Accountability in Agentic Secure Software Development Life Cycles: Challenges, Degradation Scenarios, and Governance Principles

Rubens Abraão da Silva Sousa
State University of Ceará
Fortaleza, Ceará, Brazil
abraao.sousa@aluno.uece.br

Gabriel Pinheiro Rezende de Lima
State University of Ceará
Fortaleza, Ceará, Brazil
gabriel.rezende@aluno.uece.br

Igor Ramsés Temóteo dos Santos
State University of Ceará
Fortaleza, Ceará, Brazil
igor.ramses@aluno.uece.br

Sávio Freire
Federal Institute of Ceará
Morada Nova, Ceará, Brazil
State University of Ceará
Fortaleza, Ceará, Brazil
savio.freire@uece.br

Ismayle de Sousa Santos
State University of Ceará
Fortaleza, Ceará, Brazil
ismayle.santos@uece.br

Paulo Henrique Mendes Maia
State University of Ceará
Fortaleza, Ceará, Brazil
pauloh.maia@uece.br

Abstract

Secure software development frameworks, such as OWASP SAMM, BSIMM, and NIST SSDF, structure security decisions on the premise that an identifiable professional approves, audits, and is accountable for them. LLM-based agents disrupt this premise: in threat modeling, penetration testing, and incident response, they not only produce evidence but also propose strategies, prioritize actions, and execute workflows with varying autonomy. Formal responsibility remains human, while the construction of the decision is distributed between humans and agents. This position paper characterizes this accountability decoupling and its resulting phenomenon, the Accountability Chasm: the gap between the operational autonomy of agents and the capacity of governance mechanisms to preserve traceability, contestability, and attribution over security decisions. Unlike the classic accountability gap, here, accountability remains attributed to a human; what degrades is what makes this attribution effective. This paper describes three degradation scenarios in real-world security activities, propose four accountability-oriented governance principles, and presents a research agenda.

Keywords

Accountability, Agentic AI, AI Agents, Secure Software Development Lifecycle, Software Security, AI Governance, Human-Agent Collaboration

1 Introduction

Secure Software Development Life Cycle (SSDLC) practices seek to incorporate security activities throughout the entire software development life cycle, including threat modeling, code review, security testing, and incident response [9]. Security-oriented frameworks, such as the Secure Software Development Framework (NIST SSDF) [12], the OWASP Software Assurance Maturity Model (OWASP SAMM) [13], and the Building Security In Maturity Model (BSIMM) [11] aim to structure activities so that decisions related to risks, vulnerabilities, and security controls can be analyzed, justified, and audited by professionals.

Recent advances in Large Language Model (LLM)-based systems have spurred the emergence of agents capable of planning, executing, and coordinating activities previously carried out by software teams [5, 6]. The use of these agents is increasingly common at different stages of the software development cycle, while in the security domain their application has already been explored in threat modeling, vulnerability triage, penetration testing, and incident response operations [2, 3, 15]. Agents have ceased to act merely as support tools and now operate as direct participants in the production of analyses, recommendations, and actions that influence security decisions [5].

Existing discussions focus on productivity, autonomy, reliability, and human–agent collaboration, but provide only limited treatment of accountability [5, 6]. Nevertheless, this accountability has not been duly characterized. Aleti et al. [1] treat accountability as a system property to be designed; Hoda [8], on the other hand, situate it among open sociotechnical concerns, yet neither describes the mechanism of its degradation. Thus, **how does accountability for security decisions hold when execution is delegated to agents?** Formal accountability remains with professionals and organizations, yet an increasing share of the analyses that underpin it is produced by agents at different levels of autonomy, as shown in Figure 1[7, 14]. Figure 1 illustrates the SSDLC activities in which LLM-based agents can participate, highlighting that agentic support may span the entire security lifecycle rather than isolated development tasks.

The principal challenge does not stem from autonomy itself, but from the misalignment between the operational autonomy delegated to agents and the governance mechanisms available to assign, contest, and audit security decisions. As agents assume SSDLC responsibilities, execution and accountability cease to evolve in a coupled manner. Consequently, it becomes more difficult to understand who influenced a given decision, who could contest it, and who must answer for its direct and indirect impacts [4, 7, 10].

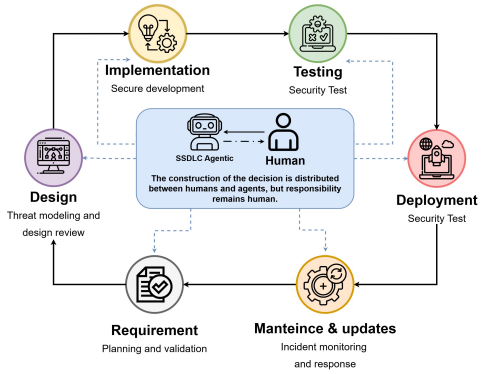

Figure 1: SSDLC cicle

This phenomenon is named in this work as **Accountability Chasm**, i.e., a progressive gap between the operational autonomy of agents in security activities and the organizational capacity to maintain adequate accountability mechanisms over security decisions. This paper argues that this gap emerges from the decoupling between execution and accountability, giving rise to challenges related to traceability, contestability, and attribution throughout the SSDLC.

This paper presents three contributions: **i)** the characterization of responsibility decoupling and the resulting Accountability Chasm, distinguishing them from the responsibility gap and the control gap described in the literature; **ii)** the description of three degradation scenarios, loss of contestability, degradation of traceability, and diffusion of responsibility, in real threat modeling, penetration testing, and incident response activities; and **iii)** the proposal of four governance principles oriented toward accountability, each associated with a verifiable mechanism, and a research agenda to investigate them.

2 Existing SSDLC Frameworks Were Not Designed for Agentic Security Decisions

SSDLC frameworks were conceived to integrate security into software engineering activities through structured governance, verification, and risk management practices. Models such as OWASP SAMM [13], BSIMM [11], and NIST SSDF [12] define processes for threat identification, code review, security testing, vulnerability management, and incident response. Although they differ in scope and operationalization, these frameworks converge on one assumption: security decisions remain under direct human supervision.

This assumption appears in the way the frameworks treat risk acceptance, change approval, security evidence validation, and vulnerability handling activities. Each of these activities is structured as a practice with designated owners, review points, and audit trails [12, 13], which justify the decisions made throughout the cycle. The objective is not merely to execute security tasks, but **to ensure that these decisions can be attributed, contested, and audited afterward.**

The architecture of these frameworks [11–13] emerged in a context in which automated tools performed predominantly instrumental functions. Static analyzers, vulnerability scanners, and monitoring platforms produced evidence to support human decisions, but rarely participated in the formulation of these decisions. Responsibility for interpreting results, prioritizing risks, and defining corrective actions remained associated with the professionals and teams responsible for system security.

The emergence of LLM-based agents changes this dynamic. Bandara et al. [3] apply agents with vision-language models to threat modeling, Shen et al. [15] automate penetration testing with LLM agents, and Ali et al. [2] employ agents in incident response. Unlike traditional tools, these agents not only produce evidence, but also propose strategies, select alternatives, prioritize actions, and execute workflows at different levels of autonomy. Thus, agents directly influence security decisions previously associated with human deliberation.

This change exposes a structural limitation of current frameworks. Their governance mechanisms were designed for environments in which operational execution and accountability remained coupled. When agents participate in the production and operationalization of security decisions, this coupling breaks: the origin of a recommendation, the possibility of contesting it, and the attribution of who answers for its effects cease to be direct. The frameworks continue to guide security management, but offer limited support for decisions produced under increasing agentic autonomy.

The resulting challenge does not stem from the replacement of SSDLC practices, but from the transformation of the assumptions upon which they were built. The introduction of agents into security activities creates a progressive separation between execution and responsibility. This separation is a condition not yet characterized in the SSDLC context and constitutes a starting point for the **Accountability Chasm** phenomenon.

3 From Responsibility Decoupling to the Accountability Chasm

The introduction of agents into security activities does not eliminate the accountability mechanisms existing in SSDLCs. The challenge emerges when operational autonomy grows faster than organizational capacity to comprehend, supervise, and audit the decisions produced in these activities. Under this condition, formal responsibility remains assigned to individuals or teams, while the actual production of analyses, recommendations, and actions becomes shared with autonomous agents.

This process results in **responsibility decoupling**. In traditional environments, those responsible for approving a decision participate directly in evidence collection, alternative evaluation, and justification of the final outcome. In agentic environments, some of these steps may be executed by agents capable of generating analyses, selecting strategies, and recommending actions without continuous human involvement. **Responsibility remains human**, but decision construction becomes progressively distributed.

This decoupling differs from the classic responsibility gap. Matthias [10] describes situations in which no one can be held accountable because system behavior is unpredictable, and Hindriks and Veluwenkamp [7] reposition the problem as a control gap. In the

case the authors examine, formal responsibility remains assigned to a human; what degrades is the traceability and contestability that make this assignment effective. The presence of an accountable party in the flow does not suffice when they cannot observe or reconstruct the process that led to the agent's recommendation [4, 7].

The first effect of the decoupling falls on contestability. To contest a decision depends on understanding which information was used, which criteria were considered, and which alternatives were discarded. In penetration tests conducted by agents [15], the professional who approves the final report may not have access to the chain that led the agent to prioritize one attack vector over another. Human supervision alone does not resolve this problem as long as the agent's decision-making process remains insufficiently observable.

The second effect falls on traceability. Security decisions result from multiple interactions among artifacts, tools, teams, and organizational processes. In agentic SSDLCs, the decision chain comes to include agents that produce intermediate artifacts, execute tasks, and influence prioritizations. In threat modeling generated by agents [3], reconstructing which premises led to a specific threat becomes difficult when multiple agents chain steps in the same workflow.

The combination of loss of contestability and degradation of traceability favors the diffusion of responsibility. In agent-assisted incident response [2], when triage and recommended actions are produced by the agent, it becomes less clear who should have identified an omitted indicator and who answers for its consequences. The problem ceases to reflect the isolated behavior of an agent and begins to expose structural limitations of governance mechanisms that supervise decisions distributed between humans and agents.

Accountability Chasm results from this process. The term describes situations in which the operational autonomy of agents exceeds the capacity of governance mechanisms to preserve traceability, contestability, and attribution over security decisions. The Accountability Chasm is not a specific technological failure: it is a sociotechnical misalignment between the levels of autonomy exercised by agents and the organizational mechanisms necessary to sustain accountability in agentic SSDLCs [14].

Rather than operating as independent phenomena, the relationship between these concepts forms a causal chain, as summarized in Figure 2. As operational autonomy increases, responsibility becomes progressively decoupled from decision-making. This shift degrades contestability and traceability, diffuses responsibility, and ultimately resultates in the Accountability Chasm.

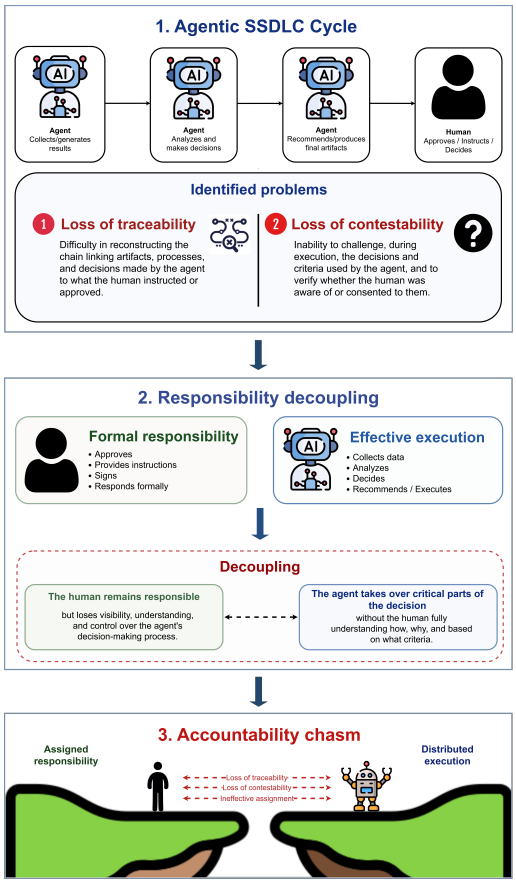

Figure 2: Causal Chain

4 Reimagining Accountability for Agentic SSDLCs

The Accountability Chasm indicates that traditional governance mechanisms are insufficient to sustain accountability under increasing levels of agentic autonomy. The answer does not lie in restricting the use of agents in security activities, but in adapting accountability mechanisms to the new forms of collaboration between humans and agents. In this direction, accountability should be treated as an architectural property of agentic SSDLCs, designed into the workflow rather than imposed as an obligation after execution. Each of the three following principles addresses one of the degradation effects described in Section 3. s The first principle is accountability by design. Instead of reconstructing accountability after an incident, every point in the flow at which an agent produces or influences a decision is associated, during system design, a named human responsible party and a mandatory record. Verification is direct: before deployment, auditing verifies whether every agent-touched decision point has a designated approver.

The second principle demands verifiable attribution and responds to the diffusion of responsibility. Every decision records not only its outcome but the provenance of its construction: which agent produced each recommendation, which evidence was used, and which human assumed the final decision. Agent identifiers and provenance trails [4] make this operational; the test is the ability to recover, from any decision, the producing agent and the responsible human.

The third principle determines that automation must not replace contestability. The capacity of agents to propose strategies, prioritize vulnerabilities, and execute actions renders human supervision insufficient by itself [7]. Before a recommendation takes effect, a responsible party must be able to inspect its inputs and criteria and reject it, with the rationale recorded—an explicit contestation point in the flow, not tacit approval.

The fourth principle demands continuous traceability throughout the human–agent flow. An append-only record captures prompts,

intermediate artifacts, actions, and approvals, so that any decision can be reconstructed afterward [4]. This evidence chain is a condition for auditing and accountability and, when multiple agents act in the same flow, prevents the origin of a decision from being lost.

These principles do not eliminate the challenges of using agents in SSDLCs. Consolidating them is an open research opportunity at the intersection of security and agentic software engineering.

5 Research Agenda

Characterizing accountability in agentic SSDLCs requires empirical and conceptual investigation into how its mechanisms can be preserved in human-agent workflows. Four research directions emerge.

RQ1 — How to measure responsibility decoupling in agentic SSDLC activities? The question calls for a metric of misalignment between execution, decision, and accountability. The provenance records from Section 4 provide the substrate: the fraction of decisions whose origin is reconstructible and the proportion of agent recommendations with a named approver are candidate indicators, to be validated in activities such as threat modeling, code review, and incident response. A key challenge is distinguishing agent influence from human deliberation when their contributions are intertwined.

RQ2 — What granularity of traceability is sufficient to audit security decisions in human agent flows, and at what cost? Section 4 proposes recording prompts, artifacts, recommendations, actions, and approvals; it remains to determine which level of detail permits reconstructing a decision without prohibitive storage cost or exposure of sensitive data [4]. Studies may compare logging levels with respect to their ability to support a real audit.

Determining sufficiency poses a challenge, as there is no consensual definition of a successful audit in agentic SSDLCs, which hinders the comparative evaluation of logging strategies against a ground truth.

RQ3 — How to ensure contestability in security decisions influenced by agents? Empirical studies should examine which information security professionals need to understand, question, reject, or revise agent recommendations before they affect the system, as well as how such contestation can be incorporated into operational workflows.

RQ4 — Under which conditions does each governance mechanism reduce the Accountability Chasm? Rather than assuming that the principles from Section 4 apply uniformly, this line empirically evaluates when mandatory human approval, autonomy limits, segregation of duties, and peer review are necessary or sufficient varying the agent's level of autonomy and the type of activity and their accountability and operational fluidity trade-offs.

RQ5 — How does agentic automation bias affect the effective exercise of contestability in security decisions?

This demands empirical investigation into how professionals actually interact with recommendations produced by agents in security activities: how they evaluate, reject, or accept these recommendations and which factors, the confidence communicated by the recommendation itself, the cumulative approval load, and the agent's accuracy history, predict the shift from genuine contestation to tacit approval.

These questions shift the debate from a generic question about trust in agents to a specific agenda about security decisions. The challenge is not to decide whether agents can participate in the SSDLC, but to establish under which conditions their actions remain attributable, contestable, and auditable.

Artifact Availability

No artifacts were produced for release

Acknowledgments

This work was developed with support from the National Council for Scientific and Technological Development - CNPq (Process 312173/2025-3), Ceará State Foundation for the Support of Scientific and Technological Development (FUNCAP) (Process No. BMD-0008-03099.01.07/26) and This study was financed in part by the Brazilian Federal Agency for Support and Evaluation of Graduate Education – CAPES – Finance Code 001

References

[1] Aldeida Aleti, Baishakhi Ray, Rashina Hoda, and Simin Chen. 2026. Trustworthy AI Software Engineers. arXiv:2602.06310 [cs.SE] https://arxiv.org/abs/2602.06310

[2] Gauhar Ali, Sajid Shah, and Mohammed ElAffendi. 2025. Enhancing cybersecurity incident response: AI-driven optimization for strengthened advanced persistent threat detection. *Results in Engineering* 25 (3 2025), 104078. doi:10.1016/j.rineng.2025.104078

[3] Eranga Bandara, Amin Hass, Sachin Shetty, Ravi Mukkamala, Ross Gore, Abdul Rahman, and Safdar H. Bouk. 2025. Deep-Stride: Automated Security Threat Modeling with Vision-Language Models. In *2025 International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*. IEEE, 1–7. doi:10.23919/SoftCOM66362.2025.11197424

[4] Alan Chan, Carson Ezell, Max Kaufmann, Kevin Wei, Lewis Hammond, Herbie Bradley, Emma Bluemke, Nitarshan Rajkumar, David Krueger, Noam Kolt, Lennart Heim, and Markus Anderljung. 2024. Visibility into AI Agents. In *The 2024 ACM Conference on Fairness, Accountability, and Transparency*. ACM, Rio de Janeiro Brazil, 958–973. doi:10.1145/3630106.3658948

[5] Ahmed E. Hassan, Hao Li, Dayi Lin, Bram Adams, Tse-Hsun Chen, Yutaro Kashiwa, and Dong Qiu. 2025. Agentic Software Engineering: Foundational Pillars and a Research Roadmap. (9 2025). http://arxiv.org/abs/2509.06216

[6] Junda He, Christoph Treude, and David Lo. 2025. LLM-Based Multi-Agent Systems for Software Engineering: Literature Review, Vision, and the Road Ahead. *ACM Transactions on Software Engineering and Methodology* 34 (6 2025), 1–30. Issue 5. doi:10.1145/3712003

[7] Frank Hindriks and Herman Veluwenkamp. 2023. The risks of autonomous machines: from responsibility gaps to control gaps. *Synthese* 201, 1 (Jan. 2023), 21. doi:10.1007/s11229-022-04001-5

[8] Rashina Hoda. 2026. Toward Agentic Software Engineering Beyond Code: Framing Vision, Values, and Vocabulary. (2 2026). http://arxiv.org/abs/2510.19692

[9] Rafiq Ahmad Khan, Siffat Ullah Khan, Muhammad Azeem Akbar, and Musaad Alzahrani. 2024. Security risks of global software development life cycle: Industry practitioner's perspective. *Journal of Software: Evolution and Process* 36 (3 2024). Issue 3. doi:10.1002/smr.2521

[10] Andreas Matthias. 2004. The responsibility gap: Ascribing responsibility for the actions of learning automata. *Ethics and Information Technology* 6, 3 (2004), 175–183. doi:10.1007/s10676-004-3422-1

[11] Gary McGraw and Brian Chess. 2009. The Building Security in Maturity Model (BSIMM). USENIX Association, Montreal, Quebec.

[12] National Institute of Standards and Technology. 2026. Secure Software Development Framework SSDF. https://csrc.nist.gov/projects/ssdf. Acesso em: 16 maio 2026.

[13] OWASP Foundation. n.d.. OWASP SAMM. https://owasp.org/www-project-samm/. Acesso em: 16 maio 2026.

[14] Filippo Santoni De Sio and Giulio Mecacci. 2021. Four Responsibility Gaps with Artificial Intelligence: Why they Matter and How to Address them. *Philosophy & Technology* 34, 4 (Dec. 2021), 1057–1084. doi:10.1007/s13347-021-00450-x

[15] Xiangmin Shen, Lingzhi Wang, Zhenyuan Li, Yan Chen, Wencheng Zhao, Dawei Sun, Jiashui Wang, and Wei Ruan. 2025. PentestAgent: Incorporating LLM Agents to Automated Penetration Testing. In *Proceedings of the 20th ACM Asia Conference on Computer and Communications Security*. ACM, Hanoi Vietnam, 375–391. doi:10.1145/3708821.3733882