Hybrid Cognitive Systems in the Era of LLMs: Distributed Cognition, the Rise of "Vibe Coding", and New Sociotechnical Risks in Brazilian Software Engineering

Marcelo Massashi Simonae[1] , Marlon Marcon[1]

[1]Programa de Pós-Graduação em Informática (PPGI-CP-DV), Universidade Tecnológica Federal do Paraná (UTFPR) Dois Vizinhos– PR– Brasil
Estrada para Boa Esperança, Km 04 - Dois Vizinhos– 85660-000

msimonae@alunos.utfpr.edu.br, marlonmarcon@utfpr.edu.br

Abstract. *The integration of Large Language Models (LLMs) into Software Engineering (SE) practices is catalyzing a profound sociotechnical transformation. While LLMs promise unprecedented productivity gains, this intuitive approach radically alters the mediation of developer intent. This research presents a* ***vision and research proposal*** *arguing that the symbiotic interaction between software developers and generative AI agents gives rise to Hybrid Cognitive Systems (HCS). This paper establishes a structured conceptual framework to analyze how HCS operate, synthesizing multi-agent agile workflows, shifting human-AI role dynamics, and cognitive adoption factors. Furthermore, we critically examine how these systems may lead to the erosion of engineering best practices and technical debt. Due to the lack of current empirical data, this work proposes a* ***comprehensive, future qualitative methodological procedure*** *through a multiple-case study in the Brazilian IT industry, modeling the tension between augmented cognition and technical degradation to evaluate the sector's "Organizational Readiness" for advanced autonomous automation and Artificial General Intelligence (AGI).*

1. Introduction

The landscape of Software Engineering is undergoing a disruption catalyzed by the massive use of Generative Artificial Intelligence (GAI) and Large Language Models (LLMs) [Abrahão et al. 2025, Salem et al. 2024]. The adoption of AI-assisted tools automates complex workflows and promotes a paradigm shift, transferring fundamental activities—such as code generation, testing, documentation, and agile sprint planning—to autonomous multi-agent ecosystems [Nasir and Hussain 2024, Cinkusz et al. 2025].

Amidst these transformations emerges the concept of "Vibe Coding", coined to reflect a new modality of creation driven purely by user intentions and conceptual descriptions, actively disregarding the existence of code at the syntactic level [Gadde 2025]. While the democratization of software development is desirable

to expand the technological ecosystem, the dominant narrative prioritizing hyper-productivity obscures profound cognitive reconfigurations and sociotechnical risks [Viswanadhapalli 2024, Dhole 2023]. .

The transition from manual, structured coding to probabilistic AI co-creation generates Hybrid Cognitive Systems (HCS), where mental labor and technical agency are dynamically distributed. However, the lack of proper oversight can result in the systemic accumulation of technical debt, translating into unstable codebases. The prompt-based interaction surface also introduces severe information security vulnerabilities to the CIA Triad (Confidentiality, Integrity, and Availability), such as Prompt Injection attacks [Derner et al. 2024], and brittle system behaviors due to extreme prompt sensitivity [Zou 2024].

The core problem outlined in this paper is the emergent tension between AI-enabled production acceleration and the maintenance of technical rigor. How Brazilian software companies react to these risks during the current adoption of Weak AI will define their organizational readiness and survival in the horizon of Artificial General Intelligence (AGI).

2. Theoretical Framework

2.1. Theoretical Operationalization of Hybrid Cognitive Systems

To move beyond purely conceptual discussions and make the Hybrid Cognitive System (HCS) construct empirically verifiable in field research, we propose two formal metrics of cognitive coupling and sociotechnical degradation:

1. **Cognitive Coupling Index (CCI):** This metric measures the distribution of cognitive effort between the human developer (H) and the AI assistant (A) during a given software development task:

$$CCI = \frac{T_{verify} + T_{explain}}{T_{total}} \times \left(1 - \frac{Lines_{AI}}{Lines_{total}}\right) \tag{1}$$

   where T_{verify} is the time spent by the human developer actively inspecting and debugging AI-generated suggestions, $T_{explain}$ is the time dedicated to requesting explanations and reverse-engineering the model's reasoning [Tadi 2025], $Lines_{AI}$ is the number of synthetic code lines accepted, and $Lines_{total}$ is the total size of the final commit. A low CCI represents a weak cognitive coupling, indicating a high risk of automation complacency, blind acceptance, and rapid developer de-skilling [Bainbridge 1983].
2. **Syntax Brittleness Rate (SBR):** This metric measures the instability and degradation introduced into the codebase as a consequence of probabilistic coupling under high prompt-sensitivity:

$$SBR = \frac{Builds_{failed_AI}}{Builds_{total}} \times \lambda_{debt} \tag{2}$$

   where $Builds_{failed_AI}$ represents CI/CD pipeline build failures caused by syntax errors, type mismatches, or API hallucinations in AI-assisted commits [Zou 2024], and λ_{debt} is a weighting multiplier reflecting the architectural technical debt, calculated using the cyclomatic complexity accrued in the modified files [Viswanadhapalli 2024].

2.2. Democratization, Intent, and "Vibe Coding"

The gradual democratization of software development—evolving from hardware configuration to high-level languages and Low-Code/No-Code platforms [Gadde 2025]—has culminated in "Vibe Coding" powered by LLMs [Meske et al. 2025]. This new paradigm shifts intent mediation from rigid syntactic rules to AI-driven probabilistic inference. While this transition minimizes cognitive barriers for novices and enables domain experts to build software, it introduces the critical risk of "illusory mastery"—a dangerous disconnect where immediate, working software masks a lack of underlying architectural integrity, ultimately driving the de-skilling of human developers [Tadi 2025]. This profound shift in agency and intent is contrasted with traditional and low-code approaches in Table 1.

Table 1. Comparative Analysis of Software Development Approaches (Adapted from Gadde, 2025 and Meske et al., 2025)

Dimension	Traditional Development	Low-Code/No-Code	Vibe Coding Approach
Time to Market	Weeks to months	Days to weeks	Hours to days
Intent Mediation	Strict and formal syntax rules	Visual manipulation of components	Probabilistic, based on natural language
Epistemic Agency	Human retains full authorship	Logic embedded in the platform	Shared agency (AI proposes, human curates)

2.3. Human-AI Collaboration and Role Dynamics

Traditional practices like Pair Programming established fixed roles of "driver" and "navigator." With LLMs, this dyad is reconfigured: humans act as "orchestrators" or "verifiers," while AI serves as "explorer" and generator [Tadi 2025]. Figure 1 illustrates the functional ecosystem of these AI assistants.

2.4. AI-Assisted Code Reviews

Alami and Ernst (2025) highlight a shift in engagement: while traditional peer reviews often involve emotional strain from harsh feedback, LLM-generated reviews avoid this friction through a consistent and professional tone. However, they increase the *cognitive load*, as engineers must carefully validate the completeness and contextual accuracy of AI suggestions [Alami and Ernst 2025].

2.5. Adoption Factors and Cultural Values

To understand why developers embrace HCS, Lambiase et al. (2025) applied the Unified Theory of Acceptance and Use of Technology (UTAUT2) combined with Hofstede's Cultural Dimensions. Their Partial Least Squares-Structural Equation Modeling (PLS-SEM) revealed that *Performance Expectancy* and *Habit* are the primary drivers of LLM adoption [Lambiase et al. 2025]. Interestingly, their findings indicate that individual cultural values (such as Collectivism or Power Distance) **do not significantly moderate** the adoption of LLMs. The pursuit of

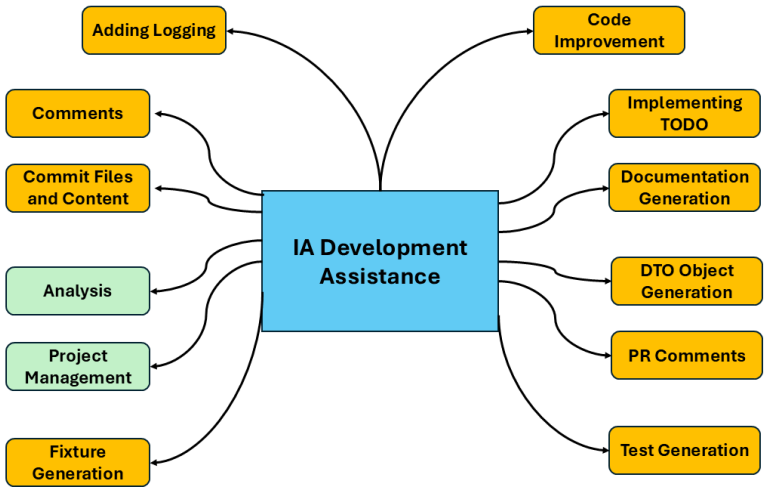


Figure 1. Functional Roles of AI Developer Assistants in Collaborative Software Development (Adapted from Tadi, 2025 [Tadi 2025]).

hyper-productivity and immediate efficiency transcends cultural barriers in software engineering [Lambiase et al. 2025]. Table 2 summarizes the impact of these core UTAUT2 constructs and their empirical implications for LLM adoption.

Table 2. Key Drivers of LLM Adoption in Software Engineering (Adapted from Lambiase et al., 2025)

Construct (UTAUT2)	Empirical Finding & Implication
Performance Expectancy	Strongest predictor of adoption. Engineers are highly motivated by tools that demonstrably enhance coding efficiency and debugging accuracy.
Habit	Second most influential factor. Regular integration of LLMs into daily workflows solidifies long-term reliance on these systems.
Cultural Values	Found to be statistically insignificant as moderators. The disruptive, highly beneficial nature of LLMs overshadows local cultural contexts.

2.6. Multi-Agent Systems in Agile Workflows

The evolution of HCS extends to Multi-Agent Systems (MAS). Frameworks such as *CogniSim* show how LLMs enable cognitive personas (e.g., Manager, Executor, Quality Checker) that automate Agile and SAFe workflows [Cinkusz et al. 2025, Nasir and Hussain 2024]. These agents collaborate to analyze requirements, generate code, and maintain documentation, reshaping the Software Development Life Cycle (SDLC) [Salem et al. 2024]. Figure 2 illustrates this layered MAS architecture integrating LLMs with agile APIs to support human teams.

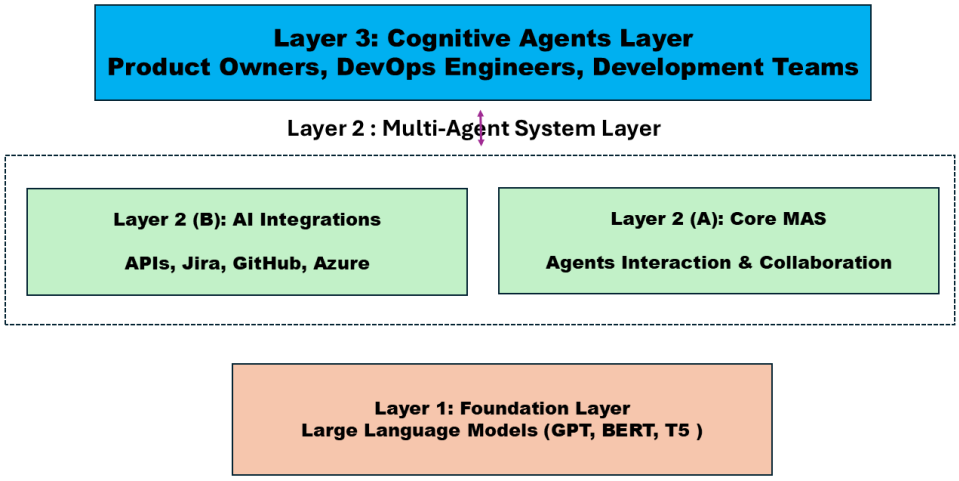


Figure 2. Layered ecosystem of CogniSim integrating LLMs, MAS, and cognitive personas (Adapted from Cinkusz et al., 2025).

2.7. Sociotechnical Risks: Security, Formalism, and Brittleness

The Theory of Distributed Cognition warns that cognitive offloading to LLMs embeds opacity into applications. Treating them as neutral tools leads to "abstraction and formalism traps" [Dhole 2023], introducing risks classified in the LLM Security Taxonomy [Derner et al. 2024]:

- **Prompt Injection:** Malicious inputs that bypass alignment or inject harmful logic.
- **Data Poisoning:** Manipulating training data to degrade performance or insert backdoors.
- **Brittleness:** Extreme sensitivity to prompt syntax, undermining reliability [Zou 2024].

To mitigate opacity, organizations must integrate Explainable AI (XAI) frameworks such as SHAP [Lundberg and Lee 2017], LIME [Ribeiro et al. 2016], and Anchor [Ribeiro et al. 2018]. Embedding these into Artifact Analysis routines bridges traditional governance with modern MLOps and FinOps, ensuring secure, transparent, and scalable codebases.

2.8. Dynamic Capabilities and the Transition to Advanced Automation

Building on Teece's *Dynamic Capabilities* framework [Teece 2007], current sociotechnical routines—such as prompt sanitization [Derner et al. 2024], automated data mesh auditing, FinOps profiling [Mäkinen et al. 2021], and XAI review gates [Alami and Ernst 2025]—form organizational **microfoundations** (sensing, seizing, transforming) for advanced autonomous workflows. Table 3 summarizes our Sociotechnical Maturity Model.

2.9. Regulatory Compliance and Privacy Governance (LGPD)

While "Vibe Coding" accelerates delivery, it creates regulatory blind spots under Brazil's LGPD [Brasil 2018]. Delegating tasks to external LLM APIs often exposes

Table 3. Sociotechnical Maturity Model for Advanced Software Automation.

Level	Current Focus	Transition Capability
1. Reactive	Ad-hoc coding; silent technical debt.	Low adaptability; vulnerability to LLM failures.
2. Managed	Prompt guidelines; manual AI review.	Human oversight; containment of brittleness.
3. Dynamic	Continuous XAI; FinOps/MLOps gates.	High resilience; governance of autonomous MAS.

proprietary code and sensitive data, violating principles of security and purpose limitation. LLMs may also memorize and leak training data, enabling extraction attacks [Carlini et al. 2021].

Table 4 summarizes key sociotechnical risks and their LGPD impacts.

Table 4. Matrix of LLM Sociotechnical Risks and LGPD Compliance Impacts.

Risk	Engineering Impact	LGPD Impact
Data Memorization	Inclusion of PII (Personally Identifiable Infor mation in prompts.	Violation of Article 46; unauthorized exfiltration.
Prompt Injection	Malicious logic bypassing controls.	Breach of confidentiality; mandatory notifications (Art. 48).
Abstraction Traps	Non-compliant processing algorithms.	Violation of Privacy by Design and transparency.

To mitigate these risks, Hybrid Cognitive Systems must embed architectural safeguards. Figure 3 shows a Privacy-Oriented Governance Pipeline where local sanitization anonymizes sensitive data before reaching external LLMs, ensuring compliance.

2.10. MLOps, FinOps, and the Economic Sustainability of HCS

LLM-assisted workflows introduce a critical dimension: the economic sustainability of cloud infrastructure. While "Vibe Coding" accelerates delivery, it often produces code with excessive complexity or inefficient execution paths. In production, these anti-patterns inflate cloud costs and waste resources. Sustainable governance therefore requires integrating Machine Learning Operations (MLOps) and Cloud Financial Management (FinOps) [Mäkinen et al. 2021].

Rigorous benchmarking of models and generated code helps prevent infrastructure debt. Table 5 highlights how common coding inefficiencies impact cloud spending and how MLOps strategies mitigate them [Kreuzberger et al. 2023].

To enforce sustainability, deployment architectures must embed FinOps and MLOps gates. Figure 4 shows a CI pipeline where automated checks act as barriers against costly synthetic code.

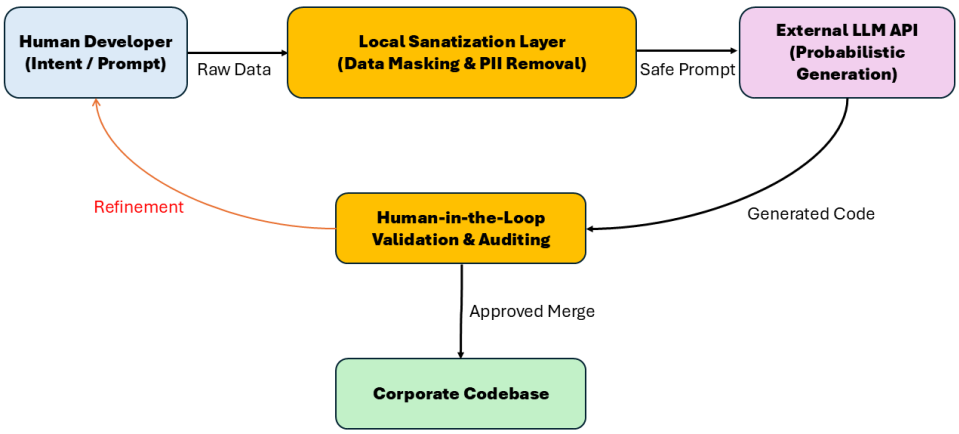


Figure 3. Privacy-Oriented Governance Pipeline ensuring LGPD compliance in LLM-assisted development (Adapted from Alami, 2025).

Table 5. FinOps Impact Matrix on LLM-Generated Code Anti-patterns.

Anti-Pattern	Cloud Impact	MLOps Mitigation
Unoptimized Queries	Rising compute and data egress costs.	Query benchmarking and CI/CD cost gates.
Redundant API Calls	Higher billing and throttling risks.	Semantic analysis enforcing caching and routing.
Over-provisioned Instances	Wasted spending on bloated execution.	Dynamic profiling and strict budgeting policies.

2.11. Cognitive Resilience and the Mitigation of Human Deskilling

As highlighted in the conceptual framework (Figure 5), "Human Deskilling" acts as a critical resisting force against the sustainable adoption of LLMs. The continuous delegation of complex algorithmic problem-solving to artificial agents induces automation complacency, a well-documented human factors phenomenon where operators lose situational awareness and foundational expertise [Bainbridge 1983]. In Hybrid Cognitive Systems, this manifests as junior developers acting merely as "prompt dispatchers," incapable of debugging deep architectural flaws when the AI hallucinates or fails. To counteract this, organizations must establish Cognitive Resilience Frameworks [Endsley 2017].

These frameworks enforce pedagogical feedback loops where knowledge is actively retained by the human workforce. Table 6 maps specific cognitive risks introduced by AI dependency to actionable engineering mitigation strategies.

3. Methodological Procedures

To investigate the socio-cognitive dynamics of HCS and evaluate sociotechnical risk mitigation within the Brazilian software ecosystem, this study proposes a flexible,

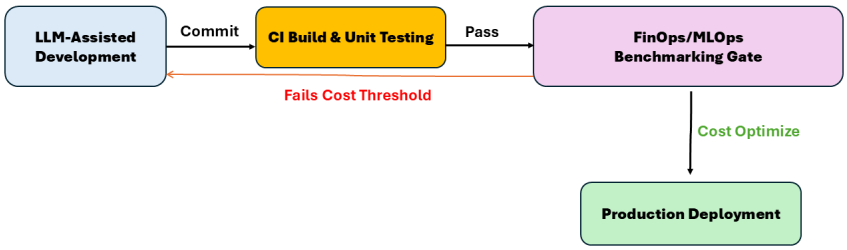


Figure 4. CI pipeline with automated MLOps and FinOps gates to prevent cloud spending inflation.

Table 6. Cognitive Resilience Matrix for HCS Environments.

Cognitive Risk	AI Cause	Mitigation Strategy
Loss of debugging skills	LLMs abstract stack traces.	"AI-off days" and Chaos Engineering.
Erosion of reasoning	Over-reliance on AI design.	Reverse code reviews with human explanations.
Automation complacency	Blind trust in AI accuracy.	Synthetic bug injection to test vigilance.

qualitative research design based on an **Ethnographic Multiple Case Study** [Runeson and Höst 2009, Yin 2018].

3.1. Research Questions and Theoretical Propositions

Aligned with Yin's (2018) guidelines for case study research, we formulate the following Research Questions (RQs) and testable Propositions (Ps):

- **RQ1:** How do "Vibe Coding" practices alter the distribution of cognitive load and debugging effort between Brazilian software engineers and LLM assistants?
- **RQ2:** How do local regulatory constraints (LGPD) and cloud cost structures (FinOps) influence the adoption of cognitive safeguards in software teams?
- **Proposition 1 (P1):** Organizations prioritizing fast time-to-market over structured validation accumulate silent architectural debt due to weak human-AI cognitive coupling.
- **Proposition 2 (P2):** Legal and financial regulatory maturity acts as a catalyst for cognitive resilience by enforcing mandatory human validation in the deployment pipelines.

3.2. Case Selection and Participant Profile

This research adopts a *Theoretical Replication* strategy [Yin 2018] by selecting three contrasting cases within the Brazilian IT sector:

1. **Case A (Agile Startup):** A SaaS provider with no formal AI policies. Highly focused on speed. Target sample: 2 teams ($N \approx 12$ developers).
2. **Case B (Nearshoring Provider):** An IT exporter utilizing metric-driven developer productivity standards. Target sample: 2 teams ($N \approx 16$ developers).
3. **Case C (Financial Institution):** A banking corporation subjected to strict LGPD audits and Central Bank compliance. Target sample: 2 platform engineering teams ($N \approx 15$ developers).

The total target sample size across cases is $\mathbf{N = 43}$ **participants**.

3.3. Data Triangulation and Qualitative Analysis

Data will be gathered from three independent streams to ensure construct validity: (1) semi-structured interviews probing developer habits [Lambiase et al. 2025] and trust [Alami and Ernst 2025], (2) digital shadowing of AI-assisted code reviews to observe interaction bottlenecks, and (3) artifact analysis mining commit histories for SBR metrics. Qualitative transcripts will undergo Miles and Huberman's (2014) two-cycle thematic analysis [Runeson and Höst 2009], using an editing approach to derive open, axial, and selective codes.

3.4. Threats to Validity and Mitigation Strategies

Table 7 presents our systematic mitigation framework for methodological threats based on Runeson and Höst's guidelines [Runeson and Höst 2009].

Table 7. Methodological Validity Threats and Mitigations.

Threat Type	Proposed Mitigation Strategy
Construct Validity	Triangulation of multiple data sources (interviews vs. shadowing vs. repositories) and participant review of transcribed texts (Member Checking).
Internal Validity	Pattern matching against theoretical propositions and active search for negative cases (e.g., developer seniority or baseline language proficiency).
External Validity	Multi-case design spanning startups, outsourcing, and banking, enabling analytical generalization to the broader software engineering ecosystem.
Reliability	Development of a formal, under-version-control Case Study Protocol and preservation of data in an anonymized Replication Package.

3.5. Ethical Considerations and LGPD Compliance

The study will adhere to Brazilian national ethics regulations. All participants will sign an Informed Consent Form (TCLE) guaranteeing total confidentiality and the right to withdraw at any time. To comply strictly with the Brazilian General Data Protection Law (LGPD, Art. 46) [Brasil 2018], all harvested source code artifacts,

logs, and commit histories will be scrubbed to remove Personally Identifiable Information (PII) and proprietary algorithmic business logic before analysis. The study will await formal institutional Research Ethics Committee (CEP) approval before execution.

3.6. Execution Timeline and Exequibility

To address exequibility concerns, the study is scheduled across a 12-month period as detailed in Table 8.

Table 8. Multiple Case Study Research Timeline.

Research Phase	Key Activities and Deliverables	Timeline
Phase 1: Setup	Institutional CEP approval; recruitment of case companies; NDA signing; Case Study Protocol finalization.	Months 1- 2
Phase 2: Collection	Semi-structured interviews ($N = 43$); digital shadowing sessions; commit history and repository mining.	Months 3- 7
Phase 3: Analysis	Transcribing interviews; thematic coding (Miles & Huberman); cross-case synthesis; metric calculations.	Months 8- 10
Phase 4: Reporting	Member checking workshops with participants; final report drafting; SBC template preparation.	Months 11- 12

4. Expected Results and Discussion

As this paper presents a **concept vision and research proposal**, the potential outcomes discussed here represent our **theoretical hypotheses to be tested empirically** in the Brazilian IT sector. We hypothesize that:

- **Hypothesis 1 (H_1):** Startup environments (Case A) fall into Maturity Level 1 (Reactive), where vibe coding accelerates delivery times but induces high Syntax Brittleness Rates (SBR) and invisible architectural technical debt.
- **Hypothesis 2 (H_2):** Nearshoring companies (Case B) operate at Maturity Level 2 (Managed), using prompt guidelines to mitigate immediate security vulnerabilities, but demonstrating weak Cognitive Coupling (CCI) leading to long-term de-skilling of junior engineers.
- **Hypothesis 3 (H_3):** Regulated environments (Case C) naturally catalyze Maturity Level 3 (Dynamic) capabilities, as compliance with LGPD and strict MLOps/FinOps policies organically enforces human-in-the-loop validation and cognitive resilience.

Acknowledgments. The authors thank the supporting funding agencies and professors. We also acknowledge using Grammarly, DeepL, and ChatGPT strictly for grammatical revision, translation, and stylistic refinement. These tools supported linguistic presentation while the authors maintained full intellectual authorship and integrity of the content.

References

Abrahão, S., Grundy, J., Pezzè, M., Storey, M.-A., and Tamburri, D. (2025). Software engineering by and for humans in an ai era. *ACM Transactions on Software Engineering and Methodology*, 34.

Alami, A. and Ernst, N. (2025). Human and machine: How software engineers perceive and engage with ai-assisted code reviews compared to their peers.

Bainbridge, L. (1983). Ironies of automation. *Automatica*, 19(6):775–779.

Brasil (2018). Lei nº 13.709, de 14 de agosto de 2018. lei geral de proteção de dados pessoais (lgpd). Acessado em: 23 jun. 2026.

Carlini, N., Tramer, F., Wallace, E., Jagielski, M., Herbert-Voss, A., Lee, K., Roberts, A., Brown, T., Song, D., Erlingsson, U., Oprea, A., and Raffel, C. (2021). Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2633–2650. USENIX Association.

Cinkusz, K., Chudziak, J. A., and Niewiadomska-Szynkiewicz, E. (2025). Cognitive agents powered by large language models for agile software project management. *Electronics*, 14(1).

Derner, E., Batistič, K., Zahálka, J., and Babuska, R. (2024). A security risk taxonomy for prompt-based interaction with large language models. *IEEE Access*, PP:1–1.

Dhole, K. (2023). Large language models as sociotechnical systems.

Endsley, M. R. (2017). From here to autonomy: Lessons learned from human–automation research. *Human Factors*, 59(1):5–27.

Gadde, A. (2025). Democratizing software engineering through generative ai and vibe coding: The evolution of no-code development. *Journal of Computer Science and Technology Studies*, 7:556–572.

Kreuzberger, D., Kühl, N., and Hirschl, S. (2023). Machine learning operations (mlops): Overview, definition, and architecture. *IEEE Access*, 11:31866–31879.

Lambiase, S., Catolino, G., Palomba, F., Ferrucci, F., and Russo, D. (2025). Investigating the role of cultural values in adopting large language models for software engineering. *ACM Transactions on Software Engineering and Methodology*, 35.

Lundberg, S. M. and Lee, S.-I. (2017). A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Meske, C., Hermanns, T., Weiden, E., Loser, K.-U., and Berger, T. (2025). Vibe coding as a reconfiguration of intent mediation in software development: Definition, implications, and research agenda.

Mäkinen, S., Skogström, H., Laaksonen, E., and Mikkonen, T. (2021). Who needs mlops: What data scientists seek to accomplish and how can mlops help? *IEEE Software*, 38(2):109–115.

Nasir, B. and Hussain, I. (2024). Automating agile workflows: The role of multi-agent llms in modern software engineering.

Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). "why should i trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144.

Ribeiro, M. T., Singh, S., and Guestrin, C. (2018). Anchors: High-precision model-agnostic explanations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Runeson, P. and Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, 14(2):131–164.

Salem, N., Hudaib, A., Al-Tarawneh, K., Salem, H., Tareef, A., Salloum, H., and Mazzara, M. (2024). A survey on the application of large language models in software engineering. *Computer Research and Modeling*, 16:1715–1726.

Tadi, S. R. C. C. T. (2025). Developer and llm pair programming: An empirical study of role dynamics and prompt-based collaboration. *International Journal of Advanced Research in Science Communication and Technology*, 5:436–444.

Teece, D. J. (2007). Explicating dynamic capabilities: the nature and microfoundations of (sustainable) enterprise performance. *Strategic Management Journal*, 28(13):1319–1350.

Viswanadhapalli, V. (2024). Ai-augmented software development: Enhancing code quality and developer productivity using large language models. pages 2456–4184.

Yin, R. K. (2018). *Case study research and applications: Design and methods*. Sage publications, 6th edition.

Zou, C. (2024). Sociotechnical risks of simulating humans with language model agents.