Research Statement

XIE Xiaofei
School of Computing and Information Systems, Singapore Management University
Tel: (65) 6826-4861; Email: xfxie@smu.edu.sg
20 (Day) 12 (Month) 2023 (Year)

Background

Today software is everywhere such as the mobile, computer and TV, everything runs on software. With the booming of intelligent systems such as smart phones, self-driving cars, and robotics, we are increasingly dependent on the correct operation of software. As software is becoming larger and more complex, it is a big challenge for guaranteeing correctness and security of software. Moreover, the form of software is becoming diverse, from traditional code-based software to new data-driven software (e.g., intelligent software). Due to the fundamental differences between code-based software and data-driven software, existing techniques on traditional software are not available in AI software. Hence, with the increasing complexity and diversity of modern software systems, we are forced to rethink questions such as how to test and verify different kind of traditional software effectively and how to assure the quality of the intelligent software in such context.

Motivated by the above challenges, my research goal is to develop automated techniques and practical tools for quality analysis of traditional software and intelligent software. Specifically, for traditional software, I work on fundamental program analysis and software testing techniques to verify the correctness and discover bugs. I have proposed techniques on different kinds of software like JavaScript engines, game software and others. For intelligent software, I have worked on automated analysis on trustworthy AI including the data-centric analysis and mode-centric analysis.

To address the challenges of assuring software quality, my research work mainly focuses on the following areas: program analysis, software testing, AI for software engineering and software engineering for AI.

Program and Loop Analysis

Program analysis is the fundamental technique of automatically analyzing the behavior of computer programs regarding some properties such as correctness, robustness, and liveness. Loop structures are one of the biggest challenges for program analysis. As a classical problem (e.g., in software verification and termination proof), loop analysis has been studied for many years. An effective loop analysis can greatly improve the overall program analysis, making it an important area of study in computer science.

Focused on this hard problem, I first proposed a loop classification [1] to systematically understand the difficulty of analyzing different types of loops. Based on the classification, a series of summarization techniques [1,2,3] have been developed and customized for handling different loops such as linear loops, non-linear loops,

and string loops. The calculated loop summary could be applied to improve the performance on symbolic execution, loop bound analysis and vulnerability detection.

In addition, with the loop summary, I also developed a lightweight framework to automatically analyze the program termination [4] that is a classical halting problem. Furthermore, we conducted an empirical study on the termination bugs in real-world projects [36]. We found that there is a big gap between existing termination benchmark and real-world termination bugs. The existing techniques mainly focus on theory while that cannot be used in detecting real-world termination bugs. To address this problem, we developed the first practical non-termination detection method [39] that can detect real-world non-termination issues.

Future Research Plan. In the future, I will focus on: 1) to analyze more complex loops and recurrent functions. I plan to apply advanced AI techniques to handle complex loops and 2) develop practical techniques and tools to detect real-world non-termination issues, particularly focusing on non-terminating recursive functions.

Software Testing

Testing is the main method to test the correctness and robustness of software. I am quite interested in detecting software bugs especially vulnerabilities that could be exploited by attackers. The main challenges that I mainly focus on in software testing include: 1) how to develop effective algorithm that can discover bugs fast; 2) how to discover different kinds of bugs (e.g., logical bugs, memory related bugs, GUI bugs) and 3) how to test different kinds of software (e.g., smart contract, JavaScript engine, Web application and Android application). For the algorithm level, I have developed coverage-guided testing techniques (e.g., fuzzing) [5,6,7] to detect bugs, including the test case selection, test case mutation, testing guidance etc. With these techniques, I have discovered 100+ security vulnerabilities in open-source projects. I also developed techniques to detect different kinds of bugs such as use-after-free bugs [5], memory corruption bugs [25], performance bugs [37] and logical bugs [26]. Moreover, I developed techniques for testing different software such as game software [9, 27, 28, 29, 40], Web applications [10], network protocols [26], JavaScript engines [7] and deep learning frameworks [24,30]. After new bugs are detected, I developed the root cause analysis technique [8] towards more effective debugging. In addition, I developed approaches to mitigate cross-site scripting attacks in web applications [11] and Android malware detection [12,13].

In addition, I also developed testing methods to evaluate the quality and reliability of autonomous driving systems [41, 42]. We mainly focus on the problem of discovering realistic and diverse critical scenarios.

Future Research Plan. In the future, I plan to focus on 1) test driver synthesis. Existing test drivers mainly rely on human, which could be costly and incomplete. 2) Targeting on new types of bugs and software such as block chain security and cloud infrastructure security. I plan to collect new types of bugs/vulnerabilities, which can be further used to improve the bug detection capability. 3) How to better apply Large Language Model on software testing is a promising direction.

AI for Software Engineering (AI4SE)

With the booming development of open-source software, the amount of available code-related data (a.k.a. “big code”) has reached an unprecedented scale, which inspires researchers from both academia and the industry to explore employing data-driven approaches (e.g., machine learning) for diverse code-related tasks such as type inference, clone detection, code summarization and code summarization. I am interested in how to better mine knowledge from big code, which can provide useful information to developers. For example, we proposed a novel retrieval-based augmentation for effectively learning summarization from source code [31], which can effectively improve the learning performance. Considering the different programming language data in big code, we proposed the cross-lingual adaptation for type inference, i.e., we would like to transfer the knowledge from one program language to another language [32]. Moreover, we also proposed the techniques to better learn the code semantics and representations [43, 44].

Another interesting direction is how to better apply machine learning in software testing. We conducted an empirical study on deep learning for fuzz testing. We observed that existing DL based approaches are not effective and some future directions are concluded based on the study results [33]. After the bugs are identified (e.g., from testing), a problem is how to automatically repair the buggy programs. We adopted the deep learning on generating patches by learning from the history information of big code [34,35].

Future Research Plan. In the future, I mainly focus on 1) data quality analysis for different tasks. Although there are a lot of open-source code, the quality of the code and documentation is not guaranteed, which could affect the training performance. 2) I plan to focus on model robustness evaluation and interpretation to make trained models more reliable and trustworthy. 3) I will also plan to study the large language models on different code learning tasks and how to improve the performance.

Software Engineering for AI (SE4AI)

Deep learning (DL) has been widely applied in many cutting-edge applications. However, deep neural networks (DNNs) are vulnerable (e.g., suffering from adversarial examples). Considering the fundamental difference between traditional software and deep learning software, it is a new challenge on quality assurance of deep learning systems. In this new area, my research has contributed to quality assurance of DL software during the development phase and deployment phase.

- *DL Testing in Development Phase*. Considering the unique characteristics of DL software, the testing requires the analysis on the model as well as the training data. I have proposed techniques on data analysis [14,15,16] and model testing [17, 18, 45, 46]. For data quality analysis, we characterize the data sensitivity [14], data uncertainty [15] and data distribution [16] for better understanding the training data and test data. For model testing, we proposed new testing criteria [18,19,38], input mutation techniques [20], seed input selection [46], distribution-guided testing [45] and coverage-guided testing frameworks [17,18]. The techniques have been demonstrated effective in finding weaknesses for image classification, speech to text, and natural language processing. After the problems are identified, I proposed the automatic repair techniques to improve the quality of the target models [21,22].

- *DL Testing during Deployment Phase*. Although DNNs have been well-tested during the training phase, it can still cause issues when they are deployed in environments that are different with the training environment, e.g., different devices, different frameworks, and different platform. To ensure the quality of the DNN after deployment, I have proposed the testing techniques to detect compatibility issues [23] on different environments. I also developed the framework to detect bugs in the deep learning frameworks (e.g., PyTorch, TensorFlow) [24, 30], which can largely affect the quality of DNNs.

Future Research Plan. In the future, I mainly focus more on applications based on existing research, which can further demonstrate the usefulness of our research. Specifically, I will focus on the trustworthy AI in different domains, e.g., cybersecurity (malware detection, intrusion detection), autonomous driving and healthcare. Furthermore, with the rapid development of large language models, I will also focus on testing and analyzing the quality of large language models.

Selected Publications and Outputs

[1] **Xiaofei Xie**, Bihuan Chen, Yang Liu, Wei Le, and Xiaohong Li. "Proteus: Computing disjunctive loop summary via path dependency analysis." In Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE), pp. 61-72. 2016

[2] **Xiaofei Xie**, Yang Liu, Wei Le, Xiaohong Li, and Hongxu Chen. "S-looper: Automatic summarization for multipath string loops." In Proceedings of the 2015 International Symposium on Software Testing and Analysis (ISSTA), pp. 188-198. 2015.

[3] **Xiaofei Xie**, Bihuan Chen, Liang Zou, Yang Liu, Wei Le, and Xiaohong Li. "Automatic loop summarization via path dependency analysis." IEEE Transactions on Software Engineering 45, no. 6 (2017): 537-557.

[4] **Xiaofei Xie**, Bihuan Chen, Liang Zou, Shang-Wei Lin, Yang Liu, and Xiaohong Li. "Loopster: Static loop termination analysis." In Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering (ESEC/FSE), pp. 84-94. 2017.

[5] Wang, Haijun, **Xiaofei Xie**, Yi Li, Cheng Wen, Yuekang Li, Yang Liu, Shengchao Qin, Hongxu Chen, and Yulei Sui. "Typestate-guided fuzzer for discovering use-after-free vulnerabilities." In 2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE), pp. 999-1010. IEEE, 2020.

[6] Chen, Hongxu, Yinxing Xue, Yuekang Li, Bihuan Chen, **Xiaofei Xie**, Xiuheng Wu, and Yang Liu. "Hawkeye: Towards a desired directed grey-box fuzzer." In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS), pp. 2095-2108. 2018.

[7] Xiaoyu He, **Xiaofei Xie**, Yuekang Li, Jianwen Sun, Feng Li, Wei Zou, Yang Liu, Lei Yu, Jianhua Zhou, Wenchang Shi and Wei Huo, ``SoFi: Reflection-Augmented Fuzzing for JavaScript Engines'', In the ACM Conference on Computer and Communications Security (CCS), 2021

[8] Wang, Haijun, **Xiaofei Xie**, Shang-Wei Lin, Yun Lin, Yuekang Li, Shengchao Qin, Yang Liu, and Ting Liu. "Locating vulnerabilities in binaries via memory layout recovering." In Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE), pp. 718-728. 2019.

[9] Zheng, Yan, **Xiaofei Xie**, Ting Su, Lei Ma, Jianye Hao, Zhaopeng Meng, Yang Liu, Ruimin Shen, Yingfeng Chen, and Changjie Fan. "Wuji: Automatic online combat game testing using evolutionary deep reinforcement learning." In 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE), pp. 772-784. IEEE, 2019

[10] Zheng, Yan, Yi Liu, **Xiaofei Xie**, Yepang Liu, Lei Ma, Jianye Hao, and Yang Liu. "Automatic Web Testing Using Curiosity-Driven Reinforcement Learning." In 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE), pp. 423-435. IEEE, 2021

[11] Xu, Guangquan, **Xiaofei Xie**, Shuhan Huang, Jun Zhang, Lei Pan, Wei Lou, and Kaitai Liang. "JSCSP: a Novel Policy-Based XSS Defense Mechanism for Browsers." IEEE Transactions on Dependable and Secure Computing (TDSC), 2020

[12] Fan, Ming, Wenying Wei, **Xiaofei Xie**, Yang Liu, Xiaohong Guan, and Ting Liu. "Can we trust your explanations? Sanity checks for interpreters in Android malware analysis." IEEE Transactions on Information Forensics and Security (TIFS), 2020, 838-853.

[13] Feng, Ruitao, Sen Chen, **Xiaofei Xie**, Guozhu Meng, Shang-Wei Lin, and Yang Liu. "A performance-sensitive malware detection system using deep learning on mobile devices." IEEE Transactions on Information Forensics and Security (TIFS), 2020, 1563-1578.

[14] Hu, Qiang, Lei Ma, **Xiaofei Xie**, Bing Yu, Yang Liu, and Jianjun Zhao. "Deepmutation++: A mutation testing framework for deep learning systems." In 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE), pp. 1158-1161. IEEE, 2019

[15] Zhang, Xiyue, **Xiaofei Xie**, Lei Ma, Xiaoning Du, Qiang Hu, Yang Liu, Jianjun Zhao, and Meng Sun. "Towards characterizing adversarial defects of deep learning software from the lens of uncertainty." In 2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE), pp. 739-751. IEEE, 2020

[16] Berend, David, **Xiaofei Xie**, Lei Ma, Lingjun Zhou, Yang Liu, Chi Xu, and Jianjun Zhao. "Cats are not fish: Deep learning testing calls for out-of-distribution awareness." In Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering (ASE), pp. 1041-1052. 2020

[17] **Xiaofei Xie**, Lei Ma, Felix Juefei-Xu, Minhui Xue, Hongxu Chen, Yang Liu, Jianjun Zhao, Bo Li, Jianxiong Yin, and Simon See. "Deephunter: a coverage-guided fuzz testing framework for deep neural networks." In Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA), pp. 146-157. 2019.

[18] Du, Xiaoning, **Xiaofei Xie**, Yi Li, Lei Ma, Yang Liu, and Jianjun Zhao. "Deepstellar: Model-based quantitative analysis of stateful deep learning systems." In Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE), pp. 477-487. 2019

[19] **Xiaofei Xie**, Tianlin Li, Jian Wang, Lei Ma, Qing Guo, Felix Juefei-Xu and Yang Liu. "NPC: Neuron Path Coverage via Characterizing Decision Logic of Deep Neural Networks'' in ACM Transactions on Software Engineering and Methodology (TOSEM), accepted, 2021

[20] Guo, Qing, Felix Juefei-Xu, **Xiaofei Xie**, Lei Ma, Jian Wang, Bing Yu, Wei Feng, and Yang Liu. "Watch out! motion is blurring the vision of your deep neural networks." In Proceedings of the Thirty-fourth Conference on Neural Information Processing Systems (NeurIPS), 2020

[21] Yu, Bing, Hua Qi, Qing Guo, Felix Juefei-Xu, **Xiaofei Xie**, Lei Ma, and Jianjun Zhao. "DeepRepair: Style-Guided Repairing for Deep Neural Networks in the Real-World Operational Environment." IEEE Transactions on Reliability (2021).

[22] **Xiaofei Xie**, Wenbo Guo, Lei Ma, Wei Le, Jian Wang, Lingjun Zhou, Yang Liu, and Xinyu Xing. "RNNRepair: Automatic RNN Repair via Model-based Analysis. " In International Conference on Machine Learning (ICML), pp. 11383-11392. PMLR, 2021

[23] **Xiaofei Xie**, Lei Ma, Haijun Wang, Yuekang Li, Yang Liu, and Xiaohong Li. "DiffChaser: Detecting Disagreements for Deep Neural Networks." In IJCAI, pp. 5772-5778. 2019

[24] Guo, Qianyu**, Xiaofei Xie**, Yi Li, Xiaoyu Zhang, Yang Liu, Xiaohong Li, and Chao Shen. "Audee: Automated testing for deep learning frameworks." In 2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE), pp. 486-498. IEEE, 2020
[25] Wen, Cheng, Haijun Wang, Yuekang Li, Shengchao Qin, Yang Liu, Zhiwu Xu, Hongxu Chen, **Xiaofei Xie**, Geguang Pu, and Ting Liu. "Memlock: Memory usage guided fuzzing." In Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering, pp. 765-777. 2020.
[26] Quan, Lili, Qianyu Guo, Hongxu Chen, **Xiaofei Xie**, Xiaohong Li, Yang Liu, and Jing Hu. "SADT: syntax-aware differential testing of certificate validation in SSL/TLS implementations." In 2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE), pp. 524-535. IEEE, 2020.
[27] Li, Zhuo, Yuechen Wu, Lei Ma, **Xiaofei Xie**, Yingfeng Chen, and Changjie Fan. "GBGallery: A benchmark and framework for game testing." Empirical Software Engineering 27, no. 6 (2022): 1-27
[28] Wu, Yuechen, Yingfeng Chen, **Xiaofei Xie**, Bing Yu, Changjie Fan, and Lei Ma. "Regression Testing of Massively Multiplayer Online Role-Playing Games." In 2020 IEEE International Conference on Software Maintenance and Evolution (ICSME), pp. 692-696. IEEE, 2020.
[29] Ye, Jiaming, Ke Chen, **Xiaofei Xie**, Lei Ma, Ruochen Huang, Yingfeng Chen, Yinxing Xue, and Jianjun Zhao. "An empirical study of GUI widget detection for industrial mobile games."
In Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, pp. 1427-1437. 2021.
[30] Lili Quan, Qianyu Guo, **Xiaofei Xie**, Sen Chen, Xiaohong Li, and Yang Liu, “Towards Understanding the Faults of JavaScript-Based Deep Learning Systems”, The 37th IEEE/ACM International Conference on Automated Software Engineering 2022.
[31] Shangqing Liu, Yu Chen, **Xiaofei Xie**, Jing Kai Siow and Yang Liu, “Retrieval-Augmented Generation for Code Summarization via Hybrid GNN”. In Proceedings of the 9th International Conference on Learning Representations, Virtual Event, May 4-8, 2021
[32] Zhiming Li, **Xiaofei Xie**, Haoliang Li, Zhengzi Xu, Yi Li, and Yang Liu. 2022. Cross-lingual transfer learning for statistical type inference. In Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA 2022). Association for Computing Machinery, New York, NY, USA, 239–250.
[33] Li, Siqi, **Xiaofei Xie**, Yun Lin, Yuekang Li, Ruitao Feng, Xiaohong Li, Weimin Ge, and Jin Song Dong. "Deep Learning for Coverage-Guided Fuzzing: How Far are We?." IEEE Transactions on Dependable and Secure Computing (2022).
[34] Wu, Bozhi, Shangqing Liu, Ruitao Feng, **Xiaofei Xie**, Jingkai Siow, and Shang-Wei Lin. "Enhancing Security Patch Identification by Capturing Structures in Commits." IEEE Transactions on Dependable and Secure Computing (2022).
[35] Xueyang Li, Shangqing Liu, Ruitao Feng, Guozhu Meng, **Xiaofei Xie**, Kai Chen and Yang Liu. “TransRepair: Context-aware Program Repair for Compilation Errors” In Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering 2022.
[36] Shi, Xiuhan, **Xiaofei Xie**, Yi Li, Yao Zhang, Sen Chen, and Xiaohong Li. "Large-scale analysis of non-termination bugs in real-world OSS projects." In Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, pp. 256-268. 2022.

[37] Zhang, Yao, **Xiaofei Xie**, Yi Li, Yun Lin, Sen Chen, Yang Liu, and Xiaohong Li. "Demystifying Performance Regressions in String Solvers." IEEE Transactions on Software Engineering (2022).
[38] Hu, Qiang, Yuejun Guo, Maxime Cordy, **Xiaofei Xie**, Lei Ma, Mike Papadakis, and Yves Le Traon. "An empirical study on data distribution-aware test selection for deep learning enhancement." ACM Transactions on Software Engineering and Methodology (2022).
[39] Zhang, Yao, **Xiaofei Xie**, Yi Li, Sen Chen, Cen Zhang, and Xiaohong Li. "EndWatch: A practical method for detecting non-termination in real-world software." In 2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE), pp. 686-697. IEEE, 2023.
[40] Yu, Jiongchi, Yuechen Wu**, Xiaofei Xie**, Wei Le, Lei Ma, Yingfeng Chen, Jingyu Hu, and Fan Zhang. "GameRTS: A regression testing framework for video games." In 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE), pp. 1393-1404. IEEE, 2023.
[41] Li, Zhuo, Xiongfei Wu, Derui Zhu, Mingfei Cheng, Siyuan Chen, Fuyuan Zhang, **Xiaofei Xie**, Lei Ma, and Jianjun Zhao. "Generative Model-Based Testing on Decision-Making Policies." In 2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE), pp. 243-254. IEEE, 2023.
[42] Cheng, Mingfei, Yuan Zhou, and **Xiaofei Xie**. "Behavexplor: Behavior diversity guided testing for autonomous driving systems." In Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis, pp. 488-500. 2023
[43] Liu, Shangqing, **Xiaofei Xie**, Jingkai Siow, Lei Ma, Guozhu Meng, and Yang Liu. "Graphsearchnet: Enhancing GNNs via capturing global dependencies for semantic code search." IEEE Transactions on Software Engineering (2023).
[44] Shangqing Liu, Bozhi Wu, **Xiaofei Xie**, Guozhu Meng, and Yang Liu. 2023. ContraBERT: Enhancing Code Pre-Trained Models via Contrastive Learning. In Proceedings of the 45th International Conference on Software Engineering (ICSE '23)
[45] Wang, Longtian, **Xiaofei Xie**, Xiaoning Du, Meng Tian, Qing Guo, Zheng Yang, and Chao Shen. "DistXplore: Distribution-Guided Testing for Evaluating and Enhancing Deep Learning Systems." In Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, pp. 68-80. 2023.
[46] Zhi, Yuhan, **Xiaofei Xie**, Chao Shen, Jun Sun, Xiaoyu Zhang, and Xiaohong Guan. "Seed Selection for Testing Deep Neural Networks." ACM Transactions on Software Engineering and Methodology (2023).