NATIONAL UNIVERSITY OF SINGAPORE

School of Computing

PH.D DEFENCE - PUBLIC SEMINAR

Title: **Static Analysis Driven Testing of Performance and Energy-consumption Properties of Software**

Speaker: Mr Abhijeet Banerjee

Date/Time: 2 June 2016, Thursday, 04:30 PM to 06:00 PM

Venue: Video Conference Room, COM1-02-13

Supervisor : Dr Abhik Roychoudhury, Professor, School of Computing

Abstract:

Software testing is the process of evaluating the properties of a software. Properties of a software can be divided into two categories:
functional properties and non-functional properties. Properties that influence the input-output behavior of the software can be categorized as functional properties. On the other hand, properties that do not influence the input-output behavior of the software directly can be categorized as non-functional properties. In context of real-time system software, testing functional as well as non functional properties is equally important. Over the years considerable amount of research effort has been dedicated in developing tools and techniques that systematically test various functional properties of a software.

However, the same cannot be said about testing non-functional properties. Systematic testing of non-functional properties is often much more challenging than testing functional properties. This is because non-functional properties not only depends on the inputs to the program but also on the underlying hardware. Additionally, unlike the functional properties, non-functional properties are seldom annotated in the software itself. Such challenges provide the objectives for this work.

The primary objective of this work is to explore and address the major challenges in testing non-functional properties of a software. To attain this objective, we have designed a technique that can be summarized into four key steps (i) identifying scenarios for sub-optimal non-functional behavior (ii) static analysis to identify potential program points that may lead to such sub-optimal non-functional behavior (iii) representing sub-optimal non-functional behavior by means of assertions, at appropriate program points and finally, (iv) dynamic exploration of these assertions, guided by a well-defined coverage metric, in order to generate sub-optimal non-functional behavior revealing test-cases. It is worthwhile to note that in our technique, generation of assertions (in step iii) is done in an automated fashion. In this work, we have presented instantiations of our technique for specific applications such as

performance-stressing test-input generation for caches and energy-inefficiency revealing test-input generation for mobile apps. We also present a couple of follow-up works on energy-aware code re-factoring and energy-aware debugging to extend the support for energy-aware programming for mobile apps