Aspects, Dependencies and Interactions
Report on the 3rd Workshop ADI at ECOOP 2008

Frans Sanen\textsuperscript{1}, Katharina Mehner\textsuperscript{2}, Ruzanna Chitchyan\textsuperscript{3}, Lodewijk Bergmans\textsuperscript{4}, Johan Fabry\textsuperscript{5}, and Mario Sudholt\textsuperscript{6}

\textsuperscript{1} K.U. Leuven, Leuven, Belgium
\texttt{frans.sanen@cs.kuleuven.be}
\textsuperscript{2} Siemens, Germany
\texttt{Katharina.Mehner@siemens.com}
\textsuperscript{3} Lancaster University, Lancaster, UK
\texttt{rouza@comp.lancs.ac.uk}
\textsuperscript{4} University of Twente, Enschede, The Netherlands
\texttt{L.M.J.Bergmans@ewi.utwente.nl}
\textsuperscript{5} Computer Science Department (DCC), University of Chile
\texttt{jfabry@dcc.uchile.cl}
\textsuperscript{6} Ecole des Mines de Nantes, Nantes, France
\texttt{Mario.Sudholt@emn.fr}

\textbf{Abstract.} The topics on aspects, dependencies and interactions are among the key remaining challenges to be tackled by the Aspect-Oriented Software Development (AOSD) community to enable a wide adoption of AOSD technology. This third workshop, organized and supported by the AOSD-Europe project, aimed to continue the wide discussion on aspects, dependencies and interactions started at ADI 2006 and continued at ADI 2007.

\textbf{Keywords:} Aspects, dependencies, interactions.

\section{Introduction}

Interaction problems between different modules, program parts, units of specifications are a central challenge to many program structuring paradigms, including Aspect-Oriented Software Development, feature-based programming and component-based software engineering. Furthermore, interaction problems are relevant to all phases of the software development life cycle: from requirements through to implementation and often exert a broad influence on these concerns, e.g. by modifying their semantics, structure and / or behavior. Such dependencies often lead to both desirable and unwanted or unexpected behaviors of large-scale applications. The workshop was focused on identifying, understanding, and resolving all kinds of issues related to such dependencies and interactions, by bringing together researchers and practitioners from across the whole spectrum of software development activities and methodologies. The goal of this third workshop was to continue the wide discussion on aspects, dependencies and...
interactions, started at ADI 2006 and continued at ADI 2007, thus investigating the lasting nature of such dependency links across all development activities:

– starting from the early development stages (i.e., requirements, architecture, and design), looking into dependencies between requirements (e.g., positive/negative contributions between aspectual goals) and interactions caused by aspects (e.g., quality attributes) in requirements, architecture, and design;
– analyzing these dependencies and interactions both through modeling and formal analysis;
– considering language design issues which help to handle such dependencies and interactions;
– studying such interactions in applications.

In the rest of this workshop report, we present the main topics that were discussed at the workshop, including a comparative overview of the main topics of the accepted papers, a summary of the keynote speech by James Noble on “We Demand Rigidly Defined Areas of Doubt and Uncertainty!”, a summing-up of the debates hold in the discussion breakout group and a synthesis of the panel chaired by Theo D’Hondt on “Does Model Driven Engineering make Aspects Obsolete?"

2 Accepted Papers

Papers accepted to the workshop covered a broad spectrum of problems related to aspects, dependencies and interactions. We have clustered these papers into three sets, with each set briefly summarized below.

2.1 Architecture

This set of papers focuses mainly on architecture, namely the management of aspect interactions using statically-verified control-flow relations and analyzing layering violations in aspect-oriented software architectures.

In [10], the feasibility of a technique for managing control-flow interactions in layered architectures is demonstrated. The technique proposes to document aspects with policies that specify the expected control-flow relations between different aspects or between aspects and the base application. The policies are expressed as logic formulae that employ a set of predicates that represent relevant control-flow situations. In order to verify these policies, the authors employ and extend existing static analyses to produce interprocedural control-flow graphs of an application with woven aspects. This graph then is traversed in a controlled manner to characterize the realizable paths. The paper starts with the observation that although various aspect-oriented approaches provide support for the management of aspect interactions, most techniques are only applicable when the aspects share a common join point (e.g. [7,8,19]). However, De Fraine et al. motivate that aspect interactions also occur on coarser levels based on two example interactions between the following three aspects in a typical multi-tier architecture: authorization, authentication and caching. E.g. caching should not