Software Qual J (2009) 17:1–2
DOI 10.1007/s11219-008-9061-8

In this issue

James Bieman

Published online: 10 September 2008

I wish you a late Happy New Year, and hope that 2009 is a great year for you.

This is my last "In This Issue" for the *Software Quality Journal*. I have served as Editor-in-Chief since the fall of 2001—the first issue that I put together was SQJ Vol 10, No. 1, which was the first issue of 2002. Since then, the journal ownership has changed from Kluwer to Springer, and we have changed from a hard-copy and snail mail reviewing process to a fully electronic web based system. During past seven years the rate of new submissions to the *SQJ* has more than doubled.

Now it is time for new leadership to take over. Thus, I am pleased to announce that Rachel Harrison will replace me as the Software Quality Journal's new Editor-in-Chief starting with the next issue, Vol. 17, No. 2. This will allow me to focus on new research projects involving the challenges of developing highly-adaptable medical and scientific software.

Thanks to the readers, the Editorial Board, the reviewers, and especially to the professional staff at Springer. You have all made the journal a success. In particular, I want to thank Susan Lagerstrom-Fife, the journal's Publishing Editor. I also want to thank Sharon Palleschi and Jill Strathdee for their help in managing the production of the journal. I thank Melissa Andersen for her work in managing the review process for the past several years. She made the transition to electronic web-based reviewing possible. I also thank Cheryl Knight for her work in supporting the review process when I first took over as Editor-in-Chief. Finally, I thank all of those in the Kluwer and Springer production and editorial offices who have helped in editing and producing the *SQJ* over the last seven years.

Please join me in welcoming Rachel Harrison as the new Editor-in-Chief of the *SQJ*. See my introduction and welcome to Rachel for a description of her impressive credentials.

The five research papers in this issue deal with the following topics: measuring coupling in component-based software, constructing quality models with varied data, evaluating legacy systems, quality models for service-oriented systems, and the use of Bayesian analysis to determine when to release software.

J. Bieman (✉)
Fort Collins, CO, USA
e-mail: bieman@cs.colostate.edu

Since the early days of software engineering, researchers and practitioners have recognized that *coupling* is a key design attribute. We now have empirical evidence that design entities that are more tightly coupled to other entities are more likely to be fault prone. Additional knowledge about the variety of mechanisms that can couple design elements should help us to develop better software designs. In "Multiple-Parameter Coupling Metrics for Layered Component-Based Software", Liguo Yu, Kai Chen, and Srini Ramaswamy introduce coupling measures for component-based software that addresses the distance between coupled elements. These measures were applied to the Apache system. Results suggest that more tightly coupled components tend to be more change-prone.

The software engineering community continues to seek better ways to predict software attributes from early data. Improved methods to generate prediction models will certainly help to improve software quality and improve the quality of the software process. In Software Quality Analysis by Combining Multiple Projects and Learners, Taghi M. Khoshgoftaar, Pierre Rebours, and Naeem Seliya demonstrate a new method for merging model training data generated by multiple learners on multiple data sets. The method was evaluated on data from seven NASA software projects. Results show that the greatest improvements in model performance occur when a data is used from a single-learner on multiple data sets.

The reuse of existing software in new systems can both potentially lower development costs and improve quality. However, these benefits can be outweighed by the effort required to use these legacy systems in a new environment. In Evaluating Legacy Assets in the Context of Migration to SOA, Vinay Kumar Reddy, Alpana Dubey, Sala Lakshmanan, Srihari Sukumaran, and Rajendra Sisodia examine the relationship between several internal design attributes and the effort required for reuse. Like others, they argue that no single measure can fully capture the notion of reusability.

A comprehensive quality model that includes attributes of interest to a variety of stakeholders can help to fully evaluate the quality of a system. A number of quality models exist; each model has a particular focus. In A Comprehensive Quality Model for Service-Oriented Systems, Ivan J. Jureta, Caroline Herssens, and Stephane Faulkner integrate existing models into a single model using UML—QVDP. To fully describe the model the authors extend the UML meta-model. The use of QVDP is demonstrated with examples involving the European Space Agency.

A key concern in software development is determining when to release a software system. If you release a system too soon, the software is likely to fail repeatedly during operational use. If you release the system too late, competitive deadlines can be missed with great economic cost. In Bayesian Updating of Optimal Release Time for Software Systems, Kuei-Chen Chiu, Jyh-Wen Ho, and Yeu-Shiang Huang analyze software reliability growth using a Bayesian approach. This method supports analyses of the learning effect of testing and debugging on reliability. Such support is needed to make reliability models more realistic, since testing and debugging clearly affect the reliability of a software product.

I hope that you enjoy this issue of the Software Quality Journal. Thanks again for allowing me to serve as your editor over the past 7 years.