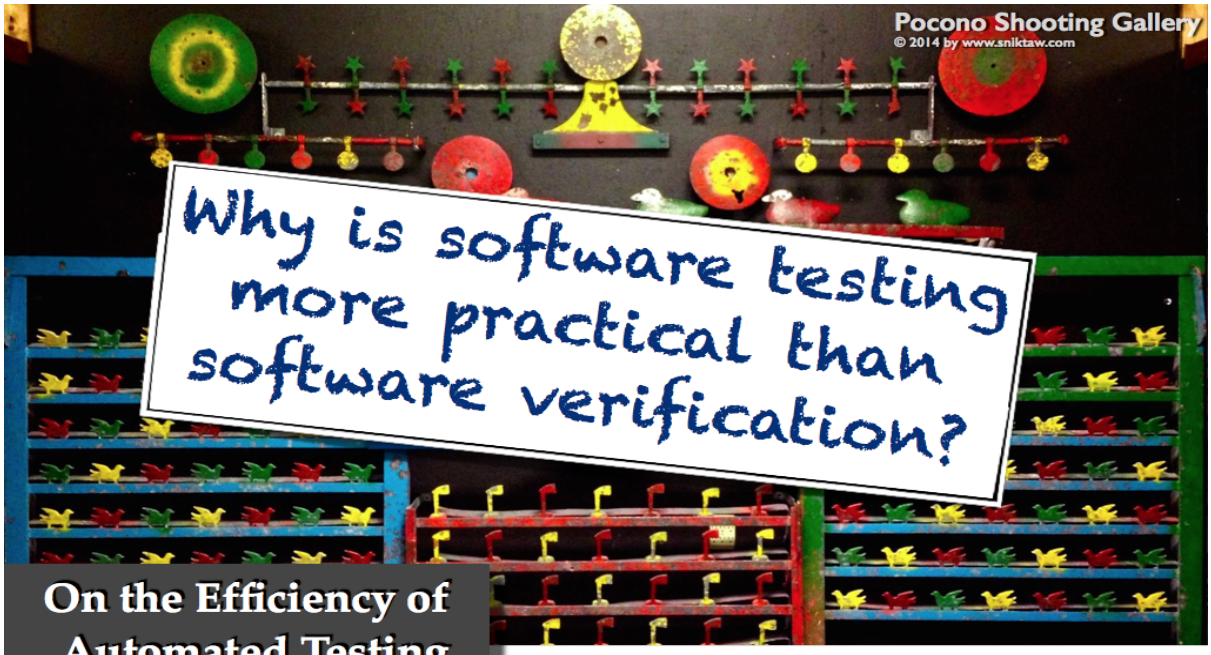

On the Efficiency of Automated Testing

Marcel Böhme and Soumya Paul
@FSE'14

NUS National University of Singapore
School of Computing

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Bet at the Carnival Shooter. Randolph and Sandy are at the carnival and see a shooting booth where one can shoot plastic stars for prices. Randolph proposes a bet: *There are 100 stars on the board, whoever hits 90 first wins*! There is infinite ammo and no need to recharge. All of the board is covered in stars and each star can be hit only once.

Randolph's shooting strategy. Randolph has never shot in his life and decides to shoot randomly at the board keeping his finger on the trigger. Every point on the board has the same probability to be hit. He takes 100ms per shot.

Sandy's shooting strategy. What Randolph does not know is that Sandy has trained shooting at plastic stars for several years and won several shooting competitions. In fact, she is the most effective shooter. Every shot hits a star. However, because she has to home in on her target, compute the speed and direction of the wind, and remain focussed, she takes c milliseconds per shot.

What is the maximum time c_0 that Sandy can take for each shot to be expected to win the bet? Some simple probabilistic analysis reveals that $c_0 = 0.255$sec. Recall, that Randolph takes only 0.1 seconds per shot. What if we change the bet to hitting 99 stars? Then, $c_0 = 0.463$sec. Wow! So, on the one hand we have Randolp's "dumb" random strategy and on the other hand Sandy's "smart" and most effective strategy. Yet, if Sandy is not fast enough, Randolph will win the bet. Note that Sandy hits all 100 stars in $100 \cdot c$ seconds while for Randolph the expected number of stars hit will *never* reach 100. The bet, however, is not about hitting *all* stars but rather some *portion* of them. Similarly, the problem of software testing is not to show the absence of errors for all inputs but rather for some portion of them.

The work above has been published at FSE'14. At Dagstuhl, I would like to discuss a follow up of this work which is to be submitted to ESEC/FSE'15. We present some surprising results, such as: For every systematic testing technique, there exists a program large enough that random testing will always be more efficient. What other implications does this work have on testing, in practice? Do you have practical insights to improve the (very simple) probabilistic framework? Check out: http://bit.ly/1CjoAkF