Clustering and Visualizing Solution Variation ~~In~~ in Massive Programming Classes

by

Elena L. Glassman

Submitted to the Department of Electrical Engineering
and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

August 2016

Author . . . . . .
Department of Electrical Engineering
and Computer Science
August 7, 2016

Certified by . . . . . .
Robert C. Miller
Professor of Electrical Engineering
and Computer Science
Thesis Supervisor

Accepted by . . . . . .
_____
Chairman, Department Committee on Graduate Students

Clustering and Visualizing Solution Variation ~~In~~ in Massive Programming Classes

by

Elena L. Glassman

Submitted to the Department of Electrical Engineering
and Computer Science
on August 7, 2016, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

~~In a massive open online course (MOOC)~~In large programming classes, a single ~~programming exercise~~ problem may yield thousands of student solutions~~that vary in many ways, some superficial and some fundamental~~. Solutions can vary in correctness, approach, and readability. Understanding large-scale variation in ~~programs~~ solutions is a hard but important problem. For teachers, this variation ~~can~~ could be a source of ~~pedagogically valuable examples and expose corner casesnot yet covered by autograding. For students,~~ innovative new student solutions and instructive examples. Understanding solution variation could help teachers write better feedback, test cases, and evaluation rubrics. Theories of learning, e.g., analogical learning and variation theory, suggest that students would benefit from understanding the variation in ~~a large class means that~~ their peers' solutions as well. Even when there are many solutions to a problem, when a student is struggling in a large class, other students may have struggled along a similar solution path, hit the same bugs, and ~~can offer~~ have hints based on that earned expertise.

This thesis describes ~~three systems that explore the value of solution variation in large-scale programming and simulated digital circuit classes. All three systems~~ systems that exploit large-scale solution variation and have been evaluated using data or live deployments in on-campus or edX courses with thousands of students. ~~(1)~~ OverCode visualizes thousands of programming solutions using static and dynamic analysis to cluster similar solutions. ~~It~~ Compared to the status quo, OverCode lets teachers quickly develop a high-level view of student understanding and misconceptions and provide feedback that is relevant to ~~many~~ more student solutions. ~~(2) Foobaz clusters variables in student programs by their names and behavior so that teachers can give feedback on~~ Foobaz helps teachers give feedback at scale on a critical aspect of readability, i.e., variable naming. ~~Rather than requiring the teacher to comment on thousands of students individually, Foobaz~~ Foobaz displays the distribution of student-chosen names for each common variable in student solutions and,

with a few teacher annotations, it generates personalized quizzes that help students ~~evaluate their own names by comparing them with~~ learn from the good and bad ~~names from other students. (3) ClassOverflow collects and organizes solution hints indexed by the autograder test that failed or a performance characteristic like size or speed. It helps students reflect on their debugging or optimization process, generates hints that can help other students with the same problem, and could potentially bootstrap an intelligent tutor tailored to the problem~~naming choices of their peers. Finally, this system describes two complementary learnersourcing workflows that help students write hints for each other while reflecting on their own bugs and comparing their own solutions with other student solutions. These systems demonstrate how clustering and visualizing ~~student solutions helps teachers and students provide types of one-on-one design feedback at scale that was previously only possible in a small classroom or one-on-one tutoring. The design choices around which these scaled-up forms of feedback rest can be curated by teachers directly from~~ solution variation can help teachers directly respond to trends and outliers within ~~the students' own solutions. The feedback generated by both teachers and students can be re-used by future students who attempt the same programming or hardware design problem.~~

student solutions, as well as help students help each other.

Thesis Supervisor: Robert C. Miller
Title: Professor of Electrical Engineering
and Computer Science

Preface

I am not a professional software developer, but learning how to write programs in middle school was one of the most empowering skills my father could ever have taught me. I did not need access to chemicals or heavy ~~machinery–just~~ machinery. I only needed a computer and ~~, occasionally ,~~ occasionally an internet connection. I acquired datasets and wrote programs to extract interesting patterns in them. It ~~was–and still is–a~~ was and still is a creative outlet.

More recently, the value of learning how to code, or at least how to ~~"think computationally"~~ think computationally, has gained national attention. Last September, New York City Mayor Bill de Blasio announced that all public schools in NYC will be required to offer computer science to all students by 2025. In January of this year, the White House released its Computer Science ~~For~~ for All initiative, "offering every student the ~~hands-on~~ hands–on computer science and math classes that make them ~~job-ready~~ job–ready on day one" (President Obama, 2016 State of the Union Address).

The economic value and employment prospects associated with knowing how to program, the subsiding stigma of being ~~"a computer geek~~a computer "geek" or "nerd," and the production and popularity of movies about ~~(now rich) programmers like The Social Network, has driven~~ programmers may be responsible for driving up enrollment in computer science classes to unprecedented levels. Hundreds or thousands of students enroll in programming ~~classes~~ courses at schools like MIT, Stanford, Berkeley and the University of Washington.

One-on-one tutoring is considered a gold standard in education, and programming education is likely no exception. However, most students are not going to receive that kind of personalized instruction. We, as a computer science community, may not be able to offer one-on-one tutoring at a massive scale, but can we create systems that enhance the teacher and student experiences in massive classrooms in ways that would never have been possible in one-on-one tutoring? This thesis is one particular approach to answering that question.

Acknowledgments

~~There are countless individuals who have helped me in ways, large and small, complete this particular marathon.~~

~~I thank my father for inviting me to think with him about interesting problems and principles in electrical engineering and computer science (even while we were in the middle of a Saturday morning jog through town when I was growing up), for teaching me how to program, for helping me learn new concepts by explaining them to him, and for providing emotional support throughout my time becoming an engineer.~~

~~I thank my mother for helping me put together science fair boards at the last minute, copy-editing my paper drafts, and listening to my many updates about life and drama at the Institute.~~

~~I thank my brother for demonstrating how to completely switch fields of expertise, become a computer scientist and software developer, and look good doing it.~~

~~I thank my partner, Victor, for making my final year of graduate school so much fun, giving meencouraging pep-talks when I was felt down, and reminding me to "squeeze limes" and do what needs to be done, even when I did not want to. I thank my labmates, Carrie, Juho, Tom, Katrina, ...~~

~~First, I would like to thank my parents for fostering my love of scientific inquiry (literally, a love of asking good questions and problems worth working on), for encouraging me to sit by the fire in a comfy chair and ask 'what if?' and for proofreading all my camera-ready submissions.~~

~~I would also like to thank my brother, for being a model of courage and steady progress on a PhD in a new field. We both switched fields to get to where we are, and making progress side by side (virtually) has been a resevoir of quiet support.~~

~~I am grateful for my thesis advisor Rob Miller's guidance, support, and insight. Bumping~~

into him in the Gates tower stairway was one of the best things that happened to me in graduate school.

My collaborators...

My lab neighbors...

My friends outsid the lab...

I am indebted to many people who both directly and indirectly contributed to this thesis. First, Iwould like to thank those collaborators who directly contributed. Most of all, I'm grateful for the help and friendship of Mark Styczynski. Mark was my collaborator on all matters computational for the past four years — his influence is evident throughout this document. I'm also grateful for my collaboration with Christopher Loose, who performed many of the experiments on antimicrobial peptides described in Chapter **??**. Finally, I would like to thank Isidore Rigoutsos, who straddled the line between collaborator and advisor. Isidore taught me an attention to detail and a penchant for the UNIX command line and vi editor.

Most importantly, I am indebted to Greg Stephanopoulos, my advisor, whose guidance and support was unwavering these past six years. Greg is the perpetual optimist — always positive in the face of my many failures along the way. He also gave me the freedom to pursue projects of my own choosing, which contributed greatly to my academic independence, if not the selection of wise projects.

I am much obliged to my thesis committee members: my advisor Greg, Isidore, Bill Green, and Bob Berwick. My committee was always flexible in scheduling and judicious in their application of both carrots and sticks.

There are numerous people who contributed indirectly to this thesis. First among these is my intelligent, lovely, and vivacious wife Kathryn Miller–Jensen. NextThank you, thank you, thank you to my advisor, Rob Miller, who took a chance on me, and my other co-authors on this thesis work, Rishabh Singh, Jeremy Scott, Philip Guo, Lyla Fischer,

Aaron Lin, and Carrie Cai. The past and present members of our User Interface Design group were instrumental in helping me feel at home in a new area and get up to speed while having fun. I'm also grateful to my past internship mentors at Google and MSR, specifically Dan Russell, Merrie Ringel Morris, and Andrés Monroy-Hernández, who gave me the chance to try out my chops in new environments. I also want to thank my friends outside MIT, especially the greater Boston community of wrestling coaches, who helped me learn important lessons outside the classroom. And, last but not least, my partner Victor, my parents~~Carl and Julie, my sister Heather, and my in-laws, Ron, Joyce, Suzanne, Jeff, and Mindi. Finally, there are innumerable friends who contributed and to whom I am greatly appreciative including Michael Raab, Joel Moxley, Bill Schmitt, Vipin Guptda, and Jatin Misra.~~ , and my brother, who have each supported me in their own way, from hugs, to proofreading papers, to being technical sounding-boards, and finally, to demonstrating, as my brother has, how to switch fields, pick up a whole new set of skills, and look good doing it.