Software Engineering Seminar

State-Based Fault Localization

Description

Automated fault localization techniques assist developers with the task of pointing out program elements that are most probable to be responsible for a detected error. Over the years, many different techniques have been developed [7].

State-based fault localization techniques comprise several different strategies to locate software bugs by monitoring and manipulating the state of a program at various points in its executions, and to then draw conclusions about the cause of the error from the executions' results. Notable techniques in this category include, e.g., the *delta debugging* approach [8] and its extension – the *cause transition* technique [1, 2]. There also exist techniques that alter the outcome of decisions during program execution [9, 6, 5, 4]. A very recent approach uses a decision tree learner to learn input features that are responsible for a specific program behavior [3].

The goal of this topic is to examine and discuss the current state of the art of state-based fault localization techniques, to evaluate their relevancy and to analyze/estimate their capabilities compared to other fault localization techniques.

References

[1] Holger Cleve and Andreas Zeller. Locating causes of program failures. In *Proceedings of the 27th international conference on Software engineering - ICSE '05*, Proceedings of the 27th International Conference on Software Engineering, pages 342–351, New York, NY, USA, 2005. IEEE, ACM Press.

[2] Neelam Gupta, Haifeng He, Xiangyu Zhang, and Rajiv Gupta. Locating faulty code using failure-inducing chops. In *Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering*, pages 263–272, 2005.

[3] Alexander Kampmann, Nikolas Havrikov, Soremekun Ezekiel, and Andreas Zeller. When does my program do this? learning circumstances of software behavior. 2020.

[4] Feng Li, Wei Huo, Congming Chen, Lujie Zhong, Xiaobing Feng, and Zhiyuan Li. Effective fault localization based on minimum debugging frontier set. In *Proceedings of the 2013 IEEE/ACM International Symposium on Code Generation and Optimization (CGO)*, pages 1–10. IEEE, 2013.

[5] Tao Wang and Abhik Roychoudhury. Automated path generation for software fault localization. In *Proceedings of the 20th IEEE/ACM int. Conference on Automated software engineering*, pages 347–351, 2005.

[6] Xiaoyan Wang and Yongmei Liu. Automated fault localization via hierarchical multiple predicate switching. *Journal of Systems and Software*, 104:69–81, 2015.

[7] W. Eric Wong, Ruizhi Gao, Yihao Li, Rui Abreu, and Franz Wotawa. A survey on software fault localization. *IEEE Transactions on Software Engineering*, 42(8):707–740, 2016.

[8] Andreas Zeller. Isolating cause-effect chains from computer programs. *ACM SIGSOFT Software Engineering Notes*, 27(6):1–10, 2002.

[9] Xiangyu Zhang, Neelam Gupta, and Rajiv Gupta. Locating faults through automated predicate switching. In *Proceeding of the 28th int. conference on Softw. engineering - ICSE '06*, pages 272–281. ACM Press, 2006.

Contacts

Simon Heiden (`heiden@informatik.hu-berlin.de`)
Software Engineering Group
Institut für Informatik
Humboldt-Universität zu Berlin