ABSTRACT: Mobile platforms commonly support an event-driven model of concurrent programming. In an event-driven system, the flow of a program is controlled by asynchronous events. Events processed sequentially in the same thread can be logically concurrent to each other, as they may not be ordered by any programmer-specified ordering operations. The lack of programmer-defined order between multiple noninterchangeable concurrent events — that is, only certain execution orders between these events yields correct results — leads to a new class of atomicity violation and order violation bugs in event-driven programs.

Unfortunately, the state of the art for detecting concurrency errors in event-driven systems is significantly weaker than that in traditional thread-based systems. This thesis aims to fill this important gap by developing models, algorithms and tools that aid programmers to analyze and diagnose event-driven programs to improve software reliability. Specifically, this thesis presents the following three techniques to detect concurrency errors in event-driven programs:

1. A new causality model for event-driven program is defined to infer ordering invariants between events across different executions.

2. An efficient and scalable single-pass algorithm to identify concurrent asynchronous events that may lead to concurrency errors.

3. A dynamic event precondition race detector to help programmers to detect and avoid a wide range of atomicity violations and order violations.

The techniques we have developed are broadly applicable to a majority of event-driven platforms. To translate our techniques into real-world impact, we develop a set of tools in the context of Android to help build up a more robust and reliable platform for mobile computing.

Chair: Prof. Satish Narayanasamy