From Black-Box to Glass-Box: Explainable AI for Applied Intelligent Software Systems Across the Software Development Life Cycle

Thi-Hong-Cuc Le[1,2] and Xuan-Bach Le[1,2⋆]

[1] Faculty of Computer Science and Engineering, Ho Chi Minh City University of Technology (HCMUT), Ho Chi Minh City, Vietnam
[2] Vietnam National University Ho Chi Minh City, Ho Chi Minh City, Vietnam
{lthcuc.sdh242,lexuanbach}@hcmut.edu.vn

Abstract. Artificial intelligence (AI), particularly large language models (LLMs), is increasingly embedded across the Software Development Life Cycle (SDLC), yet the opacity of these systems limits trust, verification, and certification in software engineering (SE), especially in safety- and regulation-critical domains. This paper presents a systematic mapping study (SMS) of explainable AI (XAI) in SE, spanning classical machine learning, deep learning, and LLM-based tools across all SDLC phases. Using an SDLC- and persona-aware taxonomy, we synthesize evidence on explanation techniques, evaluation practices, and deployment contexts. Our analysis reveals a pronounced imbalance: XAI research focuses on implementation and testing, while requirements, design, maintenance, and operations remain underexplored. Faithfulness validation and human-centred evaluation are limited, and LLM-based explanations introduce additional risks related to over-trust and pipeline opacity. We conclude with a research agenda centered on faithfulness-by-design, persona-aware explanations, and certification-ready XAI pipelines, outlining a path from black-box to glass-box SE in high-stakes intelligent systems.

1 Introduction

Large Language Models (LLMs) are increasingly used to support tasks throughout the Software Development Life Cycle (SDLC), including requirements analysis, code generation, vulnerability detection, testing, and maintenance [62,60]. These models have demonstrated strong empirical performance and are now being integrated into development tools, CI/CD pipelines, and operational workflows. However, despite their effectiveness, LLM-based systems largely remain opaque: their internal reasoning is difficult to inspect, their failure modes are hard to anticipate, and their outputs can be challenging to validate in practice. In particular, AI-generated artifacts may appear syntactically correct while silently violating functional, security, or safety requirements [63], raising concerns about trust, accountability, and compliance in real-world deployments [33].

⋆ Corresponding author.

Explainable Artificial Intelligence (XAI) has therefore emerged as a key enabler of trustworthy AI-assisted SE. This need is especially pronounced in applied intelligent systems such as robotics, cyber-physical systems (CPS), and industrial automation, where AI-based components support controller synthesis, test generation, and operational decision-making [25]. In these settings, failures can lead not only to software defects but also to physical harm or regulatory non-compliance, making explainability essential for safety argumentation, certification, and auditability [18]. As a result, explainability isn't just a nice-to-have—it's essential for integrating, validating, certifying, and safely deploying intelligent systems in the real world.

At the same time, XAI research in SE remains fragmented. While studies cover diverse models, tasks, and techniques, they are often evaluated in isolation with inconsistent assumptions [5,8]. As LLMs interact with distributed artifacts, like code, tests, CI/CD scripts, and logs [62], explanations are often ad hoc, poorly grounded, or misaligned with stakeholder needs [24]. As a result, practitioners lack clear guidance on which XAI methods best support specific SDLC phases, roles, and risk contexts, especially in real-world intelligent systems subject to operational and regulatory constraints [17].

This paper presents a systematic survey of XAI techniques in AI-driven software engineering tools across the full SDLC. We review classical ML (e.g., defect prediction), deep learning (e.g., vulnerability detection), and LLM-based systems for tasks like requirements analysis, code generation, and testing. Our focus is on how explanation methods are applied, how their faithfulness and usefulness are evaluated, and how they support a range of roles, from developers and testers to operators and compliance officers [32,16,45].

We structure the survey around three dimensions: SDLC phase, target user and explanation type, and the underlying XAI approach. This lets us analyze explainability at a system level, surface gaps in deployment and evaluation, and compare practices across domains. Unlike past surveys, we focus on explainability as an operational need, which is crucial for trust, certification, and lifecycle assurance in real-world intelligent systems.

The key contributions of this paper are as follows:

- We provide a system-level overview of XAI techniques in AI-assisted SE, focusing on how explanations are integrated into SDLC workflows and real-world environments (Sections 2 and 4).
- We identify common design patterns for persona-aware explainability, highlighting how developers, testers, operators, and compliance teams use explanations across SDLC phases (Sections 3 and 4).
- We analyze key deployment challenges, such as explanation faithfulness, evaluation under constraints, and certification readiness. We then propose practical research directions for explainable, LLM-based systems in safety and regulation-critical domains (Sections 5 and 6).

The remainder of this paper is organized as follows. Section 2 positions our contribution. Section 3 describes the survey methodology and classification framework. Section 4 surveys explainable AI techniques across the SDLC. Section 5

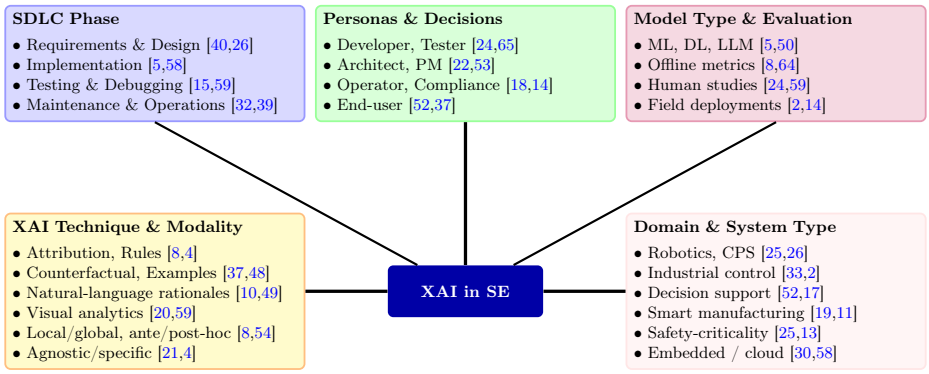

Fig. 1. Classification framework for XAI in SE spanning SDLC phases, personas, explanation techniques, model evaluation, and application domains.

synthesizes the findings and discusses their implications. Section 6 concludes the paper and outlines the research agenda.

2 Positioning

General XAI surveys provide foundational taxonomies and evaluation frameworks but remain model- and domain-agnostic, disconnected from SE artifacts such as source code and CI/CD pipelines [8]. Surveys on AI in SE cover code generation, defect prediction, and testing [5], yet treat explainability as secondary, overlooking faithfulness and stakeholder diversity. Recent task-specific studies target LLMs for code generation [50] or fault localization [12], but lack a systematic cross-phase perspective on roles and deployment settings [24]. Our work addresses this gap with an SDLC-wide, persona-aware synthesis spanning classical ML, DL, and LLM-based tools.

3 Methodology and Classification Framework

Following systematic mapping study protocols [29,38], we synthesized evidence on XAI integration in AI-driven SE tools across the SDLC. Figure 1 outlines the classification framework; Table 3 details the taxonomy.

3.1 Search and Selection

Systematic searches were conducted across IEEE Xplore, Scopus, and ScienceDirect on 15 January 2026, targeting studies (2019–2025) intersecting explainability concepts with implementation-centric (e.g., code generation) or lifecycle-oriented (e.g., DevOps) SE terms. ScienceDirect employed a sequential refinement protocol. The initial search yielded 126 records (Table 1); complete query strings are provided in Section A.

Study selection adhered to PRISMA [36] (Fig. 2). After duplicate removal ($n = 20$), 106 records underwent screening; 30 were excluded for ineligibility.

Table 1. Search results by digital library.

Database	Records	Included Studies
IEEE Xplore	43	22
Scopus	68	28
ScienceDirect	15	9
Total	**126**	**59**

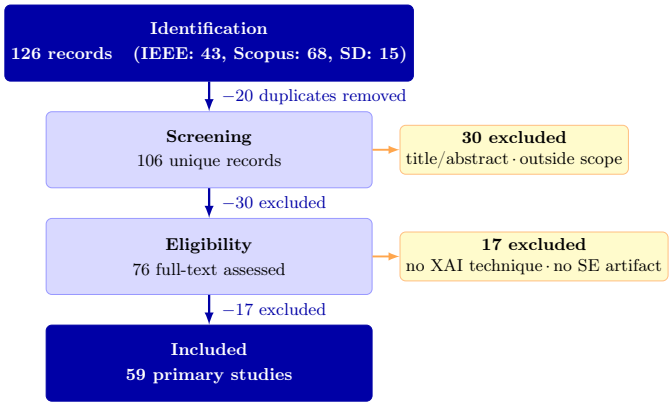

Fig. 2. Study selection process (PRISMA-style).

Full-text assessment of 76 papers excluded 17 lacking empirical XAI-SE integration, yielding 59 studies. Inclusion required AI/ML/LLM[3] deployment, explicit explainability, and SE artifact linkage; exclusions covered position papers, domain-agnostic works, non-English, and pre-2019.

3.2 Quality Assessment and Extraction

Studies were evaluated on: *Evaluation Rigor* (explicit metrics), *Faithfulness* (causal vs. proxy validation), and *Persona Grounding* (explicit role operationalization), coded as Yes/Partial/No (Table 2). Data extraction used the six-dimensional taxonomy (Table 3). Inter-rater reliability on a 20% subset yielded Cohen's $\kappa = 0.81$ [31]; discrepancies were resolved by consensus. Multi-phase studies were assigned by dominant task; survey papers spanning multiple phases were assigned to the phase most emphasized in their discussion.[4]

Although our contribution is not a new XAI method, it is an empirical synthesis: we apply PRISMA-based selection, quality coding (Q1–Q3), and a trace-

[3] Throughout this study, 'LLM' subsumes all pre-trained transformer-based language models for code, including encoder-only models (e.g., BERT [4]) and decoder-only generative models (e.g., GPT-4, CodeLlama).

[4] For example, Arora et al. [7], a phase-specific survey of XAI in the SDLC, was assigned to Maintenance & Operations based on its dominant emphasis on evaluation and operational deployment concerns.

Table 2. Quality assessment of 59 primary studies.

Indicator	Yes	Partial	No
Q1 Evaluation type reported	59	0	0
Q2 Faithfulness validated	0	16	43
Q3 Persona-grounded	8	13	38

Table 3. XAI taxonomy dimensions for SE contexts.

Dimension	Categories	SE examples
Scope	Local, Global	Line-level [32]; system-level [39]
Timing	Ante-hoc, Post-hoc	Chain-of-thought [40]; SHAP [32]
Modality	Visual, Textual, Interactive	Heatmaps [8]; rationales [16]
Technique	Attribution, Rules, Counterfactual, RAG	SHAP/LIME [63]; CWE retrieval [32]
Persona	Dev, Tester, Architect, PM, Compliance	Inline explanations [10]; risk summaries [17]
SDLC phase	Req., Des., Impl., Test., Maint.	NFR classification [40]; code review [46]

able per-study coding matrix (Section B) to support reproducible claims about coverage, evaluation, and faithfulness.

Explanation needs differ fundamentally across stakeholder roles: developers require line-level, actionable insights [32,10]; testers need failure localization cues [59,15]; architects and operators prioritize system traceability and runtime monitoring [22,14]; and compliance officers demand audit trails aligned with regulatory standards [18,52]. These variations underscore that persona-aware design is not a cosmetic concern—different roles demand fundamentally different explanation granularity, modality, and evaluation criteria.

3.3 Phase Distribution

Table 4 reveals a pronounced coverage imbalance: Implementation and Security dominate (39%), while Requirements and Design remain underrepresented (14%). Requirements studies emerged only in 2023–2025 versus 2022–2026 for Implementation, indicating early-phase XAI is nascent. Publication volume surged in 2024–2025, reflecting rapid LLM adoption. These disparities inform the research agenda in Section 6. Cross-dimensionally, persona grounding remains sparse: only 8 of 59 studies (14%) operationalize a concrete stakeholder role (Q3 = Yes), reinforcing that persona-aware XAI is still under-evaluated despite frequent claims of stakeholder alignment.

Across dimensions, Rationale-based (Ra) techniques predominate in phases—most notably Implementation & Security (Ra: 61%, LLM share: 87%) and, to a lesser extent, Requirements (Ra: 63%, LLM share: 63%)—while Attribution

Table 4. Distribution of 59 primary studies across SDLC phases, model types, and faithfulness validation (Q2). Row percentages for Q2 categories shown in parentheses.

SDLC Phase	Model Type			Q2: Faithfulness Validation			Total
	LLM	ML	DL	Yes	Partial	No	
Requirements & Design	5	3	0	0	3 (38%)	5	8 (14%)
Implementation & Security	20	3	0	0	6 (26%)	17	23 (39%)
Testing & Debugging	8	2	5	0	4 (27%)	11	15 (25%)
Maintenance & Operations	4	9	0	0	3 (23%)	10	13 (22%)
Total	**37**	**17**	**5**	**0**	**16**	**43**	**59**

* Q2 percentages computed relative to each row total. No study in the corpus achieved full causal faithfulness validation (Q2 = Yes).

Table 5. Representative XAI tools by SDLC phase, technique, persona, and evaluation.

Phase	Tool	Technique	Persona	Evaluation
Requirements	NICE [40]	GPT-4o + SLM	Req. Eng.	F1=0.84 + User study
	Robo-advice [52]	Regulatory mapping	Compliance	MiFID II alignment
Design	MLXOps4Medic [22]	Provenance traceability	Architects	ISO 13485
	XAIpipeline [58]	Multi-technique orchestration	Architects	Toolkit demo
	BabyFM [18]	SHAP + ISO compliance	Compliance	CE certification
Implementation	MalCodeAI [16]	Semantic summary	Security Eng.	4.2/5 rating
	SecureCoder [63]	RAG + CWE	Developers	Benchmark
	CoRe [57]	Multi-agent	Code reviewers	Accuracy + interpretability
	LA2SDP [32]	SHAP	DevOps	Nokia 5G trial
Testing	Scout [15]	LLM labeling	Testers	F1 score
	VISLIX [59]	Slice discovery	QA Engineers	User study
	XGraphRAG [56]	Graph retrieval + visual	Developers	Failure detection speed
Maintenance	Test-debt XAI [39]	Explainable prediction	DevOps	ESEM workshop
	DevSecOps Maritime [14]	Provenance + anomaly	Operators	Industrial deployment
	Well-integrity monitoring [2]	XAI dashboard	Operators	Field deployment

(At) techniques are more prevalent in the most ML-heavy phase, Maintenance & Operations (54%, ML share: 69%). This Technique × Phase skew—derivable from Section B—suggests that faithfulness challenges are structurally tied to the LLM-rationale pairing, not just to individual tools.

4 XAI Across the Software Development Life Cycle

Table 5 organizes representative XAI tools across SDLC phases by technique, persona, and evaluation strategy. The following sections examine how explainability shapes decisions, stakeholders, and methods at each stage, identifying key strengths and gaps.

4.1 Requirements and Design

Early SDLC phases involve requirements, regulations, and architecture decisions. Stakeholders—requirements engineers, architects, compliance officers—prioritize transparency, traceability, and standards alignment. XAI research focuses on explainable classification, rule-based reasoning, and provenance-aware design.

At the requirements level, frameworks like NICE leverage LLMs to classify non-functional requirements and generate explanatory rationales for validation [40]. In regulated domains, explainability serves as a governance imperative, exemplified by MiFID II obligation mapping in financial robo-advisors [52] and structured natural language for encoding driving rules in autonomous systems [26].

Architectural approaches embed explainability into lifecycle frameworks. MLX-Ops4Medic integrates provenance tracking and standards-based explanations into medical AI systems [22]. Compliance frameworks like BabyFM treat explanations as certification artifacts [18].

4.2 Implementation and Security

Implementation-stage decisions focus on code correctness, security, and defect risk, engaging developers, security engineers, and reviewers. XAI research centers on LLM-based code assistants, attribution techniques, RAG, and semantic summarization.

Probing and attention-based analyses reveal strengths in syntactic pattern recognition alongside weaknesses in logical reasoning and numeracy, delineating model capabilities and failure modes [54,9,34]. Applied directions integrate explanations into development workflows. Developer tools generate natural-language rationales, structured reports, or actionable suggestions for code completions and reviews [65,57]. In security contexts, RAG links code with known vulnerabilities or standards, producing risk summaries and remediation recommendations [16,63].

Attribution methods complement LLM systems by highlighting influential features behind defect predictions [32]. Explanations are most effective when aligned with developers' mental models, even with limited causal accuracy [24]. Overall, implementation-stage XAI delivers code-level insights, though evaluation prioritizes usability over faithfulness.

4.3 Testing and Debugging

Testing and debugging target failures, performance issues, and unexpected behavior, involving testers and developers. XAI research emphasizes visual, interactive, and example-based explanations for exploratory debugging.

LLMs add semantic context to low-level artifacts, such as labeling recorded GUI states to clarify failures and test intent [15]. Visual analytics tools combine program slicing and interactive views to spotlight failure-prone code and dependencies, aiding root-cause analysis [59,56]. LLMs also support performance debugging through natural-language summaries of profiling data [47].

These methods enhance debugging via state labels, code slices, graphs, and summaries.

4.4 Maintenance and Operations

Maintenance and operations address technical debt, anomaly detection, and assurance, involving DevOps, SREs, operators, and compliance teams. XAI re-

search emphasizes explainable prediction models and provenance-aware dashboards integrated into CI/CD and monitoring pipelines.

Approaches make maintenance risks interpretable by surfacing test debt and organizing metrics into meaningful factors for defect analysis [39,32]. Others support assurance by linking model outputs to standards and risks, enabling traceability in regulated domains [18,22]. At the pipeline level, orchestration platforms embed explanation methods into CI/CD workflows, supporting LLM validation and DevSecOps deployments [58,20,33,14,2]. However, limited integration with logs, traces, and SRE practices hinders reproducibility and adoption.

4.5 Mini Case Vignettes (Illustrative Deployments)

We summarize three representative deployments to ground the survey's findings: (i) **Telecom CI/DevOps (field trial):** LA2SDP [32] uses SHAP explanations for test-failure prediction in a Nokia 5G setting; evaluation emphasizes operational usefulness and decision support under pipeline constraints. (ii) **Medical MLOps (standards-oriented):** MLXOps4Medic [22] integrates provenance and traceability into an end-to-end lifecycle workflow; evaluation is framed around standards compliance (e.g., ISO 13485) and auditability artifacts. (iii) **Industrial DevSecOps (deployment):** DevSecOps Maritime [14] combines provenance/anomaly monitoring with operational dashboards; evaluation focuses on deployment feasibility and monitoring fidelity in an industrial pipeline.

4.6 Cross-Cutting: LLM-Based Explanation Strategies

Across the SDLC, LLM-based SE tools rely on recurring explanation strategies: chain-of-thought prompting, retrieve-then-explain pipelines, multi-agent reasoning, and intrinsic probing (Table 6).

Prompt-based CoT generates natural-language rationales alongside predictions, justifying non-functional requirement classifications or code decisions [40]. Retrieve-then-explain approaches ground explanations in external artifacts—vulnerability descriptions, dependency graphs, execution traces—by combining retrieval with LLM summaries, improving contextual relevance for security and debugging [63,56].

Multi-agent architectures decompose complex decisions across specialized agents, producing explanations through interaction or aggregation, particularly for code review and adaptive analysis [57,46]. Intrinsic techniques analyze internal representations to isolate syntactic and semantic properties, prioritizing mechanistic understanding [54,41]. Hybrid approaches combine AST-guided probing with prompt reasoning for specific defects [55]. While enhancing explanatory depth, these strategies share vulnerabilities—hallucinations, opacity, insufficient faithfulness—disproportionately affecting LLM tools. Among 37 LLM studies, only 4 (11%) achieved partial faithfulness ($Q2 \geq \text{Partial}$) versus 12 of 22 non-LLM studies (55%); none achieved full causal validation ($Q2 = \text{Yes}$), highlighting a critical gap. Disaggregating by LLM strategy (Section B), 0 of

Table 6. LLM explanation strategies in SE: mechanisms, tools, and limitations.

Strategy	Mechanism	Representative tools	Key limitation
Prompt CoT	Reasoning tokens	NICE [40], PN-CoT [23], AlethaNet [37]	Unverified faithfulness
RAG	Retrieve–then–explain	SecureCoder [63], XGraphRAG [56]	Pipeline opacity
Multi-agent	Role separation / consensus	CoRe [57], Real-Time Adaptive [46], MoA [35]	High latency
Intrinsic	Model probing	Code LM probing [54], InterpretSE [41], DeepTriFix [55]	Architecture-specific

Table 7. Cross-phase gaps and research directions for XAI in SE.

Gap	Impact	Proposed direction	Domains
Limited early XAI	Weak requirement – code traceability	XAI for NFR safety/compliance	Robotics, medical CPS
Unverified faithfulness	Over-trust and unsafe decisions	Causal probing and RAG-aware verification	Safety-critical SE
Limited human evaluation	Unclear real-world impact	Longitudinal studies in CI/CD pipelines	DevOps, SRE
Opaque LLM pipelines	Barriers to audit and certification	Prompt, retrieval, agent decision	CPS, finance, healthcare
Immature tooling and logging	Poor reproducibility	XAI-aware CI/CD templates	Industrial automation
Persona–XAI mismatch	Low adoption and trust	Co-designed interfaces for stakeholder roles	Governance-intensive systems

3 multi-agent and 0 of 3 intrinsic probing studies achieved even partial faithfulness (Q2 $\geq$ Partial); CoT and RAG fared only marginally better (1/3 and 2/4, respectively), confirming that hallucination and pipeline-opacity risks persist across all four explanation paradigms.

5 Discussion

This section identifies key gaps and structural challenges emerging from the survey and outlines research directions to advance XAI throughout the SDLC. A summary of findings is presented in Table 7 and Figure 3.

5.1 Coverage and Evaluation Gaps

The phase distribution in Table 4 reveals that explainability research concentrates on implementation-time artifacts, where code-level data and established benchmarks lower the evaluation barrier. A parallel persona imbalance compounds this: 40 studies (68%) target developers or testers, while architects, operators, and compliance officers—the roles most affected by certification and assurance gaps—are addressed by only 19 studies (32%), mainly in regulated

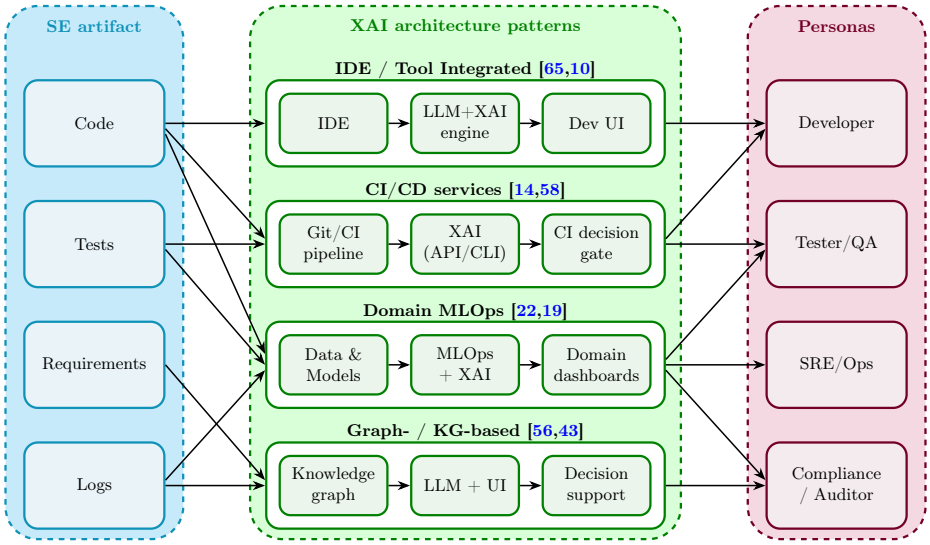

Fig. 3. Conceptual architectures for XAI-enabled SE tools: IDE assistants, CI/CD services, domain MLOps with provenance, and graph/KG-based XAI.

sectors [18,25,14]. Together, these skews mean that explainability is applied reactively—after defects surface in code—rather than proactively informing early design choices or supporting ongoing operational assurance.

Compounding the coverage gap, explanations are often presumed faithful without verifying alignment with model reasoning, particularly for post-hoc attributions and natural-language rationales assessed via plausibility or subjective perceptions. Human-centered evaluations remain small-scale, and absent longitudinal studies it remains unclear whether explanations mitigate risk or engender illusory confidence. Data quality issues such as mislabelled samples can further compromise model behaviour, rendering faithfulness attributions unreliable even with robust methods [44].

5.2 Architectural and Deployment Lessons

From an architectural standpoint, XAI-enabled SE systems are increasingly built around service-oriented and pipeline-based designs, integrating explanation components into IDEs, CI/CD workflows, and domain-specific MLOps stacks (Figure 3). These designs offer greater scalability and modularity but also introduce challenges such as latency, non-determinism, and unclear provenance. In real-world deployments, explanation logic often evolves separately from the underlying models, leading to drift and inconsistencies. Additionally, explanation logging and monitoring are still ad hoc, making it difficult to ensure reproducibility, enable audits, or compare systems reliably.

6 Research Agenda and Conclusion

Addressing gaps in Section 5, we propose three priorities for advancing XAI from ad hoc practices to principled, lifecycle-aware integration.

Table 8. Examples of certification-oriented requirements mapped to pipeline artifacts.

Requirement type	Pipeline artifact (example)
Governance / accountability (ISO/IEC 42001)	Decision logs: prompts, model/version, agent actions, approvals, access control
Traceability	Links from requirements/tests/issues to model outputs, explanations, and evidence bundles
Evaluation evidence (ISO/IEC 25059)	Benchmark reports + human/field evaluation summaries + acceptance criteria
Runtime assurance	Monitoring dashboards, drift alerts, incident postmortems, rollback records
Provenance	Retrieval provenance: sources, ranks, snapshots, data/model lineage

First, faithfulness-by-design is critical. Current post-hoc methods (e.g., attribution maps, rationales) lack verification against model behavior [8,50,33]. Future work must embed faithfulness via perturbation analysis, causal probing, and static-dynamic reasoning for both classical ML and LLMs [24,41].

Second, persona-aware explanations are essential. Evaluations remain small-scale and role-agnostic, lacking evidence of impact on decision quality or safety [16,25]. Shared benchmarks are required to link explanations to stakeholder decisions across personas and SDLC phases.

Third, certification-ready pipelines are needed. As LLM tools proliferate, explanations, provenance, and uncertainty must become core artifacts. This requires standardized logging of prompts and agent actions within XAI-aware CI/CD pipelines, aligned with AI governance and quality standards such as ISO/IEC 42001 [28] (AI management systems) and ISO/IEC 25059 [27] (quality model for AI-based systems) [22,14,10]. We emphasize that the proposed certification-ready pipeline is a conceptual blueprint derived from the mapped evidence, not a validated certification process. Validation would require end-to-end demonstrations that produce auditable artifacts (e.g., prompt/model/retrieval logs, traceability links, and evaluation reports) and are assessed in realistic CI/CD or operational deployments against applicable standards.

Collectively, these priorities shift focus from ad hoc explanations to certifiable XAI, enabling a transition from *black-box* to *glass-box* SE in high-stakes domains [25,51]. Realizing this vision requires treating explanations as core components—faithful, stakeholder-aware, and auditable—to ensure system-wide assurance.

7 Acknowledgments

We acknowledge Ho Chi Minh City University of Technology (HCMUT), VNUHCM for supporting this study.

References

1. Abboush, M., Ghannoum, E., Rausch, A.: An explainable hybrid deep learning-enabled intelligent fault detection and diagnosis approach for automotive software systems validation. Knowl.-Based Syst. **334** (2026)
2. Aditama, P., Eita, M., Koziol, T., Zhukova, K., Dillen, M., Sanchez, F., Schemmer, M., Nitsche, M., Tomina, M., Kimari, T.: Deployment of an ai-based well integrity monitoring solution. In: SPE Conf. (2024)
3. Ahi, K., Valizadeh, S.: Large language models (LLMs) and generative AI in cybersecurity and privacy: A survey of dual-use risks, AI-generated malware, explainability, and defensive strategies. In: SVCC. pp. 1–8 (Jun 2025)
4. Ahmad, N., Zhang, C.: Interpretable vulnerability detection in LLMs: A BERT-based approach with SHAP explanations. Comput. Mater. Contin. **85**(2), 3321–3334 (2025)
5. Ahmed, I., Aleti, A., Cai, H., Chatzigeorgiou, A., He, P., Hu, X., Pezzè, M., Poshyvanyk, D., Xia, X.: Artificial intelligence for software engineering: The journey so far and the road ahead. ACM Trans. Softw. Eng. Methodol. **34**(5) (2025)
6. Albaroudi, E., Mansouri, T., Hatamleh, M., Alameer, A.: Saudi arabia's vision 2030: Leveraging generative artificial intelligence to enhance software engineering. In: WiDS PSU. pp. 1–6 (Apr 2025)
7. Arora, L., Girija, S.S., Kapoor, S., Raj, A., Pradhan, D., Shetgaonkar, A.: Explainable artificial intelligence techniques for software development lifecycle: A phase-specific survey. In: COMPSAC. pp. 2281–2288 (Jul 2025)
8. Awal, M.A., Roy, C.K.: EvaluateXAI: A framework to evaluate the reliability and consistency of rule-based XAI techniques for software analytics tasks. J. Syst. Softw. **217**, 112159 (Nov 2024)
9. Baltaji, R., Thakkar, P.: Probing numeracy and logic of language models of code. In: InteNSE. pp. 8–13 (May 2023)
10. Bello, M., Bello, R., García, M.M., Nowé, A., Sevillano-Garcia, I., Herrera, F.: A three-level framework for LLM-enhanced explainable AI: From technical explanations to natural language. Inf. Syst. Front. (2025)
11. Berrouyne, O., El Bajta, M., Benelallam, I.: Systematic mapping study on AI integration in CI/CD pipelines. In: ICOA. pp. 1–6 (Oct 2025)
12. Chang, X., Li, D., Wong, W.E.: Large language models for software fault localization: A survey. In: DSA. pp. 111–120 (Nov 2025)
13. Esposito, M., Palagiano, F., Lenarduzzi, V., Taibi, D.: On large language models in mission-critical IT governance: Are we ready yet? In: ICSE-SEIP. pp. 504–515 (Apr 2025)
14. Firizqi, J.D., Indrajit, R.E.: Unified DevSecOps toolchain for secure AI-enabled maritime applications: Theory, concepts, and implementation. In: ICIC. pp. 1–6 (Oct 2025)
15. Franzosi, D.B., Alégroth, E., Isaac, M.: LLM-based labelling of recorded automated GUI-based test cases. In: ICST. pp. 453–463 (Mar 2025)
16. Gajjar, J., Subramaniakuppusamy, K., El Kachach, N.: Malcodeai: Autonomous vulnerability detection and remediation. In: IRI. pp. 31–36 (2025)
17. Gatchalee, P.: A model-driven and explainable framework for LLM-powered text classification in web intelligence and marketing applications. Adv. Artif. Intell. Mach. Learn. **5**(4), 4532–4552 (2025)
18. Gutic, B., Papić, T., Dakić, P.: Software quality and compliance in intelligent health monitoring systems: A case study of Baby FM. In: CEUR-WS Proc. vol. 4077 (2025)

19. Hegedűs, C., Varga, P.: Tailoring MLOps techniques for industry 5.0 needs. In: CNSM 2023 (2023)
20. Hu, B., Tunison, P., RichardWebster, B., Hoogs, A.: Xaitk-saliency: An open source explainable ai toolkit for saliency. In: AAAI. pp. 15760–15766 (2023)
21. Hu, B., Tunison, P., Vasu, B., Menon, N., Collins, R., Hoogs, A.: XAITK: The explainable AI toolkit. Appl. AI Lett. **2**(4) (2021)
22. Huang, J., Liu, Y.: MLXOps4Medic: A service framework for machine learning and explainability operations in medical imaging AI development. IEEE Access **13**, 158149–158169 (2025)
23. Huang, X., Li, R., Yang, J., Xiao, Q.: Positive-negative chain-of-thought prompting for table and text financial question answering. In: CBASE. pp. 196–199 (Oct 2025)
24. Huang, Z., Yu, H., Fan, G., Shao, Z., Li, M., Liang, Y.: Aligning XAI explanations with software developers' expectations: A case study with code smell prioritization. Expert Syst. Appl. **238**, 121640 (Mar 2024)
25. Ibrahim, A.A., Lim, H.K.: A deterministic assurance framework for licensable explainable AI grid-interactive nuclear control. Energies **18**(23) (2025)
26. Irvine, P., Da Costa, A.A.B., Zhang, X., Khastgir, S., Jennings, P.: Structured natural language for expressing rules of the road for automated driving systems. In: IV. pp. 1–8 (Jun 2023)
27. ISO/IEC 25059:2023: Quality model for AI-based systems (SQuaRE) (2023), standard. Geneva: ISO/IEC
28. ISO/IEC 42001:2023: AI management system (2023), standard. Geneva: ISO/IEC
29. Kitchenham, B., Charters, S.: Guidelines for performing systematic literature reviews in Software Engineering. Technical Report EBSE-2007-01, Keele University and Durham University (2007)
30. Kozub, V., Druzhynin, V., Trufanova, D., Ihnatenko, P., Kolos, K.: Using artificial intelligence in software development processes: Achievements and challenges. Sustain. Eng. Innov. **7**(2), 463–476 (2025)
31. Landis, J.R., Koch, G.G.: The measurement of observer agreement for categorical data. Biometrics **33**(1), 159–174 (1977)
32. Madeyski, L., Stradowski, S.: Predicting test failures induced by software defects: A lightweight alternative to software defect prediction and its industrial application. J. Syst. Softw. **223**, 112360 (May 2025)
33. Min, Z., Budnik, C.J.: Verification and validation of LLM-RAG for industrial automation. In: AITest. pp. 50–53 (Jul 2025)
34. Mohammadkhani, A.H., Tantithamthavorn, C., Hemmatif, H.: Explaining Transformer-based code models: What do they learn? when do they not work? In: SCAM. pp. 96–106 (Oct 2023)
35. Onan, A., Nasution, A.H., Celikten, T.: Toward reliable annotation in low-resource NLP: A mixture-of-agents framework and multi-LLM benchmarking. IEEE Access **13**, 211620–211644 (2025)
36. Page, M., McKenzie, J., Bossuyt, P., Boutron, I., Hoffmann, T., Mulrow, C., et al.: The PRISMA 2020 statement: an updated guideline for reporting systematic reviews. BMJ **372** (2021)
37. Panagoulias, D.P., Virvou, M., Tsihrintzis, G.A.: Truth in trees: A multilayered linguistic framework for automated fact-checking with AI-driven reasoning. Procedia Comput. Sci. **270**, 1032–1041 (Jan 2025)
38. Petersen, K., Feldt, R., Mujtaba, S., Mattsson, M.: Guidelines for systematic mapping studies in SE. Inf. Softw. Technol. **52**(3), 249–265 (2010)
39. Radnejad, M.: Explainable AI for identifying and managing test debt in automated testing. In: ESEM. pp. 528–531 (Oct 2025)

40. Rejithkumar, G., Anish, P.R.: NICE: Non-functional requirements identification, classification, and explanation using small language models. In: ICSE-SEIP. pp. 284–295 (Apr 2025)
41. Rodríguez-Cárdenas, D.: Towards more interpretable large language models for code. In: ACM SIGSOFT FSE. pp. 1270–1272 (2025)
42. Salem, D.O., Alahmed, Y., Fnich, R., Mazroub, M., Fnich, M.: AI-driven continuous integration: Automating code review and deployment with LLMs. In: FMEC. pp. 268–274 (May 2025)
43. Schiese, D., Perevalov, A., Both, A.: Towards llm-generated explanations for component-based knowledge graph question answering systems (2025), https://arxiv.org/abs/2508.14553
44. Shah, M.B., Rahman, M.M., Khomh, F.: Towards understanding the impact of data bugs on deep learning models in software engineering. Empir. Softw. Eng. **30**(6) (2025)
45. Sharanarthi, T.: Adaptive multi-agent AI framework for real-time energy optimization and context-aware code review in software development. In: ISCTIS. pp. 353–358 (May 2025)
46. Sharanarthi, T., Polineni, S.: Real-time adaptive code analysis with a self-learning multi-agent framework: A retrieval-augmented reinforcement learning approach. In: ICAIDE. pp. 534–540 (May 2025)
47. Sijwali, S.S., Colom, A.M., Guo, A., Saha, S.: Fixing performance bugs through LLM explanations. In: AITest. pp. 102–109 (Jul 2025)
48. Sovrano, F., Vitali, F.: An objective metric for explainable AI: How and why to estimate the degree of explainability. Knowl.-Based Syst. **278**, 110866 (Oct 2023)
49. Trigg, L., Morgan, B., Stringer, A., Schley, L., Hougen, D.F.: Natural language explanation for autonomous navigation. In: DASC. pp. 1–9 (Sep 2024)
50. Umama, Danyaro, K.U., Nasser, M., Zakari, A., Abdullahi, S., Khanzada, A., Yakubu, M.M., Shoaib, S.: LLM-based code generation: A systematic literature review with technical and demographic insights. IEEE Access **13**, 194915–194939 (2025)
51. Velasco, A., Garryyeva, A., Palacio, D.N., Mastropaolo, A., Poshyvanyk, D.: Toward neurosymbolic program comprehension. In: ICPC. pp. 377–381 (Apr 2025)
52. Vilone, G., Sovrano, F., Lognoul, M.: On the explainability of financial robo-advice systems. In: Commun. Comput. Inf. Sci. vol. 2156, pp. 219–242 (2024)
53. Walker, M., Forêt, J.M.: Unleashing the cognitive digital twin via semantic orchestration. In: IEEE Aerosp. Conf. pp. 1–15 (Mar 2025)
54. Wan, Y., Zhao, W., Zhang, H., Sui, Y., Xu, G., Jin, H.: What do they capture? – a structural analysis of pre-trained language models for source code. In: ICSE. pp. 2377–2388 (May 2022)
55. Wan, Z., Wei, L.: DeepTriFix: An LLM-based code defect detection framework guided by abstract syntax trees and prompt reasoning. In: AIoTC. pp. 708–713 (Aug 2025)
56. Wang, K., Pan, B., Feng, Y., Wu, Y., Chen, J., Zhu, M., Chen, W.: XGraphRAG: Interactive visual analysis for graph-based retrieval-augmented generation. In: PacificVis. pp. 1–11 (Apr 2025)
57. Wang, L., Zhou, Y., Zhuang, H., Li, Q., Cui, D., Zhao, Y., Wang, L.: Unity is strength: Collaborative LLM-based agents for code reviewer recommendation. In: ASE. pp. 2235–2239 (Oct 2024)
58. Wang, Z., Liu, Y.: XAIpipeline: Automated orchestration of explainable AI services for cloud AI and open-source models. In: SSE. pp. 50–56 (Jul 2025)

59. Yan, X., Xuan, X., Ono, J.P.P., Guo, J., Mohanty, V., Kumar, S.A., Gou, L., Wang, B., Ren, L.: VISLIX: An XAI framework for validating vision models with slice discovery and analysis. Comput. Graph. Forum **44**(3) (2025)
60. Yang, L., Chen, Y.: Integrating RAG and LLM for automated code review in practice. In: ISCIPT. pp. 591–596 (Sep 2025)
61. Yue, Z., Yang, J., Li, R., Xiao, Q.: A BIM code compliance checking method based on classification awareness and domain-constrained alignment. In: CBASE. pp. 598–602 (Oct 2025)
62. Zhang, Y., Chen, S., Fan, L.: A web-based tool for using storyboard of android apps. In: ICSE-Companion. pp. 117–121 (May 2023)
63. Zhao, J., Sun, Y., Huang, C., Liu, C., Guan, Y., Zeng, Y., Liu, Y.: Towards secure code generation with LLMs: A study on common weakness enumeration. IEEE Trans. Softw. Eng. **51**(12), 3507–3523 (Dec 2025)
64. Zhou, B., Zhao, H., Wen, Y., Ding, G., Xing, Y., Lin, X., Xiao, L.: Software defect prediction based on semantic views of metrics: Clustering analysis and model performance analysis. Comput. Mater. Contin. **84**(3), 5201–5221 (Jul 2025)
65. Zhu, S., Zhang, Y., Xu, W., Guo, W., Ye, P.: LLMSDH: An integrated software-assisted development tool based on large language models. In: QRS-C. pp. 671–678 (Jul 2025)

A Database-Specific Search Strings and Protocol

Search executed **15 Jan 2026**. Criteria: 2019–2025, English, peer-reviewed only.

IEEE Xplore *All Metadata*; year filters via interface.

```
("explainable AI" OR "XAI" OR "interpretability" OR "model explanation")
AND ("code generation" OR "code completion" OR "program synthesis"
     OR "code LLM" OR "large language model")
AND ("software engineering" OR developer OR "code review"
     OR "pull request" OR "test generation")
```

Scopus `TITLE-ABS-KEY`; two strings. *String 1 (Implementation):*

```
TITLE-ABS-KEY(("explainable AI" OR "interpretable AI" OR XAI
               OR "model explanation")
AND ("code generation" OR "code completion" OR "program synthesis"
     OR "large language model" OR "code LLM")
AND ("software engineering" OR developer OR "code review"
     OR "test generation" OR DevOps))
AND (PUBYEAR > 2018 AND PUBYEAR < 2027)
```

String 2 (Lifecycle):

```
TITLE-ABS-KEY(("explainable AI" OR "interpretable AI" OR XAI
               OR "interpretability")
AND ("software development lifecycle" OR SDLC OR DevOps OR "MLOps"
     OR "V-Model" OR "verification and validation")
AND (code OR "source code" OR "test" OR "test case"
     OR "software system"))
AND (PUBYEAR > 2018 AND PUBYEAR < 2027)
```

ScienceDirect Sequential refinement (interface constraints).

1. **Initial** (Title/Abstract/Keywords):

   `"code generation" OR "program synthesis" OR "AI-assisted programming"`

2. **Refinement**:

   `"explainable AI" OR "interpretable" OR "interpretability" OR XAI`

3. **Filters**: 2019–2025 via interface.

Total initial records: 126 (Table 1).

B Primary Study Coding Scheme

Table 9: Coding scheme for 59 studies (refs: Tables 3, 2). **Keys:** Ph: R=Req&Des, I=Impl&Sec, T=Test, M=Maint; Md: ML, DL, LLM; Sc: L/G; Ti: A/P; Mo: Vi/Tx/In; Te: At/Ru/Cf/Ra; Pe: De/Te/Ar/Op/Co; Q1–3: Y/P/N. †=included not cited; dominant-task rule [38].

Study	Ph	Md	Sc	Ti	Mo	Te	Pe	Q1	Q2	Q3
Req. & Design *n=8*										
LLM text classif. (ReqEng) [17]	R	LLML		A	Tx	Ra	Ar	Y	N	N
SW quality & compliance (BabyFM) [18]	R	ML	G	P	Tx	Ru	Co	Y	P	Y
Struct. NL for driving rules [26]	R	ML	G	A	Tx	Ru	Ar	Y	N	P
NICE: NFR classif. & expl. [40]	R	LLML		A	Tx	Ra	Ar	Y	P	Y
NL expl. (autonomous nav.) [49]	R	LLML		A	Tx	Ra	Op	Y	N	P
Explainability of robo-advice [52]	R	ML	G	P	Tx	Ru	Co	Y	P	Y
BIM compliance checking (LLM) [61]	R	LLML		A	Vi	Ra	Ar	Y	N	N
Storyboard tool (Android) [62]	R	LLML		P	Vi	Ra	De	Y	N	N
Impl. & Security *n=23*										
LLMs in cybersecurity/privacy [3]	I	LLMG		P	Tx	Ra	De	Y	N	N
Interp. vuln. detect. (BERT+SHAP) [4]	I	LLML		P	Tx	At	De	Y	P	N
AI for SE: journey & road ahead [5]	I	LLMG		P	Tx	Ra	De	Y	N	N
EvaluateXAI (rule-based) [8]	I	ML	L	P	Tx	At	De	Y	P	N
Probing numeracy/logic in code LMs [9]	I	LLMG		P	Tx	At	De	Y	N	N
3-level LLM-XAI framework [10]	I	LLML		A	Tx	Ra	De	Y	N	N
MalCodeAI: vuln. detect. & remediat. [16]	I	LLML		A	Tx	Ra	De	Y	N	N
Aligning XAI w/ dev. expectations [24]	I	ML	L	P	Tx	At	De	Y	P	Y
Predicting test failures (SHAP) [32]	I	ML	L	P	Tx	At	De	Y	P	N
Explaining Transformer code models [34]	I	LLMG		P	Tx	At	De	Y	N	N
Interpretable LLMs for code [41]	I	LLML		P	Tx	At	De	Y	N	N
Multi-agent code review (energy) [45]	I	LLML		A	Tx	Ra	De	Y	N	P
Real-time adaptive code analysis [46]	I	LLML		A	Tx	Ra	De	Y	N	N
LLM code gen.: SLR [50]	I	LLMG		P	Tx	Ra	De	Y	N	N
Neurosymbolic program comprehension [51]	I	LLML		P	Tx	At	De	Y	N	N
What do code LMs capture? [54]	I	LLMG		P	Tx	At	De	Y	N	N
DeepTriFix: AST-guided defect detect. [55]	I	LLML		A	Tx	Ra	De	Y	N	P
CoRe: multi-agent code reviewer [57]	I	LLML		A	Tx	Ra	De	Y	N	P
RAG+LLM automated code review [60]	I	LLML		A	Tx	Ra	De	Y	P	N
SecureCoder: CWE-guided code gen. [63]	I	LLML		A	Tx	Ra	De	Y	P	N
LLMSDH: LLM-assisted dev. tool [65]	I	LLML		A	Tx	Ra	De	Y	N	P
Gen. AI to enhance SE (Vision 2030)† [6]	I	LLMG		P	Tx	Ra	De	Y	N	N
AI-driven CI: code review w/ LLMs† [42]	I	LLMG		P	Tx	Ra	De	Y	N	N
Testing & Debugging *n=15*										
Expl. fault detect. (automotive CPS) [1]	T	DL	L	P	Vi	At	Te	Y	P	N
LLMs for fault localization (survey) [12]	T	LLMG		P	Tx	Ra	De	Y	N	N
LLMs in mission-critical governance [13]	T	LLMG		P	Tx	Ra	Op	Y	N	P
LLM-based GUI test labelling [15]	T	LLML		A	Tx	Ra	Te	Y	N	P
XAITK-Saliency toolkit [20]	T	DL	L	P	Vi	At	Te	Y	N	N
XAITK: XAI toolkit [21]	T	DL	L	P	Vi	At	Te	Y	N	N
PN-CoT: financial QA [23]	T	LLML		A	Tx	Ra	Te	Y	N	N
Assurance framework (nuclear CPS) [25]	T	ML	L	P	Tx	At	Co	Y	P	Y
Reliable annotation (MoA/NLP) [35]	T	LLML		A	Tx	Ra	Te	Y	N	N
Truth in Trees (fact-checking) [37]	T	LLML		A	Tx	Ra	De	Y	N	N
Impact of data bugs on DL [44]	T	DL	G	P	Tx	At	De	Y	N	N
Fixing perf. bugs via LLM expl. [47]	T	LLML		A	Tx	Ra	De	Y	N	P
Cognitive digital twin (semantic) [53]	T	ML	G	P	In	At	Op	Y	P	P
XGraphRAG: graph RAG visual analytics [56]	T	LLML		P	Vi	Ra	De	Y	N	P
VISLIX: vision model validation [59]	T	DL	G	P	Vi	At	Te	Y	P	Y
Maint. & Ops *n=13*										
AI well integrity monitoring [2]	M	ML	L	P	Vi	At	Op	Y	P	Y
AI integration in CI/CD (mapping) [11]	M	LLMG		P	Tx	Ra	Op	Y	N	N
DevSecOps toolchain (maritime) [14]	M	ML	G	P	Tx	At	Op	Y	N	P
MLOps for Industry 5.0 [19]	M	ML	G	A	Tx	Ru	Ar	Y	N	N
MLXOps4Medic (medical AI MLOps) [22]	M	ML	L	P	Vi	At	Ar	Y	P	Y
AI in SW dev: achievements [30]	M	ML	G	P	Tx	Ra	De	Y	N	N
V&V of LLM-RAG (industrial) [33]	M	LLML		A	Tx	Ra	Op	Y	N	P
XAI for test-debt management [39]	M	ML	L	P	Tx	At	De	Y	N	N
LLM explanations for KGQA [43]	M	LLML		A	Tx	Ra	De	Y	N	N
Objective XAI metric (DoX) [48]	M	ML	G	P	Tx	At	Ar	Y	P	N
XAIpipeline: XAI orchestration [58]	M	ML	G	P	In	At	Ar	Y	N	N
SW defect predict. (semantic views) [64]	M	ML	L	P	Tx	At	De	Y	N	N
XAI techniques for SDLC (survey)† [7]	M	LLMG		P	Tx	Ra	Ar	Y	N	N