Postprint of article in *International Journal of Software Engineering and Knowledge Engineering* 16 (5): 653–654 (2006)

GUEST EDITORS' INTRODUCTION

KAI-YUAN CAI

Department of Automatic Control
Beijing University of Aeronautics and Astronautics
Beijing 100083, China
kycai@buaa.edu.cn

ATSUSHI OHNISHI

Department of Computer Science
Ritsumeikan University
Shiga 525-8577, Japan
ohnishi@cs.ritsumei.ac.jp

T. H. TSE

Department of Computer Science
The University of Hong Kong
Pokfulam, Hong Kong
thtse@cs.hku.hk

The software development process involves numerous technical and management activities. The resulting software products are usually huge in size. Software executions are highly dynamic with unforeseen invocations and communications among components as well as intensive interactions with the operating environment. This is further intensified by concurrency and non-determinism issues in distributed computing. All these aspects contribute to intrinsic complexities in software systems. Limitations in modeling, analysis, and control methods further make software uncertain and defect-prone. It is a great challenge to alleviate the unwarranted complexities and defects of software systems with a view to enhancing their quality.

This special issue on Quality Software aims at addressing this challenge. It is associated with the Fifth International Conference on Quality Software (QSIC 2005) held in Melbourne, Australia on September 19–20, 2005. Kai-Yuan Cai and Atsushi Ohnishi served as the Program Co-Chairs and T. H. Tse was the Steering Committee Chair. 37 papers were selected from 130 submissions to the conference. The authors of six selected papers were invited to submit extended versions of their work for consideration by the special issue. Every paper was peer-reviewed by at least two referees. Five papers were finally selected for inclusion in the special issue after one to two rounds of revisions according to the reviews.

The first paper, "Cost Simulation and Performance Optimization of Web-Based Applications on Mobile Channels" by Book, Gruhn, Hülder, Köhler, and

Kriegel presents a PETTICOAT method to simulate the effect of adding a mobile presentation channel to an existing web-based application. Typical interaction sequences and volume statistics are generated via a dialog flow model and web server log files. The volume and time costs can then be evaluated by simulating the interaction sequences on various mobile channels.

The second paper, "Integration Testing of Context-Sensitive Middleware-Based Applications: A Metamorphic Approach" by Chan, Chen, Lu, Tse, and Yau, tackles the problems in testing context-sensitive software. The verification of test outcomes is difficult because of the volatile situations resulting from middleware activations. The authors recommend the selection of test cases between specific "checkpoints", where the middleware temporarily ceases to activate the functions under test. The expected relations among test cases between the same checkpoints enable testers to verify the test results.

The third paper, "Action Machines: A Framework for Encoding and Composing Partial Behaviors", by Grieskamp, Kicillof, and Tillmann, introduces action machines that model software behaviors in the form of labeled transition systems. The labels represent observable activities and the states represent data models. Compositions of action machines can be used to model more complex behaviors. The motivation and experience of applying action machines to model-based conformance testing in a production project at Microsoft is also discussed.

The fourth paper, "Model-Based Testing of Concurrent Programs with Predicate Sequencing Constraints" by Wu and Lin, proposes a logic for predicate sequencing constraints, which specifies the functions of concurrent programs that should be tested. The logic not only expresses the sequencing relationships among input and output events, but also the data dependencies among the parameters of such event parameters. Based on the logic, a method has been developed to generate symbolic test cases automatically. A case study has been conducted to verify the advantages of the approach.

The fifth paper, "Slicing Execution for Model Checking C Programs" by Yi, Wang, and Yang, proposes a method of slicing execution for model checking the temporal safety properties of C programs. It significantly reduces state space and execution paths by variable abstraction and partial strongest post-conditions. Experimental studies have been conducted on the initial handshake process of SSL, the application layer protocol for transmitting secure documents via the Internet. The results show that the method is both efficient and practical.

We are grateful to Prof. S.-K. Chang, Editor-in-Chief of the *International Journal of Software Engineering and Knowledge Engineering*, for giving us the excellent opportunity to guest-edit this special issue. We are indebted to the reviewers of the papers who have generously devoted their time and effort to assure the quality of the issue. Last, but not least, we appreciate the outstanding contributions of the authors.