ISSN 2186-7437

NII Shonan Meeting Report

No. 229

A Tale of Two Cities: Large Language Models and Computer Programming

Yuekang Li
Yulei Sui
Yang Cao
Ling Shi

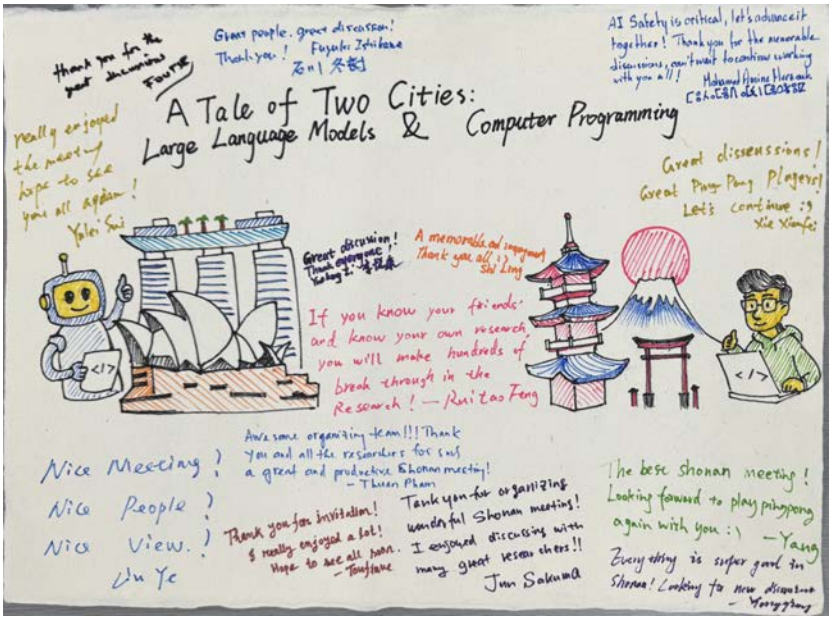
December 8–11, 2025

National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-Ku, Tokyo, Japan

A Tale of Two Cities: Large Language Models and Computer Programming

Organizers:
Yuekang Li (UNSW)
Yulei Sui (UNSW)
Yang Cao (Science Tokyo)
Ling Shi (NTU)

December 8–11, 2025



Abstract

In recent years, the proliferation of Large Language Models (LLMs) has captivated the world with their remarkable language generation, comprehension capabilities, and their transformative impact across various domains. These models, typified by examples like GPT (Generative Pre-trained Transformer) series, Bard, and Llama series, have ushered in a new paradigm in natural language processing, enabling machines to comprehend, generate, and manipulate human-like text at an unprecedented scale and quality.

The emergence of Large Language Models (LLMs) has catalyzed a paradigm shift in software engineering, programming languages, and software security. This Shonan Meeting aims to explore the intricate interplay

between LLMs and computer programming by delving into their pivotal role in advancing software engineering and security practices, addressing programming language problems, and tackling the challenges associated with securing and analyzing LLM applications/frameworks. The meeting serves as a collaborative platform, inviting experts from diverse backgrounds to dissect the multifaceted landscape where LLMs intersect with programming, fostering knowledge exchange and innovation in response to evolving complexities.

Background and Introduction

This report provides a structured summary of the NII Shonan Meeting No. 229, held from 8–11 December 2025. The meeting brought together international experts in software engineering, security, programming languages, and AI to examine the rapidly evolving influence of Large Language Models (LLMs) on software development, verification, and trustworthiness. The program featured invited talks, panel discussions, and collaborative exchanges designed to foster future research directions and community building.

This Shonan Meeting aims to bring together participants from a broad spectrum of fields including but not limited to computer security, software engineering, formal verification, programming language, and AI. Therefore, the interdisciplinary nature of the meeting is expected to yield outcomes such as:

- Innovative Methodologies: Cross-disciplinary collaboration fostering new methods and techniques leveraging LLMs in program analysis, software security, and programming language design.
- Enhanced Security Practices: Combining insights from various fields to develop robust security practices for LLM applications. Exploring innovative techniques to fortify LLM applications and frameworks, addressing new complexities introduced by LLMs.
- Comprehensive Frameworks: Formulating comprehensive frameworks for analyzing and securing LLM-based applications, integrating theoretical and practical perspectives.

Overview of the Meeting

The four-day meeting was structured to balance technical depth with open-ended discussion, fostering both focused knowledge exchange and broader community building. The meeting included 17 participants from Australia, Japan, Singapore, Canada, and the United States, as listed in the List of Participants section. Each day combined invited research talks with panel discussions that encouraged cross-disciplinary dialogue on the role of large language models in programming, verification, and software security. Lightning introductions at the beginning of the meeting helped participants rapidly share their backgrounds and interests, setting the stage for collaborative engagement throughout the meeting. Beyond formal sessions, extended breaks, shared meals, and a mid-meeting excursion provided opportunities for informal discussions, enabling participants to explore emerging research directions, identify common challenges, and initiate potential collaborations in a collegial and interactive environment.

Overview of Talks

Large Language Models for Code Tasks: From Performance Improvement to Robustness Analysis

Xiaofei Xie
Singapore Management University, Singapore

Large language models (LLMs) have demonstrated remarkable capabilities across domains, particularly excelling in code generation and text analysis. This talk focuses on the practical application of LLMs in code correctness analysis, bug fixing, and code review–driven modification. We explore how LLMs can be used to identify inconsistencies between specifications and implementations, and how intent analysis combined with multi-round verification mechanisms can effectively enhance the quality and reliability of code generation tasks. Through empirical studies, we observe that an LLM's understanding of program semantics plays a crucial role. Notably, many seemingly successful fixes rely more on the model's memorisation ability than on genuine semantic comprehension. These findings highlight the importance of introducing more rigorous and robust evaluation methodologies when leveraging LLMs for code generation tasks.

Toward Trustworthy Code Generation: A Specification-Guided Self-Examination Framework for LLMs

Yuxiang Shang
Hiroshima University, Japan

Large language models show impressive code generation ability, yet their outputs often contain logical errors and hallucinations. This talk presents SOFL-TestCoT, a specification-guided framework that improves the reliability of LLM-generated code. The method uses a simple SOFL formal specification to guide structured reasoning, derive test–oracle pairs, and drive an iterative self-examination loop based on real code execution. I will outline the core ideas and discuss implications for building trustworthy generative AI systems.

Static Analysis for AI Models

Yulei Sui
University of New South Wales, Australia

Static analysis has long served as an important basis for program reasoning and verification, enabling the extraction and interpretation of abstract representations of software systems. This talk explores how the principles of static analysis, as developed in our SVF framework, can be extended beyond traditional code analysis to areas such as abstract execution of AI models. As part of this ongoing effort, we are developing the ACT tool, which applies static reasoning to the analysis and verification of neural networks. It builds on the shared foundations between program analysis and neural network verification, both grounded in abstraction and symbolic reasoning over large input spaces. Our early findings indicate that static reasoning techniques provide useful insights for improving the reliability and robustness of modern AI systems.

Safety Assurance of CPS in the Era of LLMs

Zhenya Zhang
Kyushu University, Japan

I will introduce our recent progress in advancing safety assurance of cyber–physical systems (CPS) using LLMs. In particular, we aim to involve LLMs in the loop of lightweight formal methods–based safety assurance of CPS. I will also discuss the potential of the reverse direction, namely, applying lightweight formal methods for quality assurance of LLMs.

Large Language Models for Smart Contract Security: From Property Generation to Attack Detection

Ye Liu
Singapore Management University, Singapore

Recent advances in large language models (LLMs) such as GPT-4 have opened new possibilities for automating smart contract security analysis—traditionally a domain requiring deep program reasoning and formal methods expertise. In this talk, I will introduce our recent efforts to apply LLMs to two key challenges in this space: PropertyGPT, which automatically generates verifiable formal properties (e.g. invariants and pre/post-conditions) for smart contracts, and DeFiScope, which detects complex DeFi price manipulation attacks by reasoning over on-chain price models and high-level protocol semantics. Together, these works demonstrate that LLMs can bridge the gap between human reasoning and formal verification, offering both interpretability and automation.

Repairing Formal Models: By Formal, LLM, and More?

Fuyuki Ishikawa
National Institute of Informatics, Japan

Supporting the use of formal methods with LLMs is a promising direction. We argue that the focus should shift from generating models to repairing "mostly correct" or "mostly valid" models. While LLMs may be able to perform such repairs using their general reasoning capabilities, formal techniques can provide guarantees about the correctness of the repaired results (when applicable). Related work in search-based automated program repair also offers useful insights. In this talk, I outline this problem landscape and share our experiences as a starting point for discussion in this Shonan meeting.

LLM-Assisted Fuzzing for Effective Security Testing

Thuan Pham
University of Melbourne, Australia

Fuzzing is a widely used security testing approach for discovering vulnerabilities in large, real-world software systems. However, its effectiveness depends on several factors, including the quality of fuzz drivers (i.e. test harnesses), the diversity and relevance of generated inputs, the feedback signals, and the test

oracles. Many of these challenges have traditionally required significant human effort. In this talk, I present our early exploration of using large language models (LLMs) to enhance fuzzing while minimising manual intervention. I also discuss the key challenges encountered and highlight promising directions for future research in this emerging area.

Identifying Multi-parameter Constraint Errors in Python Data Science Library API Documentation

Yi Li (unofficial remote participant)
Nanyang Technological University, Singapore

Modern AI- and data-intensive software systems rely heavily on data science and machine learning libraries that provide essential algorithmic implementations and computational frameworks. These libraries expose complex APIs whose correct usage must follow constraints among multiple interdependent parameters. However, maintaining correct and consistent multi-parameter constraints in API documentation remains a significant challenge. To address this issue, we propose MPChecker, which detects inconsistencies between code and documentation by identifying multi-parameter constraints via symbolic execution and extracting documentation constraints using LLMs. A customised fuzzy constraint logic reconciles the unpredictability of LLM outputs. Experiments on datasets collected from four popular libraries show that MPChecker achieves a precision of 92.8%, with 11 out of 14 reported issues confirmed by developers.

Generating Tests from Issues to Validate SWE Patches

Toufique Ahmed
IBM Research, United States

While there has been extensive work on generating tests from existing code, limited attention has been given to generating tests directly from issue descriptions. This talk focuses on scenarios where the code patch does not yet exist, supporting both test-driven development (TDD) and the validation of software engineering (SWE) agents that automatically generate patches. I will discuss two approaches to automatically generating tests from issue descriptions.

Disrupting Model Merging: A Parameter-level Defence without Sacrificing Accuracy

Jun Sakuma
Institute of Science Tokyo, Japan

Model merging combines multiple fine-tuned models into a single model without additional training, enabling free-riders to cheaply inherit specialised capabilities. This talk presents the first proactive defence against model merging by modifying model parameters so that merging significantly degrades performance while preserving standalone functionality. Our approach rearranges MLP parameters and scales attention heads to move models out of shared parameter

basins. Extensive experiments across vision and text tasks demonstrate strong disruption of merging with minimal loss of utility.

AI for Secure Software: Capabilities, Vulnerabilities, and Alignment for LLM-Driven Systems

Mohamed Amine Merzouk
Mila – Quebec AI Institute, Canada

This talk explores leveraging and securing large language models within modern software systems. We first present retrieval-augmented frameworks that combine static and dynamic program analysis with iterative LLM refinement to detect vulnerabilities and generate trustworthy patches. We then demonstrate how LLMs can unintentionally introduce supply-chain vulnerabilities and enable large-scale malware injection through model poisoning. The talk concludes with a trait-based alignment approach that provides a scalable path toward safer model behaviour in security-critical environments.

Privacy in Fine-tuning for Large Language Models: Attacks, Defences, and Future Directions

Yang Cao
Institute of Science Tokyo, Japan

Fine-tuning is critical for adapting large language models (LLMs) to downstream tasks, but it often involves sensitive datasets that introduce privacy risks. This talk provides a comprehensive overview of privacy challenges in fine-tuning, including vulnerabilities to membership inference, data extraction, and backdoor attacks. We review existing defence mechanisms, analyse their effectiveness and limitations, and discuss open challenges for balancing privacy protection and model utility.

Supercharging Compiler Engineering with LLMs

Yongqiang Tian
Monash University, Australia

Compiler engineering traditionally relies on deep expertise and labour-intensive manual effort. This talk presents recent work demonstrating how large language models (LLMs) can fundamentally accelerate this process by generating candidates for missed peephole optimisations and implementing language-specific transformations to automate debugging workflows, enabling unprecedented gains in productivity and analysis speed.

Correcting Decompilation Distortions with LLMs

Ruitao Feng
Southern Cross University, Australia

Decompilation enables analysis of binaries without source code but often suffers from fidelity distortions such as incorrect variable naming and structural

inconsistencies. This talk introduces FidelityGPT, an LLM-driven framework that detects and corrects such distortions using tailored prompts, retrieval-augmented generation, and a dynamic semantic intensity algorithm. Evaluations show substantial improvements over existing methods, achieving a 94% fix rate on a binary similarity benchmark.

Detecting Bugs in the LLM Ecosystem

Yuekang Li
University of New South Wales, Australia

This talk presents recent work on detecting and analysing bugs across the large language model (LLM) ecosystem, from applications to model behaviour. Topics include black-box prompt-injection attacks, automated detection of remote code execution vulnerabilities in LLM frameworks, and systematic evaluations of jailbreak attacks and defences. I will also introduce Drowzee, a logic-programming–guided metamorphic testing framework for detecting fact-conflicting hallucinations, and discuss implications for building more robust and trustworthy LLM-based systems.

List of Participants

- Yang Cao, Institute of Science Tokyo, Japan
- Yuekang Li, University of New South Wales, Australia
- Ling Shi, Nanyang Technological University, Singapore
- Yulei Sui, University of New South Wales, Australia
- Fuyuki Ishikawa, National Institute of Informatics, Japan
- Jun Sakuma, Institute of Science Tokyo, Japan
- Kailong Wang, Huazhong University of Science and Technology, China
- Xiaofei Xie, Singapore Management University, Singapore
- Ye Liu, Singapore Management University, Singapore
- Van-Thuan Pham, University of Melbourne, Australia
- Yuxiang Shang, Hiroshima University, Japan
- Zhenya Zhang, Kyushu University, Japan
- Mohamed Amine Merzouk, Mila - Quebec AI Institute, Canada
- Foutse Khomh, Polytechnique Montréal, Canada
- Ruitao Feng, Southern Cross University, Australia
- Yongqiang Tian, Monash University, Australia
- Toufique Ahmed, IBM Research United States, USA

Meeting Schedule

Day 1 – Monday, 8 December 2025

Time	Session / Speaker	Title / Description
09:00–09:10	Introduction movie	Introduction movie of NII Shonan Meeting
09:10–09:30	Opening	Opening and Introduction
09:30–10:00	Participants	Lightning introductions
10:00–10:30	Talk – Xiaofei Xie	Large Language Models for Code Tasks: From Performance Improvement to Robustness Analysis
10:30–11:00	Break	
11:00–11:30	Talk – Yuxiang Shang	Toward Trustworthy Code Generation: A Specification-Guided Self-Examination Framework for LLMs
11:30–12:00	Talk – Yulei Sui	Static Analysis for AI Models
12:00–13:30	Lunch	
13:30–14:00	Group photo	Group photo shooting
14:00–14:30	Talk – Zhenya Zhang	Safety Assurance of CPS in the Era of LLM
14:30–15:00	Talk – Ye Liu	Large Language Models for Smart Contract Security: From Property Generation to Attack Detection
15:00–15:30	Talk – Fuyuki Ishikawa	Repairing Formal Models: By Formal Methods, LLMs, and Beyond
15:30–16:00	Break	
16:00–17:30	**Panel Discussion 1** (Moderator: Yulei Sui)	**LLMs and the Future of Programming** • Will programming languages evolve for LLMs? • How far can LLMs go in formal reasoning? • Human–AI collaboration paradigms
18:00–19:30	Dinner	

Day 2 – Tuesday, 9 December 2025

Time	Session / Speaker	Title / Description
09:00–10:00	Talk – Thuan Pham	LLM-Assisted Fuzzing for Effective Security Testing
10:00–10:30	Talk – Yi Li (unofficial remote participant)	Identifying Multi-parameter Constraint Errors in Python Data Science Library API Documentation
10:30–11:00	Break	

Time	Session / Speaker	Title / Description
11:00–12:00	Talk – Toufique Ahmed	Generating Tests from Issues to Validate SWE Patches
12:00–13:30	Lunch	
13:30–14:00	Talk – Jun Sakuma	Disrupting Model Merging: A Parameter-level Defence Without Sacrificing Accuracy
14:00–14:30	Talk – Mohamed Amine Merzouk	AI for Secure Software: Capabilities, Vulnerabilities, and Alignment for LLM-Driven Systems
14:30–15:00	Talk – Yang Cao	Privacy in Fine-tuning for Large Language Models: Attacks, Defences, and Future Directions
15:00–15:30	Break	
15:30–17:30	**Panel Discussion** (Moderator: Yang Cao)	**Verification, Safety and Trust in LLM-Generated Code** • Ensuring reliability in AI-generated artefacts • Integrating symbolic/static techniques with LLMs • Safety constraints and governance
18:00–19:30	Dinner	

Day 3 – Wednesday, 10 **December 2025**

Time	Session / Speaker	Title / Description
09:00–09:30	Talk – Yongqiang Tian	Supercharge Compiler Engineering with LLMs
09:30–10:00	Talk – Ruitao Feng	Correcting Decompilation Distortions with LLMs
10:00–10:30	Talk – Yuekang Li	Detecting Bugs in the LLM Ecosystem
10:30–11:00	Break	
11:00–11:45	**Panel Discussion** (Moderator: Yuekang Li)	**Research Infrastructure for LLM-Era Software Engineering** • Benchmarks and datasets • Tooling and open platforms • Community efforts and reproducibility
12:00–13:30	Lunch	
13:30–18:15	Excursion	Visiting the Great Buddha and Hase Temple
18:15–21:00	Main banquet	

Day 4 – Thursday, 11 December 2025

Time	Session / Speaker	Title / Description
09:00–10:30	Wrap-up	Wrap-up and report planning
10:30–11:00	Break	
11:00–11:30	Closing	Closing remarks
12:00–13:30	Lunch	

Summary of Discussions

We held one panel discussion on each of the first three days of our Shonan meeting. As an experiment, we trialled an AI-enabled panel format using Zoom's AI Companion to generate real-time summaries of the discussions. Participants first discussed the necessary technical arrangements, including muting microphones and managing audio inputs to prevent echo. They confirmed that only one person should speak at a time and emphasised the importance of keeping microphones muted when not speaking. The group agreed to begin each discussion once everyone was ready, with the understanding that sessions might finish early if there were few questions. After each panel, the AI tool generated a summary and emailed it to participants so that panel members and attendees could review, edit, and correct any imprecise or inaccurate content.

Panel Discussion 1: LLMs and the Future of Programming

Summary

The panel explored the evolution of programming languages in relation to large language models (LLMs), discussing how current languages might need to adapt to better leverage LLM capabilities and considering the future of human–machine interaction. The discussion covered AI's role in programming education and research, as well as the possibility of LLMs generating their own tools or programming languages, while emphasising the continued need for human oversight to ensure correctness, reliability, and security.

LLMs and Programming Language Evolution

The panel discussed whether existing programming languages should evolve to better support LLMs or whether LLMs should instead improve their understanding of natural language. Xiaofei suggested that language adaptations may be use-case dependent, while Yulei Sui expressed optimism about AI agent programming, particularly for improving specifications, architecture design, and testing strategies. The group also considered the need for new protocols to enable effective communication between collaborative agents. Yongqiang noted that future languages may embed features to better support security and privacy and to mitigate hallucination issues in LLMs.

Future of Human–Machine Language Interaction

Participants discussed how programming languages currently act as an interface between humans and machines, and whether future interactions could become more natural and direct. The discussion was largely exploratory, considering the limitations of both human language and current language models, as well as the possibility of unified languages that integrate natural language with formal reasoning. The panel also reflected on communication among AI agents, between AI agents and software, and between AI agents and humans, and how these interactions affect the controllability of AI systems.

AI in Programming Education

The panel examined the impact of LLMs on programming education and human–AI collaboration. Ruitao shared concerns about students relying heavily on AI tools for assignments, potentially weakening their grasp of programming fundamentals. Yulei noted that tools such as copilots are now integral to modern programming practice, raising questions about how education should ensure that students still develop strong conceptual foundations.

Future Directions

The discussion concluded with reflections on how programming education might shift toward logic, design, and high-level reasoning rather than low-level coding. The panel also considered efficiency and energy consumption issues across programming languages, and whether AI might replace integrated development environments rather than programming languages themselves. Overall, the panel agreed that while LLMs may automate routine tasks, complex programming and verification will continue to require human expertise.

Panel Discussion 2: Verification, Safety, and Trust in LLM-Generated Code

Summary

This panel focused on the trustworthiness and security of AI-generated code, addressing vulnerabilities, limitations, and challenges in software engineering workflows. The discussion examined testing and verification approaches, the role of LLMs in both code generation and validation, and the need for context-dependent notions of trust. The panel emphasised the importance of human oversight and identified AI alignment and trustworthiness as key open research challenges.

Trustworthiness and Security Challenges

The panel discussed security risks and correctness issues in AI-generated code. Examples highlighted potential vulnerabilities arising from hallucination and unintended behaviour. Participants agreed that trustworthy AI and trustworthy software share overlapping concerns, including safety, robustness, privacy, and security, and that insights from both domains should inform future solutions.

Customisable Trust and Industry Perspectives

The discussion highlighted that trustworthiness requirements vary across contexts. For example, security may be paramount for smart contracts, while functionality may dominate in less critical software. Participants noted that industry adoption often prioritises reproducibility and tangible usability, underscoring the importance of understanding industrial constraints when designing research solutions.

Testing and Verification of LLM-Generated Code

The panel examined limitations of traditional testing methods and the promise of LLM-assisted testing and verification. Suggestions included leveraging domain knowledge, system specifications, and symbolic reasoning techniques. Concerns were raised about the scalability of human oversight as LLM capabilities grow, and the risk of over-automation in programming and testing tasks.

Security Risks and Emerging Vulnerabilities

Participants debated whether LLMs introduce fundamentally new vulnerability types or amplify existing ones. While some argued that LLMs mainly generate variations of known vulnerabilities, others suggested that new attack surfaces may emerge from complex interactions between AI-generated code and other systems. The discussion concluded with agreement on the need for improved tools, protocols, and safer training paradigms.

Panel Discussion 3: Research Infrastructure for LLM-Era Software Engineering

Summary

This panel explored research infrastructure needs for evaluating and supporting LLM-based software engineering. Topics included benchmarking methodologies, tooling, ethical considerations, and the role of human oversight. The discussion highlighted challenges in balancing fairness, cost, and realism in benchmarks, as well as concerns about the ethical use of LLMs in academic research and peer review.

Community Benchmarking

Participants discussed community-maintained benchmarking platforms, including "arena-style" evaluations and industry-hosted benchmarks. Suggestions included collaborating with companies to host independent benchmarks that ensure fairness while reducing costs, drawing inspiration from successful benchmark initiatives in other research areas.

Challenges in Benchmark Design

The panel discussed differences between academic and industry benchmarks, data leakage risks, and the need for dynamically generated tasks. Security-focused benchmarks were noted as particularly challenging due to sensitivity and disclosure concerns. Participants emphasised the importance of distinguishing between model bugs and specification-driven behaviours.

Tooling and Human–AI Interaction

Experiences with graphical and command-line tools were discussed, along with the need for better interaction models that support different user expertise levels. The panel highlighted the importance of human-in-the-loop approaches to guide LLMs, reduce hallucinations, and improve collaborative problem solving.

Ethics and Peer Review

The panel addressed ethical issues surrounding LLM use in academic writing and peer review. While LLMs may assist with summarisation and fact-checking, participants stressed that humans must remain responsible for evaluation and decision-making to ensure fairness, quality, and accountability.

Summary of New Findings

We sincerely appreciate all participants' contributions through their talks and discussions. The meeting was successful, inspiring, and conducted in a warm and collegial manner. It also provided an opportunity to reflect on the importance of community-driven, timely research challenges at the intersection of large language models (LLMs) and computer programming. Participants considered whether future developments should remain human-centric—emphasising capability enhancement and ethical oversight—or shift toward a more AI-centric model that embraces automation while raising new questions of trust, reliability, and governance. Strengthening automated testing, verification, and repair for LLM-generated code was identified as essential for improving system trustworthiness. In education, assessments may increasingly shift from code writing to design, comprehension, testing, and debugging. Finally, the meeting highlighted the need for deeper academia–industry collaboration to develop shared benchmarks, advance alignment and safety, and ensure that LLM technologies are deployed responsibly and effectively.

We summarise the main reflection points as follows:

- Researchers in this area are encouraged to begin by identifying well-defined and meaningful research problems, and then develop corresponding benchmarks that evaluate these problems across multiple dimensions, such as quality, robustness, diversity, comprehensiveness, and representativeness.
- Looking ahead, both research and education may be shaped by whether the future world remains fundamentally human-centric or becomes increasingly AI-centric. A human-centric perspective emphasises enhancing human capabilities and strengthening responsible oversight, where AI serves primarily as an advanced tool. In contrast, an AI-centric perspective prioritises automation, autonomous decision-making, and system-level optimisation, which also introduces societal considerations such as trust, accountability, and governance. This tension was reflected in the meeting discussions. For example, should students still learn programming in the era of LLMs? Should new programming languages be designed specifically for LLM-based development? Under a human-centric perspective, students still need to acquire coding skills, and programming languages must remain interpretable and readable for humans. Under an AI-centric perspective, coding may no longer be an essential human skill, as we increasingly rely on AI to generate and verify programs autonomously.
- Facilitating research on (semi-)automated testing, debugging, and repair for LLM-generated code is essential to improving trustworthiness.
- In education, given that LLMs can help students write code effectively, assessment focus could shift toward software design, code comprehension, code review, testing, debugging, and fixing/repairing issues.
- For academia–industry collaboration, it would be highly valuable to create community-maintained, large-scale, industry-relevant benchmarks and evaluation tools.
- Future academia–industry collaboration will be essential to address LLM-related challenges by bridging the gap between theory and practice. Academia

offers foundational insights into bias, robustness, and interpretability, while industry contributes scale, real-world data, and deployment expertise. Joint efforts should prioritise shared benchmarks, alignment and safety research, efficient model design, and ethical frameworks for responsible use.

- Finally, the meeting served as an important platform for shaping the future agenda of LLM-era software engineering and security research. Follow-up collaborations and community initiatives are expected to continue beyond this meeting.

Identified Issues and Future Directions

The discussions highlighted several open issues and future research directions at the intersection of large language models (LLMs) and computer programming:

- **Semantic understanding and evaluation:** While LLMs show strong empirical performance, their semantic reasoning capabilities remain limited. Current benchmarks often focus on surface-level behaviour, motivating the need for evaluation grounded in program semantics, testing, and formal analysis.
- **Trustworthiness of LLM-generated code:** Ensuring correctness, security, and robustness in LLM-assisted development workflows remains challenging. Common concerns include hallucination, violations of specifications, and the limited scalability of human oversight, pointing to the need for tighter integration with analysis, testing, and verification techniques.
- **Research infrastructure and benchmarks:** The community lacks shared benchmarks and tooling that reflect realistic software engineering and security scenarios. Reproducibility, cost, and transparency were identified as important considerations for future evaluation efforts.
- **Future research directions:** Promising avenues include hybrid human-AI workflows that combine LLM flexibility with formal guarantees, automated testing and repair for LLM-generated code, and new programming abstractions and specifications tailored to AI-assisted development. Educational implications were also noted, with a likely shift toward program comprehension, design, verification, and critical evaluation.

Acknowledgement

- NII Shonan Meeting Staff and Officers
- All participants
- UNSW CSE for small courtesy gifts
- AI tools to summarize our panel discussions (i.e., Zoom AI Companion) and help with polishing meeting minutes