

PART 2:

LLMs for Software Engineering

State of the Art & Open Problems

Aug 28, 2025

Hou et al. (2024) & Fan et al. (2023)




Discussion Roadmap

1. Optimization & Hybridization
2. Evaluation & Non-determinism
3. Tasks & Future Directions
4. Hallucinations: reducing techniques
5. Student-submitted questions
6. Breakout room activity: brainstorming possible project ideas.

Optimization, Prompting, and Hybridization (10 min)

- When would you choose Prompt engineering (few-shot, COT) vs. PEFT (LoRA, adapters, fine-tuning).
- Hybrid pipelines: What's the most promising way to combine LLMs with traditional SE tools (static analysis, testing, formal methods)?

Prompt Engineering vs. PEFT in SE Tasks

-  Prompt Engineering (Few-shot, CoT, ReAct)
 - Strengths: Cheap, flexible, fast for prototyping
 - Weaknesses: Less stable, long prompts, non-deterministic
 - SE Examples: Requirements ambiguity detection, design brainstorming, doc generation, commit messages
-  PEFT (LoRA, Adapters, Prefix Tuning)
 - Strengths: Stable, domain-specific, efficient storage
 - Weaknesses: Needs curated data, narrower task focus, setup effort
 - SE Examples: Code completion for proprietary APIs, program repair, vulnerability patching, CI test generation
-  **Rule of Thumb: Use Prompting for exploration/creativity; shift to PEFT for scale, correctness, and domain-specific reliability.**

Evaluation, Non-Determinism, and Ground Truth (10–12 min)

- Are current metrics (BLEU, Pass@k, F1, MRR) missing key dimensions (security, maintainability)?
- Unlike natural language, code can be executed/tested. Should execution-based validation be mandatory?
- Non-determinism: Only ~20% of surveyed papers treat non-determinism as a threat. Should standards require multiple runs?

YourQ: How to achieve the trustworthiness required for deploying LLM-generated code in mission-critical systems? How to assess deeper qualities like efficiency and security? What level of verification from such a system would be required for confident adoption?

Tasks, Domains, and Future Directions

(15 min)

- Current focus: ~71% generation, ~57% dev tasks. What would it take to push LLMs into requirements/design/management?
- Where do you see the next big win for LLM4SE (program repair, test generation, traceability)?
- How can SE methods (testing, maintainability) help build better LLMs (SE4LLM)?
- If you could extend these surveys in one year, what new themes do you expect to explode?

Hallucinations in LLM4SE: The Challenge

- LLMs generate plausible but incorrect code/specs.
- High-risk in security patches, effort estimation, API migration.
- When are hallucinations tolerable vs. unacceptable?

Reducing Hallucinations: Techniques & Hybrid Approaches

- Static analysis: type safety, detect inconsistencies.
- Dynamic analysis: unit tests, fuzzing, execution traces.
- Search-based: multiple candidates, filter via constraints.
- Hybrid workflows: LLM + checks + human-in-the-loop approval.

Your Miscellaneous Questions

The Human-Developer Interaction and the Future of SE Roles

- As the software engineer's role evolves due to AI integration, what are the concurrent psychological impacts on developers and the long-term risks of skill atrophy, and how can we proactively design optimal human-in-the-loop workflows and truly collaborative IDEs that augment developer expertise rather than simply replacing it?

Bridging the Gap Between Research and Industrial Practice

- What practical strategies and model transferability benchmarks are needed to bridge the gap between academic research on clean, open-source data and the complex reality of industrial software engineering, considering the primary adoption barriers of model size, inference latency, and the costs of adapting to proprietary, mixed-language monorepos?

Your Miscellaneous Questions – part 2

Governance, Liability, and Regulation

- For high-stakes regulated industries, what would a comprehensive governance framework entail, integrating both clear legal models for liability and accountability when AI-generated code causes harm, alongside the necessary technical and procedural mechanisms—such as code lineage tracking and automated audits—required to ensure and prove compliance?

Prompting and Fine-Tuning Techniques

- To move beyond ad-hoc methods of controlling LLMs, what are the precise technical distinctions, costs, and performance trade-offs between prompt engineering and Parameter-Efficient Fine-Tuning (PEFT), and is it feasible to develop a formal, standardized framework to make the prompt creation process more deterministic and reliable?

Your Miscellaneous Questions – part 3

Hallucinations: A Risk or an Opportunity?

- How can development teams manage the duality of LLM hallucinations by establishing criteria to identify rare, beneficial suggestions (like novel test cases) while simultaneously implementing robust training and evaluation strategies to mitigate the significant risk of developers blindly accepting harmful, bug-inducing fabrications?

Critique of the Systematic Literature Review (SLR) Methodology

- How can the Systematic Literature Review methodology be modernized to improve its comprehensiveness, both by replacing questionable heuristics like page-length filtering with advanced AI-driven discovery methods like vector search, and by adapting its scope to formally include innovation from non-traditional research sources like industry blogs and open-source forums?

Breakout Activity: Underserved SE Domains

- Domains: Requirements, Design, Refactoring, Management.
- Step 1 – Pick a task in your assigned domain.
- Step 2 – Propose 3 evaluation metrics.
- Step 3 – Address hallucinations in your domain.
- Step 4 – Address non-determinism in your domain.
- Step 5 – Share a 1-sentence summary of task + metrics.