Contents lists available at ScienceDirect

Science of Computer Programming

www.elsevier.com/locate/scico

Preface

Formal Aspects of Component Software (FACS 2010 selected and extended papers)

This issue includes extended versions of selected best papers from the 7th International Workshop on Formal Aspects of Component Software (FACS 2010) held in Guimarães, Portugal on October 14–16, 2010.

The component-based software development approach has emerged as a promising paradigm to cope with an ever increasing complexity of present-day software solutions by bringing sound production and engineering principles into software engineering. However, many conceptual and technological issues remain that challenge component-based software development theory and practice. To address these issues, FACS seeks to provide a forum for researchers and practitioners in the areas of component software and formal methods to foster a better understanding of the component-based paradigm and its applications as well as how formal methods can or should be used to make component-based software development succeed.

Formal methods consist of mathematically-based techniques for the specification, development, and verification of software and hardware systems. They have shown great utility for providing the formal foundations of component-based software and working out challenging issues such as mathematical models for components, composition and adaptation, or rigorous approaches to verification, deployment, testing, and certification. FACS aims at developing a community-based understanding to relevant and emerging research problems related to the application of formal methods for the specification, verification, and composition of components and services.

In response to the call for papers, FACS 2010 attracted 37 submissions from 19 countries, of which 13 full papers and 4 Doctoral Track submissions were accepted by the program committee for inclusion in the proceedings FACS 2010 that was published as volume 6921 in Springer's *Lecture Notes in Computer Science* series. Out of these papers, we invited the authors of the accepted full papers to submit to this special issue. After an extensive and rigorous reviewing process, in which each paper was reviewed by at least three reviewers, we decided to include five of them in this special issue.

Process calculi provide an elegant framework for algebraic characterization of software components and component interaction. However, modeling in process calculi is rather low level, an aspect that can impede effective analysis, in particular, of the service-oriented facets of software systems. To improve on the expressiveness of a process calculus-based notation, Liang Zhao, Roberto Bruni, and Zhiming Liu propose a graph representation of structured service systems in "*A Sound and Complete Theory of Graph Transformations for Service Programming with Sessions and Pipelines*." Graphs in this context encode CaSPiS processes and their behavior. CaSPiS is a service-oriented process calculus in which pipelines serve as a means to orchestrate the data flow arising from session activities. The transformation of CaSPiS processes to an algebra of hierarchical graphs, which is sound and complete with respect to the reduction semantics of CaSPiS, allows for a more effective analysis of service-oriented features in software systems due to the intuitive nature of the graph-based notation, its mathematical elegance, and the vast amount of readily available graph-based analysis and transformation techniques.

Algebraic characterization and verification of service-oriented systems is also the focal point in the paper "*Model Checking Adaptive Service Compositions*" by Michele Bugliesi, Andrea Marin, and Sabina Rossi. Multilevel security and transactional correctness represent critical properties in Service Oriented Architectures (SOAs). To guarantee information flow security and transactional correctness, Michele Bugliesi, Andrea Marin, and Sabina Rossi present a uniform model-checking framework, based on modal μ-calculus specifications, for the verification and analysis of adaptive service components and their compositions. The associated model-checking algorithm not only guarantees correctness, but also provides a means to automatically synthesize desired security constraints for SOAs, in particular, and distributed systems, in general.

Quality of Service (QoS) contracts are the topic of the paper "*QoS Contract Preservation through Dynamic Reconfiguration: A Formal Semantics Approach*" by Gabriel Tamura, Rubby Casallas, Anthony Cleve, and Laurence Duchien. Contracts are a natural means to capture expected service level agreements. To guarantee reliability and robustness for these service level agreements, Gabriel Tamura, Rubby Casallas, Anthony Cleve, and Laurence Duchien develop a two-layered denotational semantics framework for QoS contracts. In the governing layer, a QoS contract is represented by an extended finite state machine in which QoS level conditions are states, and contract reconfiguration rules are transitions. In the operating layer,

http://dx.doi.org/10.1016/j.scico.2014.03.010

QoS contracts are mapped to typed-attributed graphs that allow for context-aware reconfiguration. A particular benefit of this approach is that it permits quality attributes to be expressed as design patterns, which are encoded in dedicated reconfiguration rules to instantiate them at runtime in order to achieve required QoS levels.

The adaptation of a component-based system can be formulated as a coordination problem. In the paper "*Dynamic Adaptation with Distributed Control in Paradigm*," Suzana Andova, Luuk P.J. Groenewegen, and Erik de Vink use the coordination modeling language Paradigm through McPal to demonstrate how new and modified behavior can be incorporated in components and their interaction. The approach is illustrated by applying it to the well-known dining philosophers problem. In this scenario, system adaptation captures the process of migrating a deadlock-prone solution to a deadlock-free and starvation-free one. This paper details the Paradigm modeling process of migration and provides a proof of its correctness.

Code annotation offers an effective means to runtime verification of component-based systems. In the paper "*Monitoring Method Call Sequences Using Annotations*," Behrooz Nobakht, Frank S. de Boer, Marcello M. Bonsangue, Stijn de Gouw, and Mohammad Mahdi Jaghoori present JMSeq, a framework for monitoring sequences of method calls in Java programs. JMSeq provides a non-invasive approach to intercept Java method calls in order to test for potential sequence and protocol violations. Verification goals are expressed as annotations, metadata to drive JMSeq's verification engine. In contrast to classical instrumentation techniques, the annotation-based approach of JMSeq exhibits a lower runtime overhead, making it also applicable in high load scenarios. Some problems may only manifest themselves, if load exceeds a certain threshold.

We would like to express our gratitude to all the people who have made this special issue possible. First and foremost, our most sincere appreciation is to the authors for submitting their papers and incorporating all the corrections and improvements as advised by a thorough reviewing process. We are indebted to the reviewers for kindly contributing their time and effort to ensure the highest quality of each paper. Last but not least, we would like to thank Jan A. Bergstra, Bas van Vlijmen, and the editorial staff at Elsevier for agreeing to publish this special issue as a volume in *Science of Computer Programming*, and their assistance in bringing this special issue to publication.

Luís Soares Barbosa
University of Minho, Portugal

Markus Lumpe
Swinburne University of Technology, Australia

11 March 2014
Available online 8 April 2014