An Evaluation of Data Race Detectors Using Bug Repositories  
Jochen Schimmel, Korbinian Molitorisz, Walter F. Tichy

# Introduction:

This paper discusses on the race conditions that occurs in Multithreaded Software. There are many race condition detectors in the market. The authors of this have presented the result by analyzing four different detectors for real time applications. They claim that the race conditions can be prevented if these detectors work as expected (efficiently). The proposed models sum up to an accuracy of 92% with the individual giving at most accuracy of 50%. It is quite common to find the race conditions in rum-time. But the authors took real life bug applications and tried to find the unintentional races in their use cases selected. They have proposed ways to efficiently prevent the race during the software development phase itself. This would save on from the embarrassments of detecting the race conditions during run time. Each data race can be classified into four different sub categories. They can be categorized as atomicity violation, wrong use of JAVA library, if-race and bad optimization. The errors can be produced in different order. Again here, the errors can be further categorized into two more categories, one due to execution of parallel threads and the other caused due to multiple executions.

# Conclusion:

The authors went on to discussing more on brief introduction to detectors and illustrated results of individual accuracy. They conducted a performance analysis by combining the all the detectors and determined that they give an accuracy of 92%. However, there was a trade off on using the individual detector as they only gave an accuracy of 50 %. The advent of Multi threaded programming is prevalent everywhere and it becomes a dire necessity to avoid race conditions. If these race conditions can be identified and corrected during the software development phase, they help during the deployment. All this speaks up why race condition detectors are required.