



## Fourth International Workshop on Composition Languages

In conjunction with  
18th European Conference on Object-Oriented Programming (ECOOP)  
Oslo, Norway  
June 14, 2004

### Abstract:

The goal of this workshop is to bring together researchers and practitioners in the area of component-based software development in order to address problems concerning the design and implementation of composition languages and to develop a common understanding of the corresponding concepts. We would also like to determine the strengths and weaknesses of composition languages and compare it with similar approaches in related fields. In this workshop, we intend to continue the fruitful discussions started at previous workshops on composition languages, which were held in conjunction with ESEC/FSE 2001, ECOOP'02, and ECOOP'03.

Component-based software development, as a sub-area of object-oriented software development, has become a central focus in computing practice. However, due to the lack of appropriate language support, component-based software development remains a labor-intensive and costly undertaking.

The aim of this workshop is to provide a forum to address important issues related to component composition using composition languages. Moreover, we are particularly interested in the refinement of concepts of composition languages and in bridging the gap between theory and practice.

The main focus of the workshop will be on software composition on an architectural level, and not on component-based systems in general. In particular, we would like to emphasize the important issues of (i) the design and implementation of higher-level models and languages for component-based software development, (ii) approaches that combine architectural description, component configuration, and component composition, (iii) paradigms for the specification of reusable software assets, (iv)

expressing applications as compositions of software components, and (v) the derivation of working systems using composition languages and components. Furthermore, we would particularly like to encourage authors to submit position statements focusing on formal aspects of the issues mentioned above and case studies of using composition languages for real-world applications.

Participation at the workshop is upon invitation only and subject to acceptance of a position statement. All submissions will be peer-reviewed by at least two members of the workshop organizing committee. Based on the quality and originality, the authors of a selection of the best position statements will be invited to give a short presentation at the workshop.

The workshop will be organized in several sessions. After an initial presentation session, where all participants can formulate one or more, possibly provocative, working hypotheses, we intend to split the workshop into task forces to foster the discussion a particular subject of common interest. At the end of the workshop the task forces will reunite and we will assemble the results and formulate future work, which we intend to present to the rest of the ECOOP community in the form of a poster at the conference, if possible.

#### Call for Contributions:

The Fourth International Workshop on Composition Languages seeks position statements addressing the design and implementation of higher-level languages suitable for component-based software development.

The component-based software engineering approach mainly consists of two development steps: (i) the specification and implementation of components and (ii) the composition of components into composites or applications. Currently, there is considerable experience in component technology and many resources are spent in defining component models such as CORBA, COM, EJB, and .NET. However, much less effort is spent in investigating appropriate composition languages, which allow application developers to express applications flexibly as compositions of components and, therefore, offer support for component-based software engineering.

What kind of language support is needed to enable software composition? To give an answer to this question is not easy. Currently, despite their limitations in the areas of flexibility, adaptability, and type-safe composition, component-based programming is mainly carried out using mainstream object-oriented languages. However, these languages typically only provide an ad hoc collection of mechanisms for constructing and composing objects, and they are based on ad hoc semantic foundations. Are there any alternatives and if so, what kind of changes do we need to make to current practice?

Currently, component-based programming is mainly carried out using mainstream object-oriented languages. These languages seem to offer already some reasonable support for component-based programming (e.g. encapsulation of state and behavior, inheritance, and late binding). Unfortunately, object-oriented techniques are not powerful enough to provide flexible and type-safe component composition and evolution

mechanisms, respectively. We can identify especially (i) a lack of abstractions for building and adapting class-like components in a framework or domain specific way, (ii) a lack of abstractions for defining cooperation patterns, and (iii) a lack of support for checking the correctness of compositions.

Most available composition environments focus mainly on special application domains and offer at best rudimentary support for the integration of components that were built in a system other than the actual deployment environment. Furthermore, these systems do not enforce a clear separation of computational elements (that is, components) and their relationships, which is needed to address the flexibility and maintainability of component-based systems. The reason for this situation is not only the lack of well-defined (or standardized) component interfaces, but the ad-hoc way the semantics of the underlying language models are defined.

However, much less effort is spent in investigating appropriate techniques that allow application developers to express applications flexibly as compositions of components on an architectural level. Existing composition environments mainly focus on special application domains and offer at best rudimentary support for the integration of components that were built in a system other than the actual deployment environment.

Recently, we have been observing a paradigm shift from component-centric development to model-centric and architecture-centric development. In particular, OMG's definition of the Model Driven Architecture™ (MDA) can be considered as the next step towards solving software integration problems. MDA introduces a clear separation between application logic and application infrastructure by encapsulating infrastructure specific aspects as far as possible in code generators. Hence the specification of the architecture assets and the composition of them are done on a conceptual level and thus reducing the potential of architectural mismatches usually introduced by dependencies to infrastructure.

What are benefits and limits of model-centric approaches? How can we specify component behavior on a conceptual level? How can an existing set of components be integrated with model-centric approaches? How can we bridge the gap between conceptual component models and frameworks, programming languages, and model-driven development?

The goal of this workshop is to bring together both researchers and practitioners. By focusing on important aspects of the design and implementation of composition languages, this workshop aims to address the specific problems of existing composition systems. Suggested topics of interest include, but are not limited to:

#### **Aspect of composition languages**

- Higher-level abstractions for composition languages
- Implementation techniques for composition languages
- Scalability and extensibility of the language abstractions
- Analysis of runtime efficiency of compositional abstractions
- Formal semantics of composition languages
- Type systems for composition languages

- Domain-specific versus general-purpose composition languages
- Case studies of composition language design
- Case studies of system development using composition languages
- Tool support for composition languages
- Taxonomy of composition languages

### **Compositional reasoning**

- Representation strategies for functional and non-functional properties of components
- Prediction of properties of compositions from properties of the involved components
- Reasoning about correctness of compositions
- Specification and validation of architectural guidelines

### **Model-centric and architecture centric approaches**

- Support for the specification of software architectures and architectural assets
- Interoperability support
- Design and implementation strategies for cross-platform development
- Programming paradigms for software composition
- Model-centric and architecture-centric development and composition methods (e.g., MDA)
- Using existing components in model-centric and architecture-centric approaches, model extraction
- Modeling of components, specifically component behavior
- Mapping of architectural models to applications, model transformations
- Benefits of model-centric and architecture-centric approaches
- Case studies and success stories of model-centric and architecture-centric development
- Tool support for model-centric and architecture-centric development

Participation at the workshop is upon invitation only and subject to acceptance of a position statement. All submissions will be peer-reviewed by at least two members of the workshop organizing committee. Based on the quality and originality, the authors of a selection of the best position statements will be invited to give a short presentation at the workshop.

The workshop will be organized in several sessions. After an initial presentation session, where all participants can formulate one or more, possibly provocative, working hypotheses, we intend to split the workshop into task forces to foster the discussion a particular subject of common interest. At the end of the workshop the task forces will reunite and we will assemble the results and formulate future work, which we intend to present to the rest of the ECOOP community in the form of a poster at the conference, if possible.

Authors are encouraged to address any aspects of the design and implementation of composition languages in their position statements. We solicit submissions on (possibly) original research in the form of extended abstracts. Submissions should not exceed 8 pages (with a minimum 11pt font) and must have a cover page including the paper title,

abstract, names and affiliations of authors, postal contact addresses, email addresses, and telephone numbers. In addition, we particularly ask authors to include a list of critical questions and/or some, perhaps provocative, statements at the end of their submission which will assist the organizers to define topics for discussion in advance. Submissions should be sent in an electronic format (PDF or Postscript) to [lumpe@cs.iastate.edu](mailto:lumpe@cs.iastate.edu) and preferably prepared for letter or A4 sizes using Springer LNCS-style.

All selected submissions will be made available online prior to the workshop and be published as a technical report by one of the affiliated organizations. Aspects of the best position statements as well as the workshop results will be discussed in a chapter of the ECOOP Workshop reader. We are investigating having a special issue of a journal for revisions of selected papers after the workshop.

Workshop Home Page:

For further information about the workshop, please refer to the workshop home page at <http://www.cs.iastate.edu/~lumpe/WCL2004>.

Important Dates:

- Contribution submission: April 5, 2004
- Notification of acceptance: April 26, 2004
- Camera ready copy: May 17, 2004
- Workshop: June 14, 2004

Workshop Organizers:

Markus Lumpe, Iowa State University, USA  
Jean-Guy Schneider, Swinburne University of Technology, Australia  
Bastiaan Schönage, Compuware Europe B.V., The Netherlands