Addressing Novice Coding Patterns: Evaluating and Improving a Tool for Code Analysis and Feedback

Jacqulyn MacHardy Anderson
University of Utah

UUCS-20-002

School of Computing
University of Utah
Salt Lake City, UT 84112 USA

16 April 2020

Abstract

Successful software needs maintenance over long periods of time. The original code needs to solve the problem and be maintainable. Computer science instructors at universities try to teach students in introductory classes how to write code that meets these goals. Results are inconsistent though. Readability is an important aspect of maintainability, and writing readable code requires choosing and structuring statements and expressions in a way that is idiomatic. Instructors of advanced CS courses perceive that their students frequently don't possess or don't engage this skill of writing with good structure. Building and evaluating this skill is complicated. Could students benefit from a tool that can identify and flag poor code structure and provide hints or suggestions on how to improve it while they program? Can a batch-processing version of the same tool help instructors quickly see and address the gaps in their students' understanding? These questions were investigated through think-alouds with ten students as they coded in Eclipse using PMD, a code structure analysis tool and interviews with four professors who teach introductory computer science courses.

In the think-alouds, students did produce novice code structures, and the tool performed as intended in flagging these structures and providing feedback. All students were able to correctly interpret the tool's feedback to successfully revise at least one

of their initial implementations to use expert code structure. However, there is room for improvement in the tool's feedback.

The instructor interview results showed that although instructors believe that these patterns are important for students to learn and use, assessing and giving feedback on this aspect of student code by hand does not scale well to large class sizes, so their current grading and feedback processes do not target these kinds of code structures. Instructors believe these tools have the potential to address some of these gaps and challenges. Although no instructors want to interface with the output of the batch-processing tool the way it is currently presented, all of them were interested in seeing it extended and provided input on how the tools could be extended to better meet their needs.