



Preface to the Special Issue on Software Quality Assurance in the Era of Large Language Models

Zan Wang (王赞)¹, Ying Wang (王莹)², Bihuan Chen (陈碧欢)³, Yuan Yao (姚远)⁴,
Minling Zhang (张敏灵)⁵

¹ (College of Intelligence and Computing, Tianjin University, Tianjin 300350, China)

² (Software College, Northeastern University, Shenyang 110169, China)

³ (School of Computer Science, Fudan University, Shanghai 200438, China)

⁴ (School of Computer Science, Nanjing University, Nanjing 210023, China)

⁵ (School of Computer Science and Engineering, Southeast University, Nanjing 211189, China)

Corresponding author: Zan Wang, wangzan@tju.edu.cn

Citation Wang Z, Wang Y, Chen BH, Yao Y, Zhang ML. Preface to the special issue on software quality assurance in the era of large language models, *International Journal of Software and Informatics*, 2025, 15(2): 139–144. <http://www.ijsi.org/1673-7288/362.htm>

In recent years, with the rapid advancement of deep learning technologies, large language models (LLMs) have achieved unprecedented breakthroughs in natural language processing. Representative models such as ChatGPT and DeepSeek are evolving rapidly and have demonstrated remarkable potential and broad application prospects in a range of software engineering tasks, including code generation, software testing, and program analysis and optimization. As general-purpose artificial intelligence models endowed with deep semantic understanding and generative capabilities, LLMs are redefining conventional approaches to software quality assurance, injecting new momentum into the automation and intelligence of software engineering. Additionally, software quality assurance plays a critical role in ensuring that software systems meet design specifications and user expectations in terms of functionality, performance, reliability, and security. However, conventional methods fail to keep pace with the increasing scale and complexity of modern software systems. This challenge is especially pronounced in highly complex application scenarios such as open-source software, enterprise-level systems, and chip design programs, where the degree of automation and intelligence provided by conventional approaches often falls short of practical requirements.

Driven by evolving technological demands and resonating with industrial developments, the research on the LLM-assisted software quality assurance is gaining increasing attention from both academia and industry. LLMs not only possess powerful capabilities in code comprehension and generation but also have achieved remarkable results in critical tasks such as software testing, code optimization, and code translation. Their large-scale pre-training paradigm and high scalability make them a key technological foundation for developing intelligent software quality assurance tools. Nevertheless, significant challenges remain in effectively integrating LLMs with existing software engineering practices to build intelligent systems that are applicable to real-world scenarios.

Against the above backdrop, this special issue focuses on large language models for quality assurance (LLM4QA), aiming to explore cutting-edge research and recent advances in the application of LLMs to software quality assurance. It highlights the technical principles, system implementations, and experimental evaluations of model applications, presents diverse research outcomes and innovative solutions, and investigates the potential and prospects of LLMs in enhancing software development and maintenance processes. This special issue places particular emphasis on several key research directions, including the application of LLMs in software testing, techniques for code optimization and refactoring, code translation, license compliance risk assessment in code generation, and multi-agent collaborative systems. Through the contributions of this special issue, it is hoped to promote the deep integration of LLMs with software quality assurance, advance synergistic innovation in theory and practice, and provide valuable references for the automation and intelligence of software engineering.

In recent years, numerous researchers in China have conducted in-depth research on software quality assurance empowered by LLMs, with their achievements advancing from theoretical exploration to practical engineering implementation. Research efforts in this field from both academia and industry in China have exhibited clear trends toward technological integration, scenario diversification, and the refinement of toolchains. Representative research achievements in this area can be summarized as follows.

(1) Software testing methods driven by LLMs

The powerful reasoning and generative capabilities of LLMs are reshaping the direction of intelligent development in software testing, offering innovative solutions in various areas such as test case generation, defect detection, and fuzz testing. For instance, a research team from the University of Science and Technology of China conducted a comprehensive analysis and summary of the current development status of LLM-based fuzz testing. They outlined the overall fuzz testing workflow and commonly used LLM-related techniques in fuzz testing research, discussed the limitations of deep learning-based fuzz testing methods prior to the emergence of LLMs, analyzed the application of LLMs in different stages of fuzz testing, and explored the main challenges and potential future directions of LLM-based fuzz testing. In the domain of resource leak detection, a team from Beijing Institute of Technology investigated the effectiveness of detection methods based on deep learning, small models, and large models in identifying resource leaks. They explored several amelioration techniques, including prompt enhancement, fine-tuning, and the integration of static analysis with LLMs. Furthermore, they evaluated the performance of different models from multiple dimensions, such as root causes of resource leaks, types of resources, and code complexity. Experimental results demonstrated that fine-tuning significantly improves the detection performance of large models in the context of resource leak detection. In addition, they examined the ability of different models to identify defects involving unclosed resources along typical execution paths and analyzed the impact of code complexity on deep learning-based detection methods. In the area of testing for chip design programs, a team from Tianjin University conducted a systematic review of existing intelligent testing approaches from three perspectives: test input generation, test oracle construction, and test execution optimization. Their work focused on the evolution of chip design program testing methods from the machine learning stage, through the deep learning stage, to the stage involving LLMs. They discussed the potential of methods at each stage to improve testing efficiency and coverage while reducing testing costs. Additionally, they compiled research datasets and tools used in chip design program testing and provided an outlook on the future development directions and challenges for quality assurance in chip design programs driven by LLMs.

(2) Code analysis and optimization driven by LLMs

In code analysis and optimization, LLMs, with advanced capabilities in code comprehension

and refactoring, have overcome the limitations of static analysis methods. By leveraging context awareness and semantic association, they enable code refactoring and performance enhancement, offering new approaches to improving code quality. For example, a research team from Nanjing University conducted an in-depth investigation into code optimization tasks, evaluating the overall advantages of large models compared with small pre-trained models. They proposed an optimized evaluation method based on code change analysis to quantify the degree of adjustment during the refactoring process. In addition, their work introduced an approach integrating the strengths of multiple large models, effectively combining their capabilities to enhance the quality of optimization. In the task of long method decomposition, a team from Beijing Institute of Technology proposed a method combining heuristic rules with the reasoning capabilities of large models. This method automatically identifies functional boundaries within code and splits excessively long methods in a rational manner, thereby improving code readability and maintainability. Experimental results showed that the application of this method in Java projects leads to significant optimization outcomes, generating code that is clearer, more aligned with developers' coding practices, and less complex. As code optimization tasks continue to evolve, the adaptive optimization capabilities of LLMs are expected to further enhance the intelligence of code refactoring processes.

(3) Code translation driven by LLMs

Cross-language code transformation and dialect adaptation present great challenges to software migration and compatibility maintenance. The powerful semantic mapping capabilities of LLMs are offering novel solutions to these problems. A research team from Nanjing University proposed a Datalog code translation framework based on LLMs, which employs techniques such as divide-and-conquer translation strategies, few-shot learning prompts, and chain-of-thought reasoning to achieve more accurate translation between different Datalog dialects. In addition, the framework integrates an iterative error correction mechanism that continuously improves translation quality through a check-feedback-repair loop, thereby enhancing the accuracy of code migration. In experimental evaluations, this method not only achieved high accuracy in cross-dialect code translation tasks but also effectively supported incremental program analysis, accelerating the adaptation and optimization of cross-dialect programs. This work demonstrates that LLMs can not only improve the quality of code translation but also advance the development of cross-language static analysis, providing robust compatibility support for heterogeneous programming environments.

(4) Open-source license analysis in the context of LLMs

With the widespread application of LLMs in the code generation domain, ensuring license compliance of generated code has become a critical issue in software engineering. In view of this issue, a research team from Northeastern University proposed a code clone analysis framework for detecting violations of open-source licenses. This framework is capable of tracing the origins of code generated by LLMs and identifying potential copyright infringement risks. Through comparison between model-generated code and existing open-source code repositories, the work revealed that some generated contents exhibit high similarity and might even lack essential license information, which brings potential legal risks to the open-source software ecosystem. Experimental analyses further disclosed compliance risks associated with different license types and investigated variations among different open-source projects concerning code reuse and license declarations. Moreover, the work examined the long-term impacts of code generation by LLMs on the software ecosystem and advocated for establishing stricter compliance review mechanisms for generated code. Such mechanisms would ensure that while LLMs enhance development efficiency, they can simultaneously mitigate potential legal issues.

(5) Multi-agent collaborative systems empowered by LLMs

Multi-agent systems are becoming important automation tools in software development processes, and the integration of LLMs has further enhanced the collaborative capabilities of agents. In the domain of code review, a team from Xidian University proposed an intelligent reviewer recommendation framework based on LLMs. This method comprehensively analyzes the contents of code changes and the expertise of reviewers, intelligently matching the most suitable code reviewers to improve review efficiency and code quality. The system employs a multi-agent collaborative mode, where each agent is responsible for a specific task, such as code feature extraction, reviewer profiling, and recommendation optimization, thus forming a closed-loop collaboration mechanism. Experimental results indicated that the proposed method could maintain recommendation accuracy while providing more interpretable recommendation rationales, thereby making the code review process more transparent and efficient. Additionally, multi-agent collaboration can be extended to other stages such as software testing and defect fixing, improving task efficiency through intelligent division of labor and optimizing the overall development workflow. With the advancement of agent technologies, this collaborative model is expected to further drive the intelligent evolution of software engineering.

This special issue includes five representative papers, with brief introductions provided as follows:

The article *“Insights and Analysis of Open-source License Violation Risks in LLMs Generated Code”* focuses on the license violation risks associated with code generated by LLMs in relation to open-source repositories. The authors develop a code clone detection framework to trace the origin of model-generated code and reveal issues of copyright infringement. The paper analyzes the actual proportion of code cloning and license omission, exploring the impact of code generation by LLMs on the open-source software ecosystem.

The article *“Exploration and Improvement of Capabilities of LLMs in Code Refinement Task”* investigates the performance of LLMs and traditional small-scale pre-trained models in automated code optimization tasks. The authors evaluate the comprehensive advantages of LLMs in this domain, propose Unidiff-based code refinement evaluation metrics to quantify changes during the optimization process, and introduce the LLM-Voter method, which integrates the strengths of multiple models to enhance code optimization quality.

The article *“Detection of Resource Leaks in Java Programs: Effectiveness Analysis of Traditional Models and Language Models”* examines the effectiveness of deep learning-, small model-, and large model-based detection methods in resource leak detection tasks. The authors explore various amelioration strategies, including prompt enhancement, fine-tuning, and the integration of static analysis with LLMs. Experimental results demonstrate that fine-tuning techniques significantly enhance the performance of LLMs in detecting resource leaks.

The article *“Large-language-model-based Decomposition of Long Methods”* proposes a novel model, Lsplitter, designed to automatically decompose long methods. By combining LLMs with heuristic rules, Lsplitter achieves significant optimization results when being applied to real-world Java projects.

The article *“LLM-powered Datalog Code Translation and Incremental Program Analysis Framework”* introduces a LLM-based Datalog code translation framework. Utilizing a divide-and-conquer translation strategy, prompt engineering based on few-shot and chain-of-thought prompting, and an iterative error correction mechanism based on a check-feedback-repair loop, the framework achieves highly accurate code translation among different Datalog dialects. Moreover, it enables high-precision code transformation and incremental program analysis, accelerating cross-dialect program analysis tasks.



Zan Wang, Ph.D., professor and doctoral supervisor at Tianjin University. His research focuses on quality assurance of artificial intelligence systems and software engineering, particularly in quality assurance of deep learning systems and fundamental software testing. In the past five years, he has published over 20 academic papers in top-tier international and domestic conferences and journals in the field of software engineering, including ICSE, FSE, ASE, TSE, TOSEM, Chinese Journal of Computers, and Journal of Software. His paper entitled “Deep Learning Library Testing via Effective Model Generation”, published in FSE 2020, received the ACM SIGSoft Distinguished Paper Award. Over the past five years, he has led multiple research projects, including those funded by the National Natural Science Foundation of China. As the first contributor, he was awarded one second prize of Tianjin Science and Technology Progress. He has taught courses such as Database Principles and Software Testing, receiving two second prizes for teaching achievements in Tianjin.



Ying Wang, Ph.D., associate professor and doctoral supervisor at Northeastern University. She serves as a committee member of both the CCF Open Source Development Committee and the CCF Women’s Working Committee. She was awarded the Microsoft Research Asia StarTrack Visiting Scholar (2020), nominated for the Outstanding Doctoral Dissertation Award by the China Computer Federation (2020), received the Excellent Doctoral Dissertation Award of Liaoning Province (2021), and won ACM SIGSOFT Distinguished Paper Awards (ICSE 2021, ESEC/FSE 2023). Her research interests include intelligent software development, open-source software ecosystem governance, and software supply chain analysis. She has published a series of research achievements on ecosystem governance across various programming languages, including Java, C#, Python, Go, JavaScript, Android, and Rust, and developed a series of tool platforms named “League of Legends (LOL)” to automatically detect dependency-related defects in open-source ecosystems. Several of her technologies have been implemented in platforms at Huawei, Microsoft, Loongson, and within the OpenHarmony open-source community. She currently serves as a member of the editorial board of IEEE Transactions on Software Engineering, as chair of the ChinaSoft 2023-2024 Women in Software Engineering Forum, co-chair of the SANER 2023 Tool Track, and chair of the “Beauty of Computing” Doctoral Forum 2021. She actively participates in academic activities to encourage female computer scientists and programmers to pursue their research aspirations.



Bihuan Chen, Ph.D., associate professor and doctoral supervisor at Fudan University. His research interests include software supply chains, intelligent connected vehicles, and AI system engineering. He has led two projects supported by the National Natural Science Foundation of China and multiple industry-sponsored projects and participated in the Sci-Tech Innovation 2030 Major Project on “New Generation Artificial Intelligence”. His research achievements have been published in leading international conferences and journals, including ICSE, FSE, S&P, SEC, TSE, and TIFS. He has received the NASAC Young Software Innovation Award, three ACM SIGSOFT Distinguished Paper Awards (FSE 2016, ASE 2018, and ASE 2022), and two IEEE TCSE Distinguished Paper Awards (ICSME 2020 and SANER 2023). Based on his research achievements, he developed FUXI (<http://www.se.fudan.edu.cn/fuxi/>), an open-source risk governance platform.



Yuan Yao, Ph.D., associate professor and doctoral supervisor at Nanjing University. In recent years, his research interests have primarily focused on machine learning techniques and their applications in intelligent software engineering. His research has been supported by a series of national major and key projects, with the achievements successfully adopted by enterprises such as Huawei. He has published over 60 papers in leading international conferences, including ICSE, CCS, S&P, NDSS, ICLR, NeurIPS, and KDD. His work has also been featured in MIT Technology Review, a leading authoritative media outlet in the computer science community.



Minling Zhang, Ph.D., professor and doctoral supervisor at Southeast University, currently serves as Dean of the School of Computer Science and Engineering. His main research areas include machine learning and data mining. He is currently Vice Chair of the Machine Learning Committee of the Chinese Association for Artificial Intelligence (CAAI) and Vice President of the Jiangsu Association for Artificial Intelligence. He serves as an editorial board member for several academic journals, including SCIENCE CHINA Information Sciences, IEEE Transactions on Pattern Analysis and Machine Intelligence, ACM Transactions on Intelligent Systems and Technology, Frontiers of Computer Science, and Machine Intelligence Research. He is also a steering committee member for conferences such as ACML and PAKDD and has served as Program Chair for various international and domestic conferences including PRICAI, CCF-ICAI, and CCFAI, as well as Area Chair or a Senior Program Committee member more than 60 times for conferences such as KDD, IJCAI, AAAI, and ICDM. He has received the CCF-IEEE CS Young Scientist Award (2016) and been supported the National Science Fund for Distinguished Young Scholars (2022).