Toward Productivity Improvements in Programming Languages through Behavioral Analytics

By: Patrick Daleiden
Friday, May 1, 2020 at 10 am
Zoom Link | Password: 976784

Dr. Andreas Stefik
Advisory Committee Chairperson
Associate Professor
Department of Computer Science
University of Nevada, Las Vegas

Computer science knowledge and skills have become foundational for success in virtually every professional field. As such, productivity in programming and computer science education is of paramount economic and strategic importance for innovation, employment and economic growth. Much of the research around productivity and computer science education has centered around improving notoriously difficult compiler error messages, with a noted surge in new studies in the last decade. In developing an original research plan for this area, this dissertation begins with an examination of the Case for New Instrumentation, drawing inspiration from automated data mining innovations and corporate marketing techniques in behavioral analytics as a model for understanding and prediction of human behavior. This paper then develops and explores techniques for automated measurement of programmer behavior based on token level lexical analysis of computer code. The techniques are applied in two empirical studies on parallel programming tasks with 88 and 91 student participants from the University of Nevada, Las Vegas as well as 108,110 programs from a database code repository. In the first study, through a re-analysis of previously captured data, the token accuracy mapping technique provided direct insight into the root cause for observed performance differences comparing thread-based vs. process-oriented parallel programming paradigms. In the second study comparing two approaches to GPU programming at different levels of abstraction, we found that students who completed programming tasks in the CUDA paradigm (considered a lower level abstraction) performed at least equal to or better than students using the Thrust library (a higher level of abstraction) across four different abstraction tests. The code repository of programs with compiler errors was gathered from an online programming interface on curriculum pages available in the Quorum language (quorumlanguage.com) for Code.org’s Hour of Code, Quorum’s Common Core-mapped curriculum, activities from Girls Who Code and curriculum for Skynet Junior Scholars for a National Science Foundation funded grant entitled Innovators Developing Accessible Tools for Astronomy (IDATA). A key contribution of this research project is the development of a novel approach to compiler error categorization and hint generation based on token patterns called the Token Signature Technique. Token Signature analysis occurs as a post-processing step after a compilation pass with an ANTLR LL* parser triggers and categorizes an error. In this project, we use this technique to i.) further categorize and measure the root causes of the most common compiler errors in the Quorum database and then ii.) serve as an analysis tool for the development of a rules engine for enhancing compiler errors and providing live hint suggestions to programmers. The observed error patterns both in the overall error code categories in the Quorum database and in the specific token signatures within each error code category show error concentration patterns similar to other compiler error studies of the Java and Python programming language, suggesting a potentially high impact of automated error messages and hints based on this technique. The automated nature of token signature analysis also lends itself to future development with sophisticated data mining technologies in the areas of machine learning, search, artificial intelligence, databases and statistics.