Programmers Ain't Mathematicians, and Neither Are Testers

Jeff Offutt

George Mason University, USA

Formal methods have been developed for decades. An early promise was that we could use formal methods to "prove" our programs correct. We have also tried to use formal methods to completely specify functional behavior of programs and to partially specify specific aspects of software behavior. Research into formal methods have led to weaker techniques to "model" functional behavior, less completely and less precisely, but in ways that are easier to use. Despite these years of activity, formal methods are still seldom used in industry. As software engineering researchers, we are compelled to take the view that there must be a path from our research to actual use in industry, where real software developers use our ideas to help create real, and better, software. Thus we must ask, are formal methods a solution in search of a problem?

In this talk, I will draw a distinction between mathematical thinking, which is required to develop formal models of software behavior, and engineering thinking, which is required to develop working software. In a broad sense, formal methods are used to create abstractions, and abstraction should be used to handle complexity, not to ignore it. The talk will explore this distinction and suggest specific ways formal methods can be successfully integrated into software development, testing and education. Mathematicians who specify software models and design software tests should delve into this abstraction, using formal engineering methods to help real programmers build real software, faster, better and cheaper.