GenAI-powered Software Engineering

CSCI 7000-011
Tue/Thu @2pm

Danny Dig

Today’s goals

Review what we covered in GenAI for Software Engineering: practice and research examples

Revisit your expectations from the class

What is NEXT?

Overview of 24 research papers

Revisit topics we covered

Foundations of GenAI for SE:

- major surveys of LLMs-for-SE and SE-specific LLM capabilities
- knowledge injection
- prompting effectiveness, impact of fine-tuning

Deep dives:

- LLM-based refactoring (ExtractMethod, MoveMethod, ExtractClass),
- code migration at Google
- test generation (TestSpark)
- automatic program repair (AutoCodeRover, RepairAgent)
- debugging (NIODebugger)
- documentation updates
- code generation / inefficiency taxonomies, AI Coding assistants
- agents (CoRenameAgent, RepairAgent, Mantra, effort estimation), human-AI collaboration

What was your main takeaway from the research papers we covered?

1. LLMs as Core SE Infrastructure

Survey & SLR papers position LLMs as foundational for many SE tasks downstream

KEY Idea: LLMs are not one-off tools; they're treated as **core infrastructure** that underpins many downstream tasks (generation, repair, refactoring, documentation, estimation, etc.)

Hybrid techniques (LLMs + traditional SE) are essential for reliable, efficient workflows.

Emergent properties enable creativity but raise challenges around control and correctness.

2. Code Transformation & Refactoring at Scale

Focus on Extract/Move Method, Extract Class, migrations, and code-change patterns (CPATs).

Tools: PyCraft, EM-Assist, MM-ASSIST, MANTRA, HECS, and RefactoringMiner.

LLMs provide "expert" suggestions; static/dynamic analysis and IDEs enforce safety and mechanics.

Goal: align automated refactorings with developer intent while scaling to large codebases.

KEY Idea: **synergy between LLM “intuition” and traditional static/dynamic analysis** to make refactoring trustworthy

3. APR, Debugging & Flaky Tests

AutoCodeRover, RepairAgent, layered-context APR, and NIODebugger target real-world bugs.

LLM-based agents plan, search, call tools, and validate fixes using tests and spectra.

KEY Idea: Layered knowledge (bug, repository, project) significantly boosts fix rates.

Special focus on NIO flaky tests and state pollution, combining dynamic analysis with LLM reasoning.

4. Testing and Quality of LLM-Generated Code

TestSpark integrates search-based and LLM-based test generation directly in the IDE.

Taxonomy of inefficiencies highlights logic, performance, readability, and maintainability issues.

Prompting strategies (Few-Shot, CoT, CoT+Few-Shot) clearly outperform Zero-Shot at class level.

KEY Idea: Emphasis on both functional correctness and long-term code quality metrics (e.g., BLEU, ROUGE).

5. Agents, Collaboration & Human-in-the-Loop

SWE agents and in-IDE assistants support multi-step, interactive development workflows.

KEY Idea: Empirical studies show higher success when developers iterate and collaborate with agents.

Multi-agent frameworks coordinate roles (planner, critic, executor) and negotiate with humans.

Design goal: agents that challenge, explain, and co-create rather than act as sycophantic oracles.

6. Beyond Coding: Documentation & Estimation

LLMs update API documentation based on code changes, improving similarity to human edits.

Multi-agent LLM frameworks support agile effort estimation and consensus-building in teams.

Surveys reveal where developers already delegate work (tests, docs) to AI assistants.

Non-coding tasks become a major frontier for LLM support in the SE lifecycle.

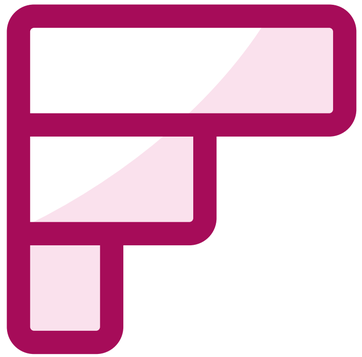
KEY Idea: LLMs aren't just for code generation—they're being pushed into **documentation, planning, estimation, and analytics**.

7. Reliability, Safety & Security

Hallucinations in refactoring suggestions are filtered via static analysis and self-consistency checks.
- applying LLM-suggested changes using trusted engines to avoid unsafe edits.

Backdoor-attack study shows Code LLMs are vulnerable even at extremely low poisoning rates.

KEY Idea: Trust, policies, defensive techniques are critical for safe deployment of LLM-based tooling.



Rank the topics you found most interesting/useful/valuable.

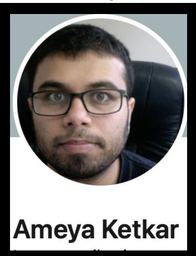
Invited Speakers: Thought leaders from Industry & Academia

Code, Critique, Cure: Advancing LLM Reasoning for AI-Augmented Software Maintenance

ICSME 2025 Auckland, New Zealand

From Tools to Teammates: Generative AI as Collaborative Partners in Industrial Workflows

intel



The Landscape of AI-assisted Automated Software Engineering

gitar

Advancing LLM Intelligence: Uniting Internal Reasoning and External Tool Interactions

NEC

Your growth through research papers

You read 24 research papers

Paper Critiques: **Equipped you with critical thinking**
- learned to evaluate research claims, empirical methods, and limitations in modern GenAI+SE research

Research paper presentation: you prepared, delivered, and led class discussion
Equipped you to communicate and connect with the audience

Your growth through hands-on projects

Exercised the whole lifecycle of open-ended, risky project: pitching ideas, forming teams, design, implement, evaluate, and present a novel GenAI SE system

Four awesome research projects:

- Project-Aware Local Variable Renaming with LLM
- RepoGraph-NIO: Verifiable LLM-Driven Repair of Order-Dependent Tests
- Prioritization for Agile Planning in SWE
- GenEC: Extract Class with GenAI

Equipped you to lead novel R&D

How confident do you feel about using LLMs in your projects after this course?

What are your expectations from CSCI 7000-011: GenAI for SE?

- [x] A. Do a cool GenAI Project that helps programmers/society
- [x] B. Learn about exciting GenAI Applications for SE
- [x] C. Exposure to research on GenAI for SE
- [x] D. Learn about the Challenges in building GenAI software
- [x] E. Learn how GenAI software differs from classical software
- [x] F. How to leverage the strengths of each tool
- [x] G. How academia changes to adapt to AI
- [x] H. Best practices and ethical use of these tools
- [x] I. How to make scalable GenAI systems

What is one expectation you had from this course that you feel was met?

You grew so MUCH

Research papers & critiques

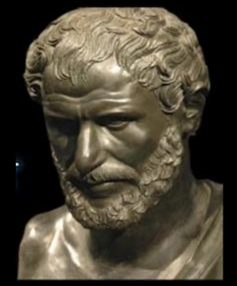
Presentations

Hands-on project work

How satisfied are you with your learning experience in this course?

Summary and What’s Next

Change is the only guaranteed constant



“AI **will not** displace programmers. But programmers who use AI **will** displace programmers who do not use AI.” – Prof Danny Dig

Together We Continue to Go Further

CSCI 7000-005: Agentic Software Systems
- build an AI Agent that you can showcase in your portfolio

CSCI 7000-006: AI-Driven Leadership
- use AI to augment your ability to communicate, lead with values, and people skills