

Explainable AI for SE: Challenges and Future Directions



EXPLAINABLE
**FOR SOFTWARE
ENGINEERING**

Chakkrit Tantithamthavorn^{ID}, Monash University

Jürgen Cito, TU Wien

Hadi Hemmati, York University

Satish Chandra, Google

IN RECENT YEARS, artificial intelligence/machine learning (AI/ML) have been widely used in software engineering (SE) to improve developer productivity, software quality, and decision making. This includes well-known tools for code completion (for example, GitHub's Copilot) but also code search; automated task

CodeGPT, CodeT5, UniXCoder, Codex, GPT3, and so on.

Therefore, Explainable AI (XAI) for SE (XAI4SE) is a pressing concern for the software industry and academia.¹ In the light of predictions made in SE contexts,^{1,2} practitioners would like to know: Why has this code been generated? Why

SE tasks (for example, explaining an Agile story point estimation,³ explaining a defect prediction,⁴ localizing which lines are likely to be defective,⁵ and providing actionable advice on how to fix a defect^{6,7}). They can be broadly categorized into white-box and black-box techniques. White-box techniques are tailored to specific ML models and exploit the internals of the model. Attention mechanisms and integrated gradients are popular white-box techniques for explaining deep learning models. Attention is used as a means for generating explanations by assigning weights to different parts of the input, indicating an attribution of importance for the prediction. Integrated gradients involve creating interpolated inputs and evaluating the model. However, these techniques are not naturally suitable for models of code as there is no natural interpolation between a zero token and a token in the input code.

Many black-box, that is, model-agnostic, techniques are based on perturbation mechanisms. While there are certainly variations in these methods, the basic idea is to modify the input data in a systematic way and observe the difference in the model's output to help understand which parts of the input are most important for the model. There are different types of perturbation-based explanation techniques, including Local Interpretable Model-Agnostic Explanations (LIME)⁸ and Shapley Additive Explanations (SHAP).⁹

Other mechanisms, which could be considered perturbation based, have used delta-debugging techniques to comprehend source code models by reducing a program to a minimum set of statements that still retain the model's initial output. The underlying assumption is that the remaining statements

Explainable AI for SE is a pressing concern for the software industry and academia.

recommendation; automated developer recommendation; automated defect/vulnerability/malware prediction, detection, localization, and repair; and many other purposes.

The Importance of Explainable AI for SE

Although AI/ML have opened up the possibilities of creating innovative developer tools that were hard to build using non-ML methods, their impact will not be fully realized until developers trust the tools. A lack of explainability of AI/ML often leads to a lack of trust in the predictions of AI/ML models in SE, which in turn hinders the adoption of AI/ML models in real-world software development practices. This problem is especially more pronounced for deep learning-based models. Most model techniques for creating ML-based tools are based on large language models that, aside from containing an enormous (and growing!) number of parameters, are black box and complex in nature: CodeBERT, GraphCodeBERT,

is this person best suited for this task? Why is this file predicted as defective? Why is this task required the highest development effort? and so forth.

XAI in a Nutshell

XAI is a set of processes and methods that allows human users to comprehend the results and output created by ML algorithms to enable trust. The explainability of AI/ML algorithms can be achieved by 1) making the entire decision-making process transparent and comprehensible and 2) explicitly providing an explanation for each decision since an explanation is not likely applicable to all decisions. In addition, explanations of AI/ML systems can be presented in various forms (for example, natural text, decision trees, rules, or important factors) to serve the various goals of different stakeholders (for example, developers, managers, testers, lawyers, and CEOs).

There is a multitude of XAI techniques that have been used to support understanding models for various

are the crucial signals being detected by the model. The objective is to identify the crucial features that affect the model's output.

Overview of the Special Issue Articles

The aim of this special issue is to be a venue for sharing practical experiences and research results on the new emerging research direction of XAI4SE. The special issue includes two articles that address two important challenges (that is, reliability and trustworthiness). Next, we give a brief overview of the contents of the special issue.

"Toward Reliable Software Analytics: Systematic Integration of Explanations From Different Model-Agnostic Techniques," the article by Gichan Lee and Scott Uk-Jin Lee,^{A1} addresses the reliability challenge of model-agnostic techniques. Recently, software analytics began to explain the reasons behind the predictions of ML and AI models for various aspects of software projects using model-agnostic techniques derived from the XAI domain. However, there is no guarantee that different model-agnostic techniques will generate consistent explanations for the same predictions. Therefore, practitioners may obtain different insights depending on the technique they use. This article discusses the problem caused by the inconsistent explanations generated from different model-agnostic techniques. In addition, the authors propose a method to integrate inconsistent explanations to derive information that can provide more useful and reliable explanations to practitioners.

"Don't Lie to Me: Avoiding Malicious Explanations with STEALTH," the article by Lauren Alvarez and Tim Menzies,^{A2} addresses the

trustworthiness challenge of model-agnostic techniques. They propose STEALTH, which is a method for using some AI-generated models without suffering from malicious attacks (that is, lying) or associated unfairness issues. After recursively biclustering the data, the STEALTH system asks the model a limited number of queries about class labels. STEALTH asks so few queries (one per data cluster) that malicious algorithms cannot 1) detect its operation or 2) know when to lie.

We also prepared Sounding Board, "Expert Perspectives on Explainability," with interviews from internationally recognized experts in industry who have deployed large AI/ML systems as part of the SE lifecycle.^{A3} We want to gather their expert perspectives on how their work has shaped the SE landscape and what role explainability has currently and in the future. We conducted interviews with Eddie Aftandilian (principal researcher at GitHub Next) and Vijay Murali (software engineer at Meta). We talked about the broader role of ML/AI for SE; the spectrum of models and software artifacts that are affected; how we could more systematically evaluate explanations; and the role of trustworthiness that explainability enables for the future of AI in SE.

Future Directions

While XAI has shown lots of promise in SE, there are various emerging challenges that remain largely unexplored.¹⁰ Possible research questions include the following.


Designing an Explanation

- What are the needs, motivations, and challenges of XAI4SE?
- Do different stakeholders need different explanations?

- What is the best form of explanations for SE tasks that are most understandable by software practitioners?
- What aspects of psychology, learning theories, cognitive science, and social sciences must be considered when designing an explanation for SE tasks?

Developing and Evaluating XAI4SE techniques

- Can XAI techniques be applied to new SE tasks to serve other purposes, for example, testing, debugging, visualizing, interpreting, and refining AI/ML models?
- Can we use XAI methods to detect and explain potential biases when applying AI tools in SE?
- What is a systematic evaluation framework of XAI techniques for SE tasks?
- How can we evaluate the impact of using XAI techniques in software development practices?
- What are the industrial case studies, experience reports, and lessons learned from XAI4SE in practice?

We look forward to seeing more publications addressing these important research questions. 

Acknowledgment

We express our sincere thanks to the authors and reviewers of all of the high-quality submissions we received for this theme issue. We also thank Editor in Chief Ipek Ozkaya and the *IEEE Software* team for their guidance and support. Chakkrit Tanthamthavorn was partly supported by the Australian Research Council's Discovery Early Career Researcher



CHAKKRIT TANTITHAMTHAVORN is a senior lecturer of software engineering and a 2020 Australian Research Council Discovery Early Career Research Award Fellow in the Faculty of Information Technology, Monash University, Clayton, Victoria 3800, Australia. His current focus is on pioneering an emerging research area of explainable artificial intelligence (AI) for software engineering and inventing many AI/machine learning/deep learning/natural language processing-based technologies to improve developers' productivity and make software systems more reliable and secure, while being explainable to practitioners. Contact him at chakkrit@monash.edu.



JÜRGEN CITO is an assistant professor at TU Wien, 1040 Vienna, Austria. His research interests include machine learning for software engineering applications. Cito received his Ph.D. in computer science from the University of Zurich, Switzerland and was a postdoctoral research scholar at the Massachusetts Institute of Technology. Contact him at juergen.cito@tuwien.ac.at.



HADI HEMMATI is an associate professor in the Department of Electrical Engineering and Computer Science, York University, Toronto, ON M3J1P3, Canada. He is also an adjunct associate professor at Calgary, AB, Canada. His main research interests include automated software engineering (with a focus on software testing, debugging, and repair) and trustworthy artificial intelligence (with a focus on robustness and explainability). His research has a strong focus on empirically investigating software/machine learning engineering practices in large-scale industrial systems. He is a Senior Member of IEEE. Contact him at hemmati@yorku.ca.



SATISH CHANDRA is a research scientist at Google, CA 94043 USA. His research interests include programming languages and software engineering, including program analysis, type systems, software synthesis, bug finding and repair, software testing and test automation, and, most recently, applications of machine learning to developer tools. Chandra received his Ph.D. in computer science from the University of Wisconsin-Madison. He is an Association for Computing Machinery Distinguished Scientist. Contact him at schandra@acm.org.

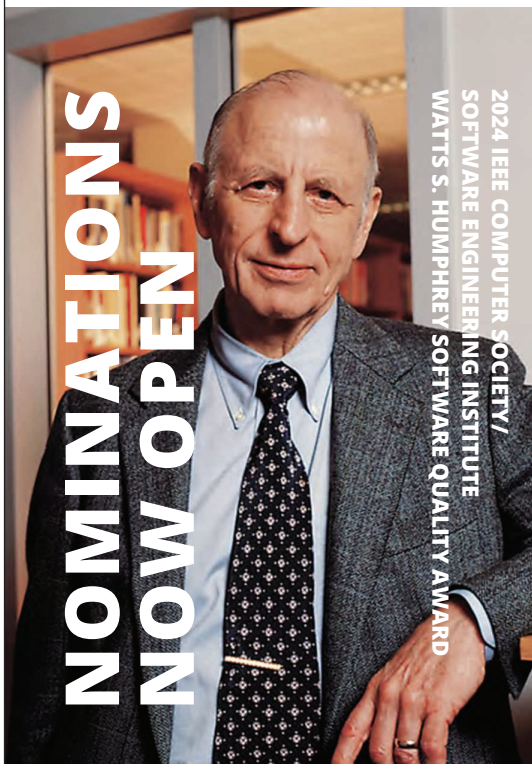
Award (DECRA) funding scheme (Grant DE200100941). Hadi Hemmati is partly supported by the NSERC Alliance (ALLRP 568643-21)–Alberta Innovates Advance Program (Grant 212200865).

References

1. C. Tantithamthavorn and J. Jiarpakdee, "Explainable AI for software engineering," in *Proc. Int. Conf. Automated Softw. Eng. (ASE)*, May 2021. [Online]. Available: <http://xai4se.github.io>
2. H. K. Dam, T. Tran, and A. Ghose, "Explainable software analytics," in *Proc. 40th Int. Conf. Softw. Eng., New Ideas Emerg. Results*, 2018, pp. 53–56, doi: 10.1145/3183399.3183424.
3. M. Fu and C. Tantithamthavorn, "GPT2SP: A transformer-based agile story point estimation approach," *IEEE Trans. Softw. Eng.*, vol. 49, no. 2, pp. 611–625, Feb. 2023, doi: 10.1109/TSE.2022.3158252.
4. J. Jirayus, C. Tantithamthavorn, and J. Grundy, "Practitioners' perceptions of the goals and visual explanations of defect prediction models," in *Proc. Int. Conf. Mining Softw. Repositories (MSR)*, 2021, pp. 432–443, doi: 10.1109/MSR52588.2021.00055.
5. W. Supatsara, P. Thongtanunam, C. Tantithamthavorn, H. Hata, and K. Matsumoto, "Predicting defective lines using a model-agnostic technique," *IEEE Trans. Softw. Eng.*, vol. 48, no. 5, pp. 1480–1496, May 2022, doi: 10.1109/TSE.2020.3023177.
6. J. Jiarpakdee, C. Tantithamthavorn, H. K. Dam, and J. Grundy, "An empirical study of model-agnostic techniques for defect prediction models," *IEEE Trans. Softw. Eng.*, vol. 48, no. 1, pp. 166–185, Jan. 2022, doi: 10.1109/TSE.2020.2982385.

7. C. Pornprasit, C. Tantithamthavorn, J. Jiarpakdee, M. Fu, and P. Thongtanunam, "PyExplainer: Explaining the predictions of just-in-time defect models," in *Proc. 36th IEEE/ACM Int. Conf. Automated Softw. Eng. (ASE)*, Nov. 2021, pp. 407–418, doi: 10.1109/ASE51524.2021.9678763.
 8. M. T. Ribeiro, S. Singh, and C. Guestrin, "'Why should I trust you?' Explaining the predictions of any classifier," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2016, pp. 1135–1144.
 9. S. M. Lundberg and S. I. Lee, "A unified approach to interpreting model predictions," in *Proc. 31st Adv. Neural Inf. Process. Syst.*, 2017, pp. 4768–4777.
 10. A. H. Mohammadkhani, N. S. Bommi, M. Daboussi, O. Sabnis, C. Tantithamthavorn, and H. Hemmati, "A systematic literature review of explainable AI for software engineering," 2023. [Online]. Available: <https://arxiv.org/abs/2302.06065>
 11. A. Vaswani et al., "Attention is all you need," in *Proc. 31st Adv. Neural Inf. Process. Syst.*, 2017, pp. 6000–6010.
- Appendix: Related Articles**
- A1. G. Lee and S. U.-J. Lee, "Toward reliable software analytics: Systematic integration of explanations from different model-agnostic techniques," *IEEE Softw.*, vol. 40, no. 3, pp. 34–42, May/Jun. 2023, doi: 10.1109/MS.2023.3244204.
- A2. L. Alvarez and T. Menzies, "Don't lie to me: Avoiding malicious explanations with STEALTH," *IEEE Softw.*, vol. 40, no. 3, pp. 43–51, May/Jun. 2023, doi: 10.1109/MS.2023.3244713.
- A3. J. Cito, S. Chandra, C. Tantithamthavorn, and H. Hemmati, "Expert perspectives on explainability," *IEEE Softw.*, vol. 40, no. 3, pp. 84–88, May/Jun. 2023, doi: 10.1109/MS.2023.3255663.

Carnegie Mellon University Software Engineering Institute



Since 1994, the SEI and the Institute of Electrical and Electronics Engineers (IEEE) Computer Society have cosponsored the Watts S. Humphrey Software Quality Award, which recognizes outstanding achievements in improving an organization's ability to create and evolve high-quality software-dependent systems.

Humphrey Award nominees must have demonstrated an exceptional degree of **significant**, **measured**, **sustained**, and **shared** productivity improvement.

TO NOMINATE YOURSELF OR A COLLEAGUE, GO TO
computer.org/volunteering/awards/humphrey-software-quality

Nominations due by September 1, 2023.

FOR MORE INFORMATION

resources.sei.cmu.edu/news-events/events/watts