9th Workshop on Aspects, Components, and Patterns for Infrastructure Software (ACP4IS ’10)

co-located with the 9th International Conference on Aspect-Oriented Software Development (AOSD) March 16, 2010, Rennes, France

Workshop Home Page: `http://aosd.net/workshops/acp4is/2010/`

Aspect oriented programming, component models, and design patterns are modern and actively evolving techniques for improving the modularization of complex software. In particular, these techniques hold great promise for the development of “systems infrastructure” software, e. g., application servers, middleware, virtual machines, compilers, operating systems, and other software that provides general services for higher-level applications. The developers of infrastructure software are faced with increasing demands from application programmers needing higher-level support for application development. Meeting these demands requires careful use of software modularization techniques, since infrastructural concerns are notoriously hard to modularize.

Aspects, components, and patterns provide very different means to deal with infrastructure software, but despite their differences, they have much in common. For instance, component models try to free the developer from the need to deal directly with services like security or transactions. These are primary examples of crosscutting concerns, and modularizing such concerns are the main target of aspect-oriented languages. Similarly, design patterns like Visitor and Interceptor facilitate the clean modularization of otherwise tangled concerns.

Building on the ACP4IS meetings at AOSD 2002–2009, ACP4IS ’10 aims to provide a highly interactive forum for researchers and developers to discuss the application of and relationships between aspects, components, and patterns within modern infrastructure software. The goal is to put aspects, components, and patterns into a common reference frame and to build connections between the software engineering and systems communities.

Scope of the Workshop

The importance of “systems infrastructure” software—including application servers, virtual machines, middleware, compilers, and operating systems—is increasing as application programmers demand better and higher-level support for software development. Vendors that provide superior support for application development have a competitive advantage. The software industry as a whole benefits from an increased base level of abstraction, decreasing the need for application programmers to continually “reinvent the wheel”.

These trends, however, mean that the demands on infrastructure software are increasing. More and more features and requirements are being “pushed down” into the infrastructure, and the developers of systems software need better tools and techniques for handling these increased demands. The design and implementation of systems-level software presents unique opportunities and challenges for AOSD techniques. These challenges include the need to address the inherent complexity of infrastructure software, the need for strong assurances of correct and predictable behavior, the need for maximum run-time performance, and the necessity of dealing with the large body of existing systems software components.

This workshop aims to provide a highly interactive forum for researchers and developers to discuss the application of and relationships between aspects, components, and patterns within modern infrastructure software. The goal is to put aspects, components, and patterns into a common reference frame and to build connections between the software engineering and systems communities.

This year's workshop puts special focus on the challenges in system's programming introduced by multi-core platforms. As hardware-supported parallelization becomes mainstream, there is an increasing pressure on systems infrastructure to exploit this new parallelism to its fullest. However, the non-modular nature of parallel execution, and the numerous levels at which parallelism can be achieved (application, systems infrastructure, hardware or even a combination thereof) make it hard to come up with an intuitive, yet efficient parallel architecture. We solicited novel ideas and experience reports on this emerging research area.

Other topics in the scope of the workshop include, but are not restricted to:

- Approaches that combine or relate component-, pattern-, and aspect-based techniques
- Dimensions of infrastructure software quality including comprehensibility, configurability (by implementers), customizability (by users), reliability, evolvability, scalability, and run-time characteristics such as performance and code size
- Merits and downsides of container-, ORB-, and system-based separation of concerns
- Architectural techniques for particular system concerns, e.g., security, static and dynamic optimization, and real-time behaviour
- Design patterns for systems software
- Component, pattern, and aspect "mining" within systems code
- Application- or domain-specific optimization of systems
- Reasoning and optimization across architectural layers
- Quantitative and qualitative evaluations

The workshop is structured to encourage fruitful discussions and build connections between workshop participants. To this end, approximately half of the workshop time will be devoted to short presentations of accepted papers, with the remaining half devoted to semi-structured discussion groups. Participants will be expected to have read the accepted papers prior to the workshop, to help ensure focused discussions.

A novelty at ACP4IS '10 is that we will invite workshop attendees to give "spontaneous" short presentations on their work if they see a relation to topics being presented and discussed at the workshop. These presentations will be limited to about ten minutes, and are intended to provide additional structured input to discussions. Spontaneous presentations will be asked for during the workshop; no paper needs to be submitted, and no publication is associated with them. There will be a session dedicated to them, just prior to discussion.

Organizers

Bram Adams is a post-doctoral fellow in the Software Analysis and Intelligence Lab at Queen's University (Canada), and is also affiliated with the SOFT lab at the Vrije Universiteit Brussel (Belgium). He obtained his PhD at the GH-SEL lab at Ghent University (Belgium). Bram has a wide range of research interests, ranging from software evolution in general, to the co-evolution of source code and the build system, and advanced separation of concerns. In his PhD, Bram developed a powerful aspect language for C (Aspicere), which he applied to large legacy C systems for reverse-engineering their behavior, re-engineering exception handling idioms and refactoring conditional compilation. Bram served in the program committees of WCRE, IWPSE and ACP4IS, and co-organized the first Practices of Linking Aspect Technology and Evolution workshop (associated with AOSD 2009). Bram is a member of the IEEE.

Michael Haupt is a post-doctoral researcher and lecturer in the Software Architecture Group at Hasso-Plattner-Institut in Potsdam. His research interests are in improving the modularity of complex software system architectures as well as in implementing programming languages, in which latter area his main focus is on faithfully regarding programming paradigms' core mechanisms as primary subjects of language implementation effort. Michael holds a doctoral degree from Technische Universität Darmstadt, where he has worked on the Steamloom virtual machine to provide run-time support for AOP languages. Michael has served as PC member for ECOOP 2008, as reviewer for TAOSD and IEEE TSE, and has been supporting reviewer for the AOSD, ECOOP, ICSE, FSE, MODELS, and VEE conference series. He has co-organized the Dynamic Aspects Workshop series in conjunction with the AOSD conferences, and the Virtual Machines and Intermediate Languages workshop series in conjunction with the AOSD and OOPSLA conferences. Michael is a member of the ACM.

Daniel Lohmann is an assistant professor at the Distributed Systems and Operating Systems group at Friedrich-Alexander University Erlangen-Nuremberg. He has been conducting research in the domain of (embedded) operating systems, software product lines, and aspect oriented programming since 2003. Daniel holds a doctoral degree from Friedrich-Alexander University; in his PhD he developed CiAO, the first purely aspect-oriented operating system. His current research activities are focused on applying AOP ideas for the fine-grained configuration of nonfunctional properties in system software and the new challenges of the many-core area. Daniel co-organized the MMB 2006 workshop on Nonfunctional Properties of Embedded Systems and the ACP4IS 2008 and 2009 workshops. Before joining the PhD programme at Friedrich-Alexander University he worked as a software developer, consultant and IT trainer. Daniel is a member of the ACM, GI, and EUROSYS.

Organization

Program Committee

Mehmet Aksit	University of Twente
Shigeru Chiba	Tokyo Institute of Technology
Eric Eide	University of Utah
Michael Engel	Technische Universität Dortmund
Franz Hauck	Ulm University
Julia Lawall	DIKU
Hidehiko Masuhara	University of Tokyo
Hridesh Rajan	Iowa State University
Doug Simon	Sun Microsystems Laboratories
Olaf Spinczyk	University of Dortmund
Eric Wohlstadter	University of British Columbia
Roel Wuyts	IMEC and K.U. Leuven

Steering Committee

Eric Eide	University of Utah
Olaf Spinczyk	University of Dortmund
Yvonne Coady	University of Victoria
David Lorenz	University of Virginia

Sponsor

The publication of this proceedings volume was sponsored by the AOSD-Europe Network of Excellence, `http://www.aosd-europe.net/`.

AOSD-europe
eu network of excellence

Table of Contents

Session 1: Adepts of Code and the Wizards of OS

Session 2: Scanners and Sensors for Components and Code

Session 3: Fantastic Frameworks and Infamous Infrastructures