**New Course Proposal: Special Topics in Computer Engineering (CPEN400P)**

**Course Title**: Program Analysis for Reliability and Security Engineering (PARSE)

**Proposer**: Karthik Pattabiraman (karthikp@ece.ubc.ca), Professor, ECE Dept., UBC.

**Motivation:** Software today pervades every aspect of our society, and is used in many critical scenarios from financial systems to medicine and automated driving. The consequences of a failure or malfunctioning of software are dire, and can lead to loss of money and human lives. Further, software often has to satisfy various requirements beyond functional correctness, such as reliability, security, quality and performance, in order to be deployed in practical scenarios. Therefore, it is challenging to ensure the quality and reliability of large-scale software systems.

This course will investigate the principles of systematic techniques to analyze large-scale software systems, and ensure that they satisfy their requirements. While functional correctness can be ensured to some extent with software testing techniques, testing cannot typically ensure adherence to attributes beyond functional correctness. Further, testing by itself is necessarily incomplete, and hence cannot even guarantee the functional correctness of the software. Therefore, this course will investigate techniques *beyond traditional software testing methods*. Because of the large scale of software systems today, it is challenging to apply techniques that require manual intervention. The course will thus focus on scalable and automated techniques.

**Topics covered:** The exact topics to be covered in the course will vary on a yearly basis, as some of the techniques are still in the research phase. Any technique for analyzing software reliability and security that is automated and scalable beyond traditional testing is relevant to the course. For the initial offerings of the course, we will focus on four classes of techniques:

1. Static Analysis Techniques to analyze the program (source) code prior to its deployment.
2. Dynamic Analysis Techniques to find violations that arise during the program execution.
3. Fault Injection/Fuzz testing Techniques to find corner cases that lead to program failures.
4. Program monitoring and Repair, to track issues in the field, and correct them on the fly.

**Pre-reqs:** 1. CPEN 221: Principles of software construction, and 2. CPEN 321: Software Engg.

**Evaluation Methods:** The course will have both exams and hands-on programming assignments that involve building automated analysis tools. The breakdown is as follows:

**50% -** Five programming assignments that each require significant tool building and analysis
**40%** - Midterm and final exam (e.g., 15% - Midterm exam, and 25% - Final exam)
**5%** - Class participation in-class and online forum (e.g., Piazza)
**5%** - Programming proficiency test as the course requires working knowledge of C/C++/Java.

The assignments will involve modifying real-world software tools such as Clang/LLVM, PIN, Valgrind, AFL, LLFI etc. They will require significant programming in C/C++/Java languages.