence	Escalate if flaky or non-deterministic
Post-release monitoring	Failure clustering	Medium	Conditional autonomous	Periodic audit by release authority	Audit log and rollback readiness	Escalate on a novel failure pattern

6.3 Discussion of results

The literature synthesis indicates that the practical value of AI agents in software testing is strongest in high-volume, pattern-intensive tasks, including initial test generation and regression candidate selection. In these settings, agent support can accelerate baseline artifact production and broaden exploratory coverage, but the resulting outputs remain sensitive to context and input formulation, which limits their direct use in release-critical decisions [4, 5, 14].

At the same time, the evidence base does not support full delegation of decisions that carry direct accountability consequences. Results from repository-scale and agent-oriented benchmarks suggest that performance degrades as task coupling, contextual depth, and cross-tool dependencies increase [3, 7]. This is particularly relevant for release gating, severity assignment, and compliance-relevant judgment, where decision quality depends on domain interpretation and organizational responsibility rather than artifact generation alone.

The proposed framework therefore treats governance overhead as a calibrated design choice rather than a uniform burden. By linking control intensity to artifact risk and autonomy level, the model preserves efficiency for lower-impact tasks while imposing stricter review and evidence requirements for high-impact outputs. This position is consistent with risk-tiered governance principles in contemporary artificial intelligence assurance guidance [9, 13], but adapts those principles to testing-specific workflow checkpoints.

A key boundary condition is that the present contribution is methodological and non-empirical. It provides an operational governance structure for decision rights, evidence obligations, and escalation logic, but it does not claim validated performance effects. Empirical evaluation remains necessary to estimate impacts on review latency, escaped defects, and governance consistency under real deployment conditions.

7. Prospects for further research development

Future research should prioritize three empirical evaluation tracks. First, an industrial pilot should compare governance and non-governance testing workflows using escaped-defect and review-latency indicators. Second, cross-domain replication should evaluate the transferability of risk thresholds and control assignments. Third, policy-as-code implementation should be assessed for enforcement reliability and override behavior in continuous integration and continuous delivery pipelines.

A secondary line of work should examine multi-agent coordination risk and governance drift over time.

8. Conclusions

This paper develops a governance-oriented contribution for AI-agent-supported software testing by specifying two linked artifacts: a human-in-the-loop integration model and an activity-level risk/control matrix. Together, these artifacts translate broad assurance principles into operational decisions about delegation boundaries, approval authority, evidence requirements, and escalation triggers across testing workflows.

The main theoretical contribution is an explicit mapping between autonomy mode, testing activity, and artifact risk at quality-assurance-gate granularity. The main practical contribution is a deployable governance baseline for continuous integration and continuous delivery contexts in which AI-generated outputs may influence merge and release outcomes. In this sense, the study extends capability-focused testing literature by formalizing governance constraints rather than proposing a new model architecture.

The principal limitation is external validation. The framework is intentionally non-empirical and has not yet been evaluated across heterogeneous organizational environments. Future research should therefore test calibration robustness, workflow overhead, and quality outcomes in real delivery settings, including policy-as-code implementations and cross-domain replication.

References:

1) Chen, M., Tworek, J., Jun, H., Yuan, Q., Ponde de Oliveira Pinto, H., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., ... Zaremba, W. (2021). Evaluating large language models trained on code. arXiv. https://arxiv.org/abs/2107.03374

2) OpenAI. (2023). GPT-4 technical report. arXiv. https://arxiv.org/abs/2303.08774

3) Jimenez, C. E., Yang, J., Wettig, A., Yao, S., Pei, K., Press, O., & Narasimhan, K. (2024). SWE-bench: Can language models resolve real-world GitHub issues? In The Twelfth International Conference on Learning Representations (ICLR). https://proceedings.iclr.cc/paper_files/paper/2024/file/edac78c3e300629acfe6cbe9ca88fb84-Paper-Conference.pdf

4) Wang, J., Huang, Y., Chen, C., Liu, Z., Wang, S., & Wang, Q. (2024). Software testing with large language models: Survey, landscape, and vision. IEEE Transactions on Software Engineering, 50(4), 911–936. https://doi.org/10.1109/TSE.2024.3368208

5) Yang, L., Yang, C., Gao, S., Wang, W., Wang, B., Zhu, Q., Chu, X., Zhou, J., Liang, G., Wang, Q., & Chen, J. (2024). On the evaluation of large language models in unit test generation. In Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering (ASE '24) (pp. 1607–1619). Association for Computing Machinery. https://doi.org/10.1145/3691620.3695529

6) Wu, Q., Bansal, G., Zhang, J., Wu, Y., Li, B., Zhu, E., Jiang, L., Zhang, X., Zhang, S., Liu, J., Awadallah, A. H., White, R. W., Burger, D., & Wang, C. (2024). AutoGen: Enabling next-gen LLM applications via multi-agent conversation. In First Conference on Language Modeling (COLM). OpenReview. https://openreview.net/forum?id=BAakY1hNKS

7) Liu, X., Yu, H., Zhang, H., Xu, Y., Lei, X., Lai, H., Gu, Y., Ding, H., Men, K., Yang, K., Zhang, S., Deng, X., Zeng, A., Du, Z., Zhang, C., Shen, S., Zhang, T., Su, Y., Sun, H., ... Tang, J. (2024). AgentBench: Evaluating LLMs as agents. In The Twelfth International Conference on Learning Representations (ICLR). https://proceedings.iclr.cc/paper_files/paper/2024/hash/e9df36b21ff4ee211a8b71ee8b7e9f57-Abstract-Conference.html

8) Xi, Z., Chen, W., Guo, X., He, W., Ding, Y., Hong, B., Zhang, M., Wang, J., Jin, S., Zhou, E., Zheng, R., Fan, X., Wang, X., Xiong, L., Zhou, Y., Wang, W., Jiang, C., Zou, Y., Liu, X., ... Gui, T. (2025). The rise and potential of large language model based agents: A survey. Science China Information Sciences, 68(2), 121101. https://doi.org/10.1007/s11432-024-4222-0

9) Tabassi, E. (2023). Artificial intelligence risk management framework (AI RMF 1.0) (NIST AI 100-1). National Institute of Standards and Technology.

10) Autio, C., et al. (2024). Artificial intelligence risk management framework: Generative AI profile (NIST AI 600-1). National Institute of Standards and Technology.

11) International Organization for Standardization. (2023). ISO/IEC 42001:2023: Artificial intelligence—Management system. https://www.iso.org/standard/81230.html

12) International Organization for Standardization. (2021). ISO/IEC/IEEE 29119-2:2021: Software and systems engineering—Software testing—Part 2: Test processes. https://www.iso.org/standard/79428.html

13) International Organization for Standardization. (2023). ISO/IEC 23894:2023: Artificial intelligence—Guidance on risk management. https://www.iso.org/standard/77304.html

14) Schäfer, M., Nadi, S., Eghbali, A., & Tip, F. (2024). An empirical evaluation of using large language models for automated unit test generation. IEEE Transactions on Software Engineering, 50(1), 85–105. https://doi.org/10.1109/TSE.2023.3334955

15) Garousi, V., & Zhi, J. (2013). A survey of software testing practices in Canada. Journal of Systems and Software, 86(5), 1354–1376. https://doi.org/10.1016/j.jss.2012.12.051