Abstract. This report covers the eleventh Workshop on Component-Oriented Programming (WCOP). WCOP has been affiliated with ECOOP since its inception in 1996. The report summarizes the contributions made by authors of accepted position papers as well as those made by all attendees of the workshop sessions.

1 Introduction

WCOP 2006, held in conjunction with ECOOP 2006 in Nantes, France, was the eleventh workshop in the successful series of workshops on component-oriented programming. The previous workshops were held in conjunction with earlier ECOOP conferences in Linz, Austria; Jyväskylä, Finland; Brussels, Belgium; Lisbon, Portugal; Sophia Antipolis, France; Budapest, Hungary; Málaga, Spain, Darmstadt, Germany, and Oslo Norway, and Glasgow, Scotland.

The first workshop, in 1996, focused on the principal idea of software components and worked towards definitions of terms. In particular, a high-level definition of what a software component is was formulated. WCOP97 concentrated on compositional aspects, architecture and gluing, substitutability, interface evolution and non-functional requirements. In 1998, the workshop addressed industrial practice and developed a major focus on the issues of adaptation. The next year, the workshop moved on to address issues of structured software architecture and component frameworks, especially in the context of large systems. WCOP 2000 focused on component composition, validation and refinement and the use of component technology in the software industry. The year after, containers, dynamic reconfiguration, conformance and quality attributes were the main focus. WCOP 2002 had an explicit focus on dynamic reconfiguration of component systems, that is, the overlap between COP and dynamic architectures. 2003, the workshop addressed predictable assembly, model-driven architecture and separation of concerns. The 2004 instance of the workshop focused on various technical issues and also on issues of industrialization of component-orientation. WCOP 2005 revolved around different aspects of trustworthiness with component-oriented programming. Considered were analyzing, asserting,
and verifying functional and non-functional properties of individual components as well as of assembled systems.

A central theme of WCOP 2006 was the composition and deployment of components, including component selection and adaption. A minor focus was the relation between components and aspects, that is between COP and AOP.

WCOP 2006 had been announced as follows:

WCOP 2006 seeks position papers on the important field of component-oriented programming (COP). WCOP 2006 is the eleventh event in a series of highly successful workshops, which took place in conjunction with every ECOOP since 1996.

COP has been described as the natural extension of object-oriented programming to the realm of independently extensible systems. Several important approaches have emerged over the recent years, including component technology standards, such as CORBA/CCM, COM/COM+, J2EE/EJB, and most recently .NET, but also the increasing appreciation of software architecture for component-based systems, and the consequent effects on organizational processes and structures as well as the software development business as a whole.

COP aims at producing software components for a component market and for late composition. Composers are third parties, possibly the end users, who are not able or willing to change components. This requires standards to allow independently created components to interoperate, and specifications that put the composer into the position to decide what can be composed under which conditions. On these grounds, WCOP’96 led to the following definition:

A component is a unit of composition with contractually specified interfaces and explicit context dependencies only. Components can be deployed independently and are subject to composition by third parties.

After WCOP’96 focused on the fundamental terminology of COP, the subsequent workshops expanded into the many related facets of component software.

WCOP 2006 will emphasize reasons for using components beyond re-use. While consider software components a technical means to increase software re-use, other reasons for investing into component technology tend to be overseen. For example, components play an important role in framework and product-lines to enable configurability (even if no component is re-used).

Another role of components beyond re-use is to use components to increase the predictability of the properties of a system. The use of components as contractually specified building blocks of software restricts the degrees of freedom during software development compared to classic line-by-line programming. However, this restriction is beneficial for the predictability of system properties. For an engineering approach to