Project Proposal

Xusheng Xiao
xxiao2@ncsu.edu

Project Topic:

Artificial Intelligence in Software Engineering.

Goal:

Software Engineering is a knowledge-intensive activity, presumably requiring intelligence. To reduce human efforts in the activities of software engineering, Artificial Intelligence (AI) techniques, which aims to create computer systems that exhibit some form of human intelligence, are employed to assist or automate various activies of software engineering, such as testing, program analysis, debugging and even self-repair. In this project, I will provide a study of how various AI techniques are used in automating or assiting various activities of software engineering.

Project Description:

Many software engineering activities, such as testing, analysis and debugging, require intensive human intelligence and are error-prone. To reduce human efforts in these activities, AI techniques are used to assist or automate some of the processes in these activities. In this project, we intend to explore the application of AI in software engineering and provide a detailed survey of the existing tools and techniques associating with AI in three important software engineering activities: testing, fault detection and software repair.

1. **AI in Software Testing.** Over the past decades, many AI techniques are applied to assist automated software testing, such as constraint solving [13] used in Dynamic Symbolic Execution [7, 6, 9] for test-input generation, heuristics used to prune search space of test-generation tools [2, 12], and machine learning used in statistical software testing [3] and coverage prediction of testing tools [4]. I plan to study in details how these research works adopt the concepts and techniques of AI to achieve better performance.

2. **AI in Fault Detection.** As the size of complexity of software has grown quickly in past decades, the difficulty of finding and fixng bugs has increased. Recent research works [10, 1] that use AI techniques have advanced the research in reducing the human efforts on fault detection: Shi et al. [10] proposes an approach to first learn the Definition-Use Invariants and then use the learned knowledge of Definition-Use for detecting concurrency and sequential bugs; Baah et al [1] and proposes a new machine-learning technique that performs fault detection for deployed software. I plan to study in details how these research works adopt the concepts and techniques of AI to assist the task of fault detection.

3. **AI in Software Repair.** By adapting the well-known AI techniques, even the most challenging tasks, debugging and self-repair, can be half or even full automated. Genetic programming, an evolutionary algorithm-based methodology in AI, is used and adapted by Weimer et al. [11]' approach to automatically find patches for programs and automatically fix bugs [5]. Inspired by these works, Schulte et al. further propose an approach to study the evolution of assembly code [8] for automated program repair. I plan to investigate these techniques to study how AI can achieve automated software repair.

References

[1] G. K. Baah, A. Gray, and M. J. Harrold. On-line anomaly detection of deployed software: A statistical machine learning approach. In *Proceedings of theThird International Workshop on Software Quality Assurance (SOQUA 2006)*, pages 70–77, Portland, Oregon, November 2006.

[2] S. Bardin and P. Herrmann. Pruning the Search Space in Path-Based Test Generation. In *ICST*, 2009.

[3] N. Baskiotis, M. Sebag, M.-C. Gaudel, and S. Gouraud. A machine learning approach for statistical software testing. In *Proceedings of the 20th international joint conference on Artifical intelligence*, pages 2274–2279, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

[4] B. Daniel and M. Boshernitsan. Predicting effectiveness of automatic testing tools. In *IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, September 2008.

[5] S. Forrest, T. Nguyen, W. Weimer, and C. Le Goues. A genetic programming approach to automated software repair. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, GECCO '09, pages 947–954, New York, NY, USA, 2009. ACM.

[6] P. Godefroid, N. Klarlund, and K. Sen. Dart: directed automated random testing. In *PLDI*, pages 213–223, 2005.

[7] J. C. King. Symbolic execution and program testing. *Commun. ACM*, 19(7):385–394, 1976.

[8] E. Schulte, S. Forrest, and W. Weimer. Automated program repair through the evolution of assembly code. In *Proceedings of the IEEE/ACM international conference on Automated software engineering*, ASE '10, pages 313–316, New York, NY, USA, 2010. ACM.

[9] K. Sen, D. Marinov, and G. Agha. Cute: a concolic unit testing engine for c. In *ESEC/FSE-13: Proceedings of the 10th European software engineering conference held jointly with 13th ACM SIGSOFT international symposium on Foundations of software engineering*, pages 263–272, New York, NY, USA, 2005. ACM.

[10] Y. Shi, S. Park, Z. Yin, S. Lu, Y. Zhou, W. Chen, and W. Zheng. Do i use the wrong definition?: Defuse: definition-use invariants for detecting concurrency and sequential bugs. In *Proceedings of the ACM international conference on Object oriented programming systems languages and applications*, OOPSLA '10, pages 160–174, New York, NY, USA, 2010. ACM.

[11] W. Weimer, T. Nguyen, C. L. Goues, and S. Forrest. Automatically finding patches using genetic programming. *Software Engineering, International Conference on*, 0:364–374, 2009.

[12] T. Xie, N. Tillmann, P. de Halleux, and W. Schulte. Fitness-guided path exploration in dynamic symbolic execution. In *Proc. the 39th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2009)*, June-July 2009.

[13] J. Zhang. Constraint solving and symbolic execution. *Verified Software: Theories, Tools, Experiments: First IFIP TC 2/WG 2.3 Conference, VSTTE 2005, Zurich, Switzerland, October 10-13, 2005, Revised Selected Papers and Discussions*, pages 539–544, 2008.