Evaluating Large Language Model Reasoning on Nested Attribute Constraint Resolution Problems

Krishna Ronanki

August 19, 2025

1 Introduction

Many state-of-the-art LLMs are demonstrating increasingly impressive capabilities in performing a wide range of tasks [2], including software engineering (SE), and are observed to increase user productivity [6]. The adoption of LLMs in industrial settings is observed to be growing thanks to easy-to-use interfaces that make the LLMs accessible to practitioners [7].

However, there are a few impediments in using LLMs for SE in practice such as confidential data privacy, extrinsic hallucinations [1], and the lack of best practices [3]. Third party LLM-based solutions such as Azure OpenAI services, Microsoft Copilot, GitHub Copilot, ChatGPT enterprise aim to mitigate some of these issues.

However, that still leaves the shortage of best practices for using LLMs for SE in practice as an open challenge, pointing towards the need for more empirical research on LLM-based SE [3]. Moreover, the use of LLMs in practice will also need to be compliant with the newly ratified EU AI Act (AIA) [5] guidelines. These two challenges combined together helped in formulating the research problem I'm addressing:

Users and organisations lack actionable recommendations to leverage LLMs to assist them in their SE activities in an AIA-compliant manner.

My research's goal is to explore how LLMs are being adopted within industrial software development environments and what can be learnt from the experiences of early adopters. The insights presented based on either secondary evidence-based studies [4] (non-empirical) or controlled empirical experiments [8] do not fully capture the nuances of adopting LLMs in industrial settings. Hence, we see a clear need for an empirical study that focuses on capturing practitioner's experiences regarding the adoption of LLMs within industrial SE environments.

To do this, I'm conducting various empirical studies such as multi-case studies and case meta-syntheses of published industrial cases. The aim is to identify the benefits, challenges, and lessons learnt from these adoption efforts, understanding how the adoption of LLMs in software development aligns with established broader AI adoption frameworks.

Another key part of my research focuses on making the recommendations developed from these cases help operationalise trustworthy AI principles of the AIA [5] into actionable practices for software practitioners.

2 Lecture Principles

2.1 AI Engineering

The lecture on SE4ML and the AI Engineering manifesto highlight the challenges and principles that distinguish AI/ML system development from traditional SE. From the lecture, a key takeaway is the ML Technology Readiness Level (ML-TRL) framework, which presents the journey from basic research and prototyping through to production, deployment and long-term maintenance. At each stage, additional concerns such as explainability, compliance, monitoring, drift management, and retraining become central. This perspective is highly relevant to my empirical studies of industrial LLM adoption, as it provides a lens through which different organisations' maturity and adoption experiences can be compared.

The AI engineering manifesto, developed as a result of the group discussions during the course, builds on these ideas by discussing some principles for AI engineering, including reproducibility, safety and ethics, data-centric design, continuous monitoring, simplicity and transparency. These principles are relevant to my research, as they provide a theory-informed framework that can be operationalised in my case studies. For example, evidence of how organisations implement traceability, fairness checks, or continuous monitoring can be coded against these principles, enabling systematic comparison across cases. Moreover, the manifesto's emphasis on transparency, explainability, and accountability connects directly to the broader goal of operationalising trustworthy AI principles, such as those outlined in the AIA, into actionable practices.

2.2 Behavioural SE

Behavioural SE (BSE) highlights human cognition and behaviour has impact on SE practices. One of the key insights is that engineers are not fully rational decision-makers and adoption choices are often influenced by cognitive biases such as availability bias, anchoring bias, confirmation bias, stereotype bias, and similarity bias.

This perspective is directly relevant to my research on LLM adoption in industry. For example, availability bias may cause organisations to overvalue widely publicised LLM success stories, while anchoring bias may lock teams onto early pilot results. Such biases can distort how benefits, risks, and long-term sustainability are evaluated during adoption.

BSE also identifies mitigation strategies hat can reduce the impact of bias such as building chains of evidence or generating multiple solutions. I intend to investigate whether and how organisations apply them when adopting LLMs, and what role they play in shaping more trustworthy practices. This enables me to go beyond technical and organisational challenges to also capture how bias-aware practices affect the quality and trustworthiness of adoption decisions.

3 Guest-Lecture Principles

3.1 Requirements Engineering

Requirements Engineering (RE) stresses the importance of clarifying the problem space: stakeholder needs, goals, and the constraints, before committing to solutions. This is relevant to my study of LLM adoption, where organisations experiment with tools without clearly articulating the underlying requirements. Applying RE concepts such as stakeholder elicitation and goal modelling can help structure my empirical analysis, revealing

whether adoption efforts are aligned with usage goals (e.g., developer productivity), system goals (e.g., trustworthiness, compliance), and business goals (e.g., competitiveness). By framing adoption in requirement-driven rather than technology-driven terms, RE provides a lens to identify mismatches between expectations and outcomes, and supports my aim of translating trustworthy AI principles into actionable practices for industry.

3.2 Human Aspects of SE4AI

The human aspects of Software Engineering for AI (SE4AI) highlight that successful AI system adoption depends not only on technical robustness but also on how engineers, teams, and organisations interact with the technology. Issues such as trust, interpretability, collaboration across roles, and the ability to manage uncertainty are central to effective integration. This perspective aligns with my research on LLM adoption in industry: understanding how practitioners perceive, trust, and adapt to LLMs is as important as assessing their technical performance. By focusing on human factors—such as how biases influence decision-making, how trust and accountability are negotiated in teams, and how organisational culture shapes adoption, I can understand why certain practices succeed or fail in real settings. Thus, incorporating the human aspects dimension of SE4AI strengthens my goal of developing recommendations that not only address technical challenges but also support practitioner needs and foster trustworthy adoption of LLMs.

4 Data Scientists versus Software Engineers

The book fairly characterises data scientists as model-centric (stats/ML background, exploratory workflows, accuracy-first evaluation) and software engineers as product-centric (requirements, architecture, testing, operations). These are real, observable differences in training, incentives, and day-to-day practice. The authors also flag that this is an oversimplification, that "unicorns" exist, and that the key is complementary expertise within one product team.

Chapters 1–2 emphasise interdisciplinary teams and T-shaped people rather than turning everyone into the same role. At the same time, Chapter 2 shows the expanding surface area around models—pipelines, deployment, monitoring, and systems integration—which pushes deeper specialisation in MLOps/platform, data/label operations, and safety/quality, while also demanding broader literacy across the boundary so teams can reason about system-level goals and not just model accuracy. Foundation models shift some work from custom training to prompting/chaining/RAG, but that increases systems thinking and integration trade-offs rather than eliminating role depth.

5 Paper analysis

Paper-1: Developer Experiences with a Contextualised AI Coding Assistant: Usability, Expectations, and Outcomes **Core ideas & SE importance**: Contextualised assistants beat generic ones in enterprise development by injecting organisation specific context (internal APIs, docs, code) via retrieval/grounding; developers report increased productivity, faster retrieval, and better internal-API usage, though responses can be variable and complex tasks remain challenging.

Relation to my research: The contributions of this work are aligned with my focus on real-world LLM adoption: it documents early adopter experiences, perceived benefits (productivity, discoverability) and frictions (context quality, answer variability),

giving constructs and measures to look for in case studies (e.g., how teams govern context sources, handle variance, and set expectations).

Integration into a larger AI-intensive project: A hypothetical AI-intensive software project, where AI/ML is a core aspect of the system's functionality could be a platform that embeds a coding copilot across various microservice repositories with organisation wide RAG over API specs. Using the paper's ideas, one approach would be to prioritise context pipelines (indexing internal sources, freshness policies) and evaluation with users around accuracy for internal APIs and perceived time savings.

Adaptation of my research: I integrate the evaluation methods from the paper—surveys, interviews, and interaction logging, into my multi-case studies and case meta-syntheses to systematically capture developers' experiences with LLM adoption. This allows me to examine workflow fit, perceived benefits, and AI response quality across industrial teams, while grounding my empirical analysis in established methodological practices without replicating their engineering focus.

Paper-2: Towards a Responsible AI Metrics Catalogue: A Collection of Metrics for AI Accountability Core ideas & SE importance: Proposes a metrics catalogue for AI accountability spanning process, resource, and product metrics, distilled via a multivocal review, with emphasis on GenAI. It operationalises high-level responsible AI principles into measurable checks teams can adopt across the lifecycle, bridging "principles → practice" for engineering and governance.

Relation to my research: My aim to operationalise trustworthy AI (AIA) principles aligns well as the catalogue is a ready coding frame for what organisations actually measure (or do not). You can assess which accountability metrics early adopters use (e.g., audit trails, incident logs, red-team coverage), and how these correlate with adoption outcomes.

Integration into a larger AI-intensive project:

Consider a fictional large-scale platform designed to embed generative AI across an enterprise for tasks such as document summarisation, customer support, and code assistance. The responsible AI metrics catalogue could be adopted here to establish measurable indicators of accountability at different lifecycle stages, covering process, resource, and product metrics (e.g., version control of prompts and models, incident reporting, harmful-output tracking). However, the current catalogue is more accountability-oriented than fully comprehensive in relation to the seven Trustworthy AI (AIA) principles.

Where my research fits? Evaluate metric adoption maturity across the platform by identifying gaps, barriers, and enablers and to link metrics to trust and deployment decisions.

Adapting mu research: For my own research, the idea of extending the catalogue to cover all AIA principles offers a valuable path forward. Instead of focusing only on accountability, I could investigate how industrial adopters of LLMs are (or are not) addressing the full set of principles through measurable practices. For example, in case studies I might ask: Which AIA principles are explicitly translated into metrics? Where do organisations lack measurable proxies (e.g., societal impact, fairness)? How do teams balance trade-offs between principles (e.g., robustness vs. efficiency, fairness vs. productivity)?

Long-term, my research could contribute to building a comprehensive, AIA-aligned metrics framework that organisations can apply to their LLM adoption efforts.

6 Research Ethics & Synthesis Reflection

Search and Screening Process: I accessed the CAIN conference website and manually reviewed the list of accepted papers. I screened papers based on their titles, identifying those that appeared relevant to my research question. For each potentially relevant paper, I located it on Google Scholar and read its abstract. If the abstract indicated relevance, I then read the full paper and extracted the key points necessary to answer **Section 5**.

Ethical Considerations: I ensured originality by summarising the content in my own words rather than copying from papers or using AI-generated content. LLMs were only used to improve the clarity and readability of the text I wrote, without introducing external information or altering the substance of the findings.

References

[1] Bang, Y., et al. A Multitask, Multilingual, Multimodal Evaluation of ChatGPT on Reasoning, Hallucination, and Interactivity. *arXiv preprint arXiv:2302.04023* (2023).

[2] Brown, T., et al. Language Models are Few-shot Learners. *Advances in neural information processing systems 33* (2020), 1877–1901.

[3] Fan, A., Gokkaya, B., Harman, M., Lyubarskiy, M., Sengupta, S., Yoo, S., and Zhang, J. M. Large Language Models for Software Engineering: Survey and Open Problems. In *2023 IEEE/ACM International Conference on Software Engineering: Future of Software Engineering (ICSE-FoSE)* (2023), pp. 31–53.

[4] Kitchenham, B. A., Dyba, T., and Jorgensen, M. Evidence-based Software Engineering. In *Proceedings. 26th International Conference on Software Engineering* (2004), IEEE, pp. 273–281.

[5] Parliament, E., and European Union, C. o. t. "Artificial Intelligence Act", 2024. Accessed: 06/03/2025.

[6] Peng, S., et al. The Impact of AI on Developer Productivity: Evidence from GitHub Copilot. *arXiv preprint arXiv:2302.06590* (2023).

[7] White, J., et al. ChatGPT Prompt Patterns for Improving Code Quality, Refactoring, Requirements Elicitation, and Software Design, 2023.

[8] Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., and Wesslén, A. *Experimentation in software engineering*. Springer Science & Business Media, 2012.