Who Really Cares If the Program Crashes?

Willem Visser

Computer Science Division
Department of Mathematical Sciences
University of Stellenbosch
South Africa

Abstract. After spending eight years at NASA doing research in model checking and testing, I decided it would be a nice change of scene to see how software is being developed in a fast-paced technology start-up (SEVEN Networks). Of course I was secretly hoping to solve all their testing problems with the cool research techniques from the verification and testing community.

At NASA software is written once, for the most part run once, and if it fails there are serious (even life-threatening) consequences. Clearly this is a fruitful hunting ground for advanced verification and testing technology. At SEVEN, on the other hand, code is maintained and adapted for years and the same programs execute thousands of times a second on various platforms. Failures are plentiful, but they only become important once they start to impact service level agreements with the paying customers; i.e. when they start to have a negative impact on the bottom-line. Failures are not necessarily crashes either, it is much more likely to be a performance bottle-neck that eventually causes a system-wide failure. What does the verification and testing community have to offer in this arena, bearing in mind there are very few "NASA"s and very many "SEVEN"s in the world?

This talk is about what I learned in the past two years at SEVEN and how it is influencing my current research. In particular I will explain why I ran a model checker on SEVEN code just once, used a static analysis tool only once as well, the reasons why model based testing is no longer used at SEVEN, why I am no longer certain deadlocks are so important (but races are), why SQL is a useful debugging aid and why performance analysis is important. I will also highlight some of the more interesting errors I encountered at SEVEN and why our current tools cannot find most of these.