Empir Software Eng (2014) 19:1259–1260
DOI 10.1007/s10664-014-9329-5

EDITORIAL

Special issue on program comprehension

Michael W. Godfrey · Arie van Deursen

Published online: 26 July 2014

This special issue of EMSE comprises four selected papers on program comprehension, an unusually wide-ranging research area within software engineering. It encompasses the human activities of comprehending software, the technologies for supporting the program comprehension, and their evaluation. It is notable for combining aspects of cognitive psychology, tool design, and empirical study of users, tools, and software artifacts.

In "High-MCC Functions in the Linux Kernel", the authors challenge the assumption that functions with high values of McCabe Cyclomatic Complexity (MCC) are likely to be problematic to understand and maintain. They explore the nuances of what a high MCC value might mean, and examine functions from the Linux kernel with high MCC values. They find that often the structure of these functions is found to be easy to understand by users, leading to the conclusion that syntactic measures alone may not be a good metric for understandability.

In "Measuring and Modeling Programming Experience", the authors describe a controlled experiment in assessing how the experience of software developers can best be measured. This is a particularly important problem in program comprehension research, as developer experience is often a key factor that must be explicitly taken into account when studying how programmers perform problem-solving tasks.

In "An Empirical Study on the Impact of Static Typing on Software Maintainability", the authors tackle one of the classic programming language debates: How does static typing affect human software development capabilities? The authors approach this question from an empirical perspective. In a carefully designed experiment, the authors assess the impact of static typing on understanding undocumented code, fixing type errors, and fixing semantic errors. The results show rigorous empirical evidence that static types are indeed beneficial to these activities, except when fixing semantic errors.

In "Labeling Source Code with Information Retrieval Methods: An Empirical Study", the authors address augmenting source code with explanatory labels. Automated techniques for this exist, which are based on, e.g., Vector Space Models, Latent Semantic Indexing (LSI), and latent Dirichlet allocation (LDA). The authors describe an experiment in which such

M. W. Godfrey (✉)
David R. Cheriton School of Computer Science, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada
e-mail: migod@uwaterloo.ca

A. van Deursen
Delft University of Technology, Mekelweg 4, 2628 CD Delft, The Netherlands

automatically generated labels are compared to a gold standard of labels created by humans. The results indicate that the simpler automated techniques (directly using words extracted from the code) better reflect human-based labeling.

These four papers well reflect the highly empirical nature of today's program comprehension research. The studies cover human aspects, source code analysis, and automated tool support for program comprehension. As such, these papers illustrate both the breadth and depth of the research that goes on in this field.

Michael W. Godfrey is an associate professor in the David R. Cheriton School of Computer Science at the University of Waterloo. His research interests span many areas of empirical software engineering including software evolution, mining software repositories, reverse engineering, program comprehension, and software clone detection and analysis.