# Project Proposal

Abhiroop Kodandapursanjeeva

My project involves investigating the use of multithreading in mutation testing tools. The reason for this is to simply make the mutation testing process more time-efficient by utilizing more computing power.

One open-source project for mutation-testing for Java is called *muJava*. Although it was considered an improvement over older mutation testing tools, it has since fallen behind other tools such as *PITest*. After inspecting the code of *muJava*, it becomes clear the developers used threads only for one purpose: to detect whether a test was taking abnormally long due to problematic mutations. However, only one thread runs at a time, thus there is no performance boost. In addition, there is no use of multithreading in the mutant generation portion. It would be beneficial to improve the performance of this tool and make it more "modern" by using multiple threads when appropriate. A paper by Mateo & Usaola [1] describes in detail the performance differences between various parallel methods for mutation test execution. They include a novel approach, which shows to be only marginally better than a simpler approach. Specifically, their novel approach involves assigning each thread a fixed amount of work initially, and any leftover work to be pulled from a common pool by a thread when it is available. The simpler approach is to allow threads to pull work from the common pool without any initial work pushed to them. This is only used for the test execution portion, though similar methods may be implemented for the mutant generation portion as well.

Performance can be measured by allowing the tool to run for some number of tests, and measuring the time to finish. The original and improved versions can be compared this way.

Citations

[1] Mateo, P.R. and Usaola, M.P. (2013), Parallel mutation testing. Softw. Test. Verif. Reliab., 23: 315-350. doi:10.1002/stvr.1471