EDITORIAL MESSAGE
Special Track on Object-Oriented Languages and Systems

***Davide Ancona**, University of Genova, Italy*

Introduction

Object-oriented programming (OOP) has become the mainstream programming paradigm for developing complex software systems in most application domains.

However, existing OO languages and platforms need to evolve to meet the continuous demand for new abstractions, features, and tools able to reduce the time, effort, and cost of creating object-oriented software systems, and improving their performance, quality and usability.

To this aim, the Special Track on Object Oriented Programming Languages and Systems (OOPS, http://oops.disi.unige.it) is seeking for research advances bringing benefits in all those typical aspects of software development, such as modeling, prototyping, design, implementation, concurrency and distribution, code generation, analysis, verification, testing, debugging, evaluation, deployment, maintenance, reuse, and software evolution and adaptation.

Statistical Information

In response to the call for papers, 19 papers were submitted to the track.

All papers were referred by a high quality program committee consisting of the following 15 academic and industrial researchers.

- Davide Ancona, University of Genova, Italy
- Alexandre Bergel, University of Chile, Chile
- Walter Binder, University of Lugano, Switzerland
- Carl Friedrich Bolz, King's College London, UK
- Shigeru Chiba, University of Tokyo, Japan
- Dave Clarke, Katholieke Universiteit Leuven/Uppsala University, Belgium/Sweden
- João Costa Seco, Universidade Nova de Lisboa, Portugal
- Wolfgang De Meuter, Vrije Universiteit Brussel, Belgium
- Erik Ernst, Google
- Robert Hirschfeld, Hasso-Plattner-Institut, University of Potsdam, Germany
- Marieke Huisman, University of Twente, The Netherlands
- Jaakko Järvi, Texas A&M University, USA
- Doug Lea, Suny Oswego, USA
- Pavel Parizek, Charles University in Prague, Czech Republic
- Marco Servetto, Victoria University of Wellington, New Zealand

67 review reports were produced with three or four reviews per paper. I am highly grateful to the members of the program committee and to the authors for submitting their contributions. Without their valuable support, it would not have been possible to schedule such a high quality program for the track.

Based on the reviewers' reports, the general ACM SAC guidelines for acceptance and rejection of submissions, and the unavoidable time and space constraints associated with any conference, it was possible to select only five full papers for presentation at the track (acceptance rate 26%).

Summary of Accepted Papers

The Omission Finder for Debugging What-Should-Have-Happened Bugs in Object-Oriented Programs. Kouhei Sakurai and Hidehiko Masuhara

This paper proposes a novel feature called the omission finder for trace-based debuggers. This feature correlates points-to analysis results with an execution history to show operations that could have been but actually were not performed on a specified instance if the program behaved differently.

Adaptive Just-in-time Value Class Optimization: Transparent Data Structure Inlining for Fast Execution. Tobias Pape, Carl Friedrich Bolz and Robert Hirschfeld

This paper presents a technique to detect and compress commonly occurring patterns of value class usage to improve memory usage and performance.

Reifying the Reflectogram -- Towards Explicit Control for Implicit Reflection. Nick Papoulias, Marcus Denker, Stephane Ducasse and Luc Fabresse

This paper presents five dimensions of meta-level control from related literature that try to remedy problems concerned with reflective facilities in OO languages, such as computational overhead, the possibility of meta-recursion and an unclean separation of concerns between base and meta-level.

Composable and Hygienic Typed Syntax Macros. Cyrus Omar, Chenglong Wang and Jonathan Aldrich

This papers focuses on type-specific languages (TSLs) for well-behaved and unambiguously composed syntax extension mechanisms. TSLs are supplemented with typed syntax macros, which are explicitly invoked to give meaning to delimited segments of arbitrary syntax, at both the level of terms and types.

The Safety of Dynamic Mixin Composition. Eden Burton and Emil Sekerinski

This paper presents a methodology for developing mixins that can be composed without negatively affecting their base object's behaviour.