Timothy Alec DuPont Henderson

Education

PhD in Computer Science, Case Western Reserve University, GPA 4.0 2011 - May 2017
Proposed Thesis: *Frequent Subgraph Sampling and its Software Engineering Applications.*
Research Areas: *Program Analysis, Data Mining, Sampling, Databases, Distributed Computing.*

BA in Computer Science, Case Western Reserve University Dec. 2010

Papers

A. Durmaz[†], **T. A. D. Henderson**[†], D. Brubaker, and G. Bebek. "Frequent Subgraph Mining of Personalized Signaling Pathway Networks Groups Patients with Frequently Dysregulated Disease Pathways and Predicts Prognosis." *PSB* 2016. Currently Under Review. [†]**Co-First Author**

T. A. D. Henderson, and A. Podgurksi. "Sampling Code Clones from Program Dependence Graphs with GRAPLE" Accepted at SWAN 2016.

G. Shu, B. Sun, **T. A. D. Henderson**, and A. Podgurski. "JavaPDG: A New Platform for Program Dependence Analysis," *ICST* 2013, pp. 408-415.

Selected Work Experience

Case Western Reserve University Instructor Fall 2014
Instructor for EECS 337 Introduction to Compiler Design and Implementation. The course covers: formal languages, lexing, parsing, intermediate language design and generation, control flow analysis, data flow analysis, type checking, run time concerns, and x86 assembly generation. The students implement a compiler for a small language I designed.

Mobile Defense Inc. Software Engineer January 2013 - May 2015
Lead a team implementing a program analysis engine for bytecode for the Dalvik Virtual Machine (Android). Reduced cost by leading a migration to Amazon EMR. Implemented a web service to database static analysis findings for all Android applications. Lead the work on a malware scanning system. Lead the transition to Ansible to simplify our process for provisioning, configuring and deploying servers on AWS. I participated in designing and optimizing the hiring process for new college graduates including: creating a candidate identification and screening process, creating technical work sample questions, and standardizing evaluation criteria.

Cigital Inc. Intern Summer/Fall 2009 and Summer 2010
Worked on a automation system for commercial static analysis tools. Designed, and built a custom Object Relational Mapping, and Query Language for security findings across a variety of formats. Researched comparing security findings across tools, project versions, and assessments.

Community

Organizer, Speaker CWRU Hacker Society (hacsoc.org) 2009 - Present
Hacker Society provides weekly technical talks to the community of Case Western Reserve. I ran the lecture series in: 2009 and in 2015. I personally have spoken on a variety of topics including: Regular Expression Engines, Context Free Grammars, Parsers, Program Analysis, Web Application Security, Secure Cookie Sessions, Linear Hashing, etc.

Selected Programming Languages and Technologies

Python, Go, Java, C, Ruby, x86, Linux, MySQL, Hadoop, Ansible, Fabric, Pyramid, SqlAlchemy.

Statement on Research

My academic research is in the area of Software Engineering. Software Engineering develops tools and processes to create higher quality software systems. The field itself was born out of a software "crisis" in the 1960s in the NATO defense community. In recent years, developments in program analysis have lead to the emergence of high quality commercial static analysis tools for widely used programming languages.

However, despite the availability of automated program verification for interesting program properties, usage has remained restricted to specialized domains. Most development teams today do not write formal specifications or correctness properties for their business logic. Instead, they rely on testing. Unfortunately, testing is necessarily optimistic and cannot ensure a program is bug free. Program verification, when it is used, is employed only to check very general properties, such as the absence of buffer overflow errors.

There are three major problems today preventing the widespread adoption of software analysis tools for verification purposes. First, it is difficult to write correctness properties for programs. My research in specification mining addresses this problem by learning program correctness properties. Second, false positive warnings from static analysis systems remain a persistent problem. My investigations into test case generation determines the accuracy of a static analysis warning. Finally, bug finding tools produce an abundance of findings. Some findings are duplicates, some are false positives and some are actual faults in the program. In order make the analyst more efficient, I am creating a finding triage system which clusters results together affording the analyst the ability to examine groups of findings.

Statement on Teaching

I enjoy teaching both practicing programmers and students here at Case Western Reserve University. My approach whether teaching in an industry setting or in the classroom is the same: a focus on clear exposition with worked examples of key concepts. Lecturing is a form of performance and I try to avoid reliance on highly structured visual aids. It is important to establish a rapport with your audience and I find when I rely on a slide deck to convey my material the connection is lost.

I have lectured on a variety of subjects including: programming basics, software craftsmanship, operating systems, databases, compilers, and program analysis. My lecturing opportunities have come through a number of avenues including: instructing EECS 337 Compilers Design, guest lecturing, departmental colloquia, recitations as a teaching assistant, ACM and Hacker Society meetings, conferences, and internal company training sessions.

When teaching complex concepts such as automatons or grammars, I make heavy utilization of hand drawn diagrams to explain concepts. By drawing, or completing a partially drawn diagram, in front of the class I give the student time to process how the diagram works. They can also see how it is constructed and in what order different components connect. This provides them an intuitive understanding of the concepts involved. When I was a student, the best professors I had utilized the blackboard or document camera and I shamelessly emulate them in my lectures.

I have a learning disability myself so I understand the importance of teaching a concept in a variety of ways. I also understand the importance of accommodating to the best of my abilities the needs of each student in the classroom. When a student is having trouble grasping a concept I try and step back and analyze why they are having difficultly. Is it because they lack background material? Is my explanation difficult for them to understand? Am I employing cultural idioms the student is unfamiliar with? Do they simply need more practice? Whatever a student's difficulty I try my best to support them in overcoming it.

Assignments, projects and homework are an important part of the learning process. Many things in computer programming can only be learned by doing. In the past I designed both written homeworks and programming projects. My written homework focuses on concepts and attempts to re-enforce key theoretical concepts.

The programming projects have a variety of purposes depending on the context. In a software craftsmanship class, the focus is on the construction and formulation of the program itself. Its design and the details like loop construction, the appropriate use of control flow structures, and prop er decomposition into procedures.

However, in an upper level class, such as compiler design, I expect the basics to be correct from the beginning and focus on the algorithmic concerns and the challenges of the problem itself. I also begin to emphasize software engineer concerns such as testing, team work, proper use of version control, and planning.

My grading philosophy reflects my philosophy on the design of the assignments. I seek to provide an honest assessment of the student's grasp of the key learning points of the assignment. The purpose of the grade is not to judge how good the student is but how well they understand particular concepts and ideas. I do not believe in force ranking students or using grades to compare the students to each other. Grades are for the benefit of the student for them to assess their progress in mastering the subject.