Bachelor/Master Thesis Topic

Identification of Crash Traces with Symbolic Execution

Motivation and Background
Seo and Kim [1] introduced their work on predicting recurring crash stacks. They identify crash traces in a crash bucket that do not include a fixed bug location. Therefore, they find missed crash traces in a resolved crash bucket, which is supposed to be fixed. Presumably, there are more program paths that lead to the crash location without reaching the fixed bug location and that are not reported with a crash trace. For example, Manevich et al. [2] showed a technique to identify program paths that lead to a specific failure. Such a technique might be improved by using symbolic execution. The Java verification tool Java PathFinder (JPF) [3] can be used to automatically generate test cases.

Goals
This project should identify ideas on how to solve the problem of the identification of crash traces with symbolic execution. Additionally, the approach should be implemented and evaluated on open source applications.

Description of the Task
The specific tasks are:
- Understand the approach by Seo and Kim.
- Develop ideas on how to identify crash traces for a given crash location with symbolic execution.
- Implement the procedure as an extension of JPF.
- Perform experimental evaluation and comparison of the implemented idea on set of established benchmarks.

Research Type
Theoretical Aspects:  *****
Industrial Relevance:  *****
Implementation:  ******

Prerequisite
The student should be enrolled in the bachelor/master of computer science program, and has completed the required course modules to start a bachelor/master thesis.

Skills required
Programming skills in Java or C++, understanding of, or willingness to learn, the software engineering methods like symbolic execution and patch testing needed for the project.

Contacts
Yannic Noller (noller@informatik.hu-berlin.de)
Software Engineering Group, Institut für Informatik, Humboldt-Universität zu Berlin

References