Editorial

Software Engineering for CSE

Jeffrey C. Carver,¹ Neil Chue Hong,² and Selim Ciraci³

¹University of Alabama, Tuscaloosa, AL, USA

²Software Sustainability Institute, University of Edinburgh, Edinburgh, UK

³Microsoft, Redmond, WA, USA

Correspondence should be addressed to Jeffrey C. Carver; carver@cs.ua.edu

Received 11 March 2015; Accepted 11 March 2015

Copyright © 2015 Jeffrey C. Carver et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

1. Workshop Overview

This special issue contains extensions of the best papers from the First International Workshops on Software Engineering for High Performance Computing in Computational Science & Engineering (SE-HPCCSE 2013), which was held during the SC'13 conference. For full details about the workshop (and others in this series), please visit the workshop website <http://SE4Science.org/workshops>, where the interested reader can find an overview of the workshop, the schedule, and links to the published proceedings. The goal of the workshop was to bring together researchers from different domains (i.e., computational science, software engineering, and high-performance computing) to present their work and discuss important issues related to the intersection of these fields. Because the format of the workshop allows for short paper presentations along with ample time for small group discussion, this workshop provides a unique venue where researchers from different backgrounds can meet and interact in a more informal setting. This editorial first briefly describes the interesting results of the group discussions. Then, it provides a brief overview of the three papers included in the special issue.

2. Summary of Workshop Discussion

During the workshop, the attendees focused their discussion on a number of topics. Here we summarize the discussion on two of the most interesting. The first topic is the use of *design patterns* in high-performance scientific software. In general, attendees thought that using appropriate design patterns in

the right situation was beneficial. There was some discussion about where scientific developers could find a good catalog of existing patterns and where new patterns could be published. One concern was whether design patterns are applicable in situations where there is a large base of existing code. The belief was that it is not worth the effort to tear down and rewrite such code, just to take advantage of a design pattern.

The second topic was the use of *unit testing*. Recent programming languages and development environments provide unit testing frameworks, which makes it easier for developers to adopt this approach. The attendees thought that unit testing is necessary to ensure the correctness of high-performance scientific software systems. However, while support for unit testing is present in many languages, the attendees agreed that the languages (such as Fortran) and frameworks typically used for developing high-performance scientific software do not have adequate support for unit testing. The general consensus among the attendees was that developers need to be introduced to widely used software testing tools, such as memory leak detectors and stress testing frameworks.

At the end of the workshop, we conducted a larger group discussion on the path forward in this field. During this discussion, the attendees highlighted a number of barriers that hamper further advancement. Some of these barriers include the following:

- (i) How to deal with reproducibility: is it really necessary to have bitwise reproducibility?
- (ii) Still a lot of tooling that is not available for the languages commonly in use.

3. Summary of Included Papers

This special issue includes three papers. The paper “Extracting UML Class Diagrams from Object-Oriented Fortran: ForUML” describes a tool that automatically extracts UML class diagrams from Fortran source code. The paper “High-Performance Design Patterns for Modern Fortran” describes and evaluates a coarray MPI implementation of some patterns that support asynchronous evaluation of expressions comprised of parallel operations on distributed data. The paper “Test-driven coarray parallelization of a legacy Fortran application” describes 17 code modernization steps used to refactor and parallelize a legacy Fortran program and evaluates the performance of the resulting code.

Acknowledgments

We would like to thank the authors and the reviewers for making the workshop and this special issue possible. Jeffrey C. Carver would like to acknowledge partial support from NSF-1243887. Neil Chue Hong was supported by the UK Engineering and Physical Sciences Research Council (EPSRC) Grant EP/H043160/1 for the UK Software Sustainability Institute.

*Jeffrey C. Carver
Neil Chue Hong
Selim Ciraci*