
Ibrahim K. El Far [Chair]  
Microsoft Corporation  
One Microsoft Way  
Redmond, Washington 98052, USA  
++ 1 425 706 6615  
ielfar@acm.org

Chia-Chu Chiang [co-Chair]  
University of Arkansas at Little Rock  
2801 S. University Ave.  
Little Rock, Arkansas 72204, USA  
++ 1 501 569 8142  
cxchiang@ualr.edu

Stefan Gruner [co-Chair]  
University of Southampton  
Southampton, Hampshire SO17 1BJ  
Great Britain  
++ 44 23 80 59 31 23  
sg@ecs.soton.ac.uk

Markus Montigel [co-Chair]  
Systransis AG  
Technopark Luzern, D4 Platz 4, CH-6039 Root Langenbold, Switzerland  
++ 41 41 455 21 31  
montigel@systransis.ch

Herbert H. Thompson [co-Chair]  
Security Innovation  
1900 S. Harbor City Blvd. Suite 209  
Melbourne, Florida 32901, USA  
++ 1 321 308 0557  
hthompson@sisecure.com

The Special Track on Software Engineering
Software engineering by nature is a broad field that brings together people from various disciplines in computing and beyond. It has a particularly wide appeal as it draws academic and commercial researchers as well as professional engineers. The goal of this special track has always been to solicit high quality submissions that reflect this diversity with an emphasis on novel, practical ideas and useful, practicable results. Authors from around the world responded to our call for papers with several strong submissions, and we are pleased with the papers that have been selected for this year’s track. The authors represented universities, research institutes, and companies from over 10 countries, including the United States, Cyprus, France, China, Italy, Spain, India, Germany, Australia, Greece, Israel, the United Kingdom, Austria, Finland, and Norway. In the end, 40 submissions were made but, after 120 reviews, only 10 papers were ultimately accepted. We briefly introduce these papers below.

The Papers
The accepted papers are only becoming of software engineering in diversity and the symposium in usefulness and practicality. The papers have something to offer for people who are interested in the subjects of requirements engineering, formal methods, software modeling, program debugging, software metrics, and aspect-oriented programming, among others. They also offer value for those who are concerned with everyday technologies such as the Unified Modeling Language (UML), the Object Constraint Language, the Java Modeling Language, and Abstract State Machines, as well as new experimental technologies with a promising future.

Missing Requirements and Relationship Discovery through Proxy Viewpoints Model, by Seok Won Lee and David C. Rine. The authors, from the University of North Carolina at Charlotte, United States and George Mason University, United States, tackle a very hard problem in software requirements engineering, that of identifying missing requirements in specifications expressed in natural language. They introduce a method they dub “Proxy Viewpoints Model-Based Requirements Discovery” that employs a number of models to view requirements from different perspectives, to add structure to the specification, and to facilitate discovery of requirements.

A Framework to Simulate UML Models: Moving from a Semi-Formal to a Formal Environment, by Alessandra Cavarra, Elvinia Riccobene, and Patrizia Scandurra. The authors, from Oxford University, UK and University of Catania, Italy, propose a framework and toolkit that assist in simulating UML models. Through validation by symbolic execution of UML models, software engineers can gain confidence in their models. In combination with existing tools that facilitate drawing and documenting UML models, this framework adds a lot of value to an engineer’s work without too much imposition from formal methods.

A Portable Virtual Machine for Program Debugging and Directing, by Camil Demetrescu and Irene Finocchi. The authors, from the University of Rome “La Sapienza” and “Tor Vergata”, Italy, present a very useful cross-platform virtual machine that makes easy implementing of various systems that monitor the runtime environment and react to its emitted events, such as program debuggers and software visualization tools.

Test Derivation for Distributed Component-based Systems, by Priyanka Kerhalkar, S. Ramesh, and Ashok Sreenivas. The authors from the TATA Research and Development and Design Center, India and the Indian Institute of Technology at Bombay, India, propose a new formalism to model distributed systems they call Communicating State Machines. They also propose test coverage criteria: communication coverage and constraint coverage, and the corresponding test generation techniques that guarantee the complete testing of each of the criterion.
Translating the Object Constraint Language into the Java Modeling Language, by Ali Hamie. The author, from the University of Brighton, UK, introduces a technique to translate the Object Constraint Language (OCL), often used to constrain UML design, to the Java Modeling Language, an interface specification language used to describe Java classes and interfaces. This proves to be a useful tool toward carrying out logical reasoning about Java classes and interfaces with a UML/OCL design.

A Relational Approach to Software Metrics, by Marco Scotto, Alberto Silliti, Giancarlo Succi, and Tullio Vernazza. The authors, from the University of Genova, Italy and the Free University of Bozen-Bolzano, Italy, address a frequently encountered problem by software metrics researchers and practitioners in large organizations. The problem is twofold: metrics are far from standardized and metrics computation tools undergo frequent changes to accommodate amendments to metric definitions. They propose a tool that employs a relational database to represent source code structure, thus isolating implementation specifics from metric computation, which is now a matter of performing a SQL query.

A New Approach to the BDI Agent-based Modeling, by Chang-Hyun Jo, Guobin Chen, and James Choi. The authors, from the University of California at Fullerton, United States, propose a novel software development process based on intelligent agents. In particular, their process employs Belief-Desire-Intention model, the elements of which are found in use cases, sequence diagrams, activity diagrams, and dataflow diagrams.

A Control Theory Based Framework for Dynamic Adaptable Systems, by Joao W. Cangussu, Kendra Cooper, and Changcheng Li. The authors, from the University of Texas at Dallas, United States, propose a framework they call the State Model Adaptive Run Time to better design dynamically adaptive systems. The framework proves useful in many ways. An example shows the framework is well suited for predicting and avoiding memory constraint violations.

Aspect Oriented Programming for a Component-based Real Life Application: A Case Study, by Odysseas Papapetrou and George A. Papadopoulos. The authors, from the University of Cyprus, Cyprus, present the results of a case study in which they evaluate the paradigm of aspect oriented programming in the context of developing a web crawler. The study compares development processes with and without aspect oriented programming, concluding with findings that favor this paradigm.

Modeling Socio-technical Specifics using Architectural Concepts, by Michael Cebulla. The author, from the Technische Universität Berlin, Germany, develops a visual notation for the modeling of complex socio-technical systems as part of an approach to analyze and models such systems. Socio-technical systems involve both human operators as well as technological components, and the argument is made that such complex systems present special design challenges and a great need for model-based reasoning.

Concluding Remarks
We believe the papers of this track to be valuable additions to the field. Considering the heavy competition for this track along with the diversity and the quality of all the submissions, we can only look forward to many years of excellence in applied computing research in software engineering.