Welcome!

It is our pleasure to welcome you to ECOOP 2002, the 16th European Conference on Object-Oriented Programming. This year ECOOP is hosted by the University of Málaga, Spain, and runs from Monday June 10 to Friday June 14. We are confident that ECOOP 2002 will follow the success of the previous editions, becoming the major European forum for exchanging ideas and experiences in the very important field of Object Orientation.

The conference is organized by the Department of Computer Science of the University of Málaga and the Department of Computer Science of the University of Extremadura, under the auspices of AITO (Association Internationale pour les Technologies Objets). All events related to the conference will be held at the premises of the School of Computer Science (E.T.S.I. Informática) of the University of Málaga.

Monday and Tuesday are dedicated to tutorials and workshops. The main conference takes place from Wednesday to Friday, and hosts the technical program. This year, the technical program consists of 24 high-quality technical papers, two invited talks given by José Meseguer and Clemens Szyperski, and one panel. In this occasion we also feature two attractive birds-of-a-feather (BoF) meetings on Thursday afternoon. Demonstrations, posters, and exhibits happen in parallel with the technical program. In addition, we are proud of having Prof. Kristen Nygaard deliver the keynote speech at the conference banquet, one of the recipients of this year's IEEE John von Neumann Medal and ACM Turing Award.

Málaga, capital of the Costa del Sol, is one of the most cosmopolitan and open cities in Europe, widely known for its superb weather, food, and hospitality. Thus, an extensive social program has also been prepared to help all participants enjoy the visit and get acquainted with the Spanish and Andalusian culture.

Of course this conference would not have been possible without the help and dedication of many people, in particular the members of the Program Committee that assembled the technical program, the members of the Organizing Committee that worked for more than one year to make the conference happen, and many other anonymous individuals just wishing to help. We are indebted to all of them, as we are grateful to all the major companies and organizations which sponsor the conference.

Let us warmly thank you for your interest in ECOOP and for joining us in Málaga to participate in this exciting event. We wish you a nice stay and a very fruitful and productive conference.

José M. Troya
Conference Chair

Antonio Vallecillo
Organizing Chair

Table of Contents

Organization

ECOOP 2002 is the 16th edition of ECOOP, and is organized by the Department of Lenguajes y Ciencias de la Computación of the University of Málaga, and the Department of Informática of the University of Extremadura.

ECOOP 2002 will be hosted by the School of Computer Science of the University of Málaga. All events will happen at the School premises.

Executive Committee

Conference Chair:	José M. TROYA	University of Málaga
Programme Chair:	Boris MAGNUSSON	University of Lund, Sweden
Organizing Chair:	Antonio VALLECILLO	University of Málaga

Programme Committee

John BOYLAND	University of Wisconsin - Milwaukee, USA
Gilad BRACHA	Sun Microsystems, USA
Krzysztof CZARNECKI	DaimlerChrysler AG, Germany
Bjorn N. FREEMAN-BENSON	University of Washington, USA
Svend FRØLUND	HP Labs, USA
Erich GAMMA	OTI International, Switzerland
Görel HEDIN	Lund University, Sweden
Urs HOELZLE	University of California, at Santa Barbara
Eric JUL	University of Copenhagen, Denmark
Kai KOSKIMIES	Tampere University of Technology, Finland
Kresten KRAB THORUP	Trifork Technologies, Denmark
Doug LEA	State University of New York at Oswego, USA
Jørgen LINDSKOV KNUDSEN	Mjølner Informatics Inc., Denmark
Satoshi MATSUOKA	Tokyo Institute of Technology, Japan
Mira MEZINI	Technical University of Darmstadt, Germany
Ana MOREIRA	Universidade Nova de Lisboa, Portugal
Linda NORTHROP	Software Engineering Institute, USA
Arnd POETZSCH-HEFFTER	Fern Universität Hagen, Germany
Tore RICH	Uppsala University, Sweden
Houari A. SAHRAOUI	University of Montreal, Canada
Douglas C. SCHMIDT	University of California at Irvine, USA
Bran SELIC	Rational Software Corporation, USA
Jan VITEK	Purdue University, USA
John VLISSIDES	IBM T. J. Watson Research Center, USA
Wolfgang WECK	Oberon microsystems Inc., Switzerland

Organizing Committee

Tutorial Chairs:	Ernesto PIMENTEL	University of Málaga
	Hanspeter MÖSSENBÖCK	University of Linz, Austria
Workshop Chairs:	Juan HERNÁNDEZ	University of Extremadura
	Ana MOREIRA	Universidade Nova de Lisboa, Portugal
Exhibit Chair:	Manuel DÍAZ	University of Málaga
Demonstration Chair:	Lidia FUENTES	University of Málaga
Poster Chair:	Juan M. MURILLO	University of Extremadura
Panel Chair:	Francisco DURÁN	University of Málaga
Sponsorship Chair:	Javier LÓPEZ	University of Málaga
Advertizing, web Chair:	Fernando SÁNCHEZ	University of Extremadura
Registration Chair:	Carlos CANAL	University of Málaga
Operational Support:	José M. ÁLVAREZ	University of Málaga
	Mercedes AMOR	University of Málaga
	Manuel F. BERTOA	University of Málaga
	Alfonso GAZO	University of Extremadura
	Francisco GUTIÉRREZ	University of Málaga
	Pablo LÓPEZ	University of Málaga
	Luis LLOPIS	University of Málaga
	Antonio MAÑA	University of Málaga
	Antonio NEBRO	University of Málaga
	Mónica PINTO	University of Málaga
	Juan J. ORTEGA	University of Málaga
	Roberto RODRÍGUEZ	University of Extremadura
	Bartolomé RUBIO	University of Málaga
	Blas C. RUÍZ	University of Málaga

Tutorials

Traditionally, ECOOP proposes, as part of the technical program, different tutorials during the first two days of the conference. ECOOP is well known for its high-quality and attractive tutorials. This year, the tutorial chairs have organized 21 high-quality tutorials. The objective is to cover the different aspects of object-orientation, with special emphasis on the hot and novel topics. Your participation will contribute making these tutorial days a success. Tutorials have been scheduled around subjects, in order to allow participants to attend more than one tutorial.

Each tutorial lasts either one half day or one full day. Tutorials take place on Monday or Tuesday at the Conference venue, from 9:30 to 13:00 and 14:30 to 18:00. Tutorial rooms are located in building number 1 (close the main entrance)

	MONDAY							TUESDAY					
Morning	T01	T02	T03	T05	T07	T09		T13	T14	T15	T19	T21	T23
Afternoon			T04	T06	T08	T10	T12			T16	T18	T22	T24

ID	Title	Speakers	Monday		Tuesday	
			A.M.	P.M.	A.M.	P.M.
T01	**The Agile Unified Process**	Craig Larman	room 1.0.4			
T02	**O2C: A Semantic Thread From Objects to Components**	Farhad Arbab Frank S. de Boer Marcello M. Bonsangue	room 1.0.6			
T03	**Foundations of Object-Oriented Languages: Types and Language Design**	Kim Bruce	room 1.0.1			
T04	**Efficient Implementation of Object-Oriented Programming Languages**	Craig Chambers		room 1.0.1		
T05	**.NET and/or Java? - Step-by-Step Comparison**	Michal Stal	room 1.0.3			
T06	**Specifying and Achieving Non-Functional Requirements**	Lenn Bass Felix Bachmann		room 1.0.2		
T07	**Design metrics for UML users: using OCL to formalize metrics definitions**	Fernando Brito e Abreu Miguel Goulão	room 1.0.2			
T08	**Advanced Aspect Composition: Obstacles and The Composition Filters Approach**	Mehmet Aksit		room 1.0.7		
T09	**From Object-Oriented Programming to Model-Driven Software Production**	Jean Bézivin	room 1.0.5			
T10	**Java Security APIs: What is Going on Behind the Scenes?**	Rolf Oppliger		room 1.0.3		
T12	**Squeak: An Open Source Smalltalk for the 21st Century!**	Andrew P. Black		1.0.5		
T13	**Architecture-Centric Software Engineering**	Jan Bosch			room 1.0.2	

ID	Title	Speakers	Monday		Tuesday	
			A.M.	P.M.	A.M.	P.M.
T14	**Using and Adapting Extreme Programming**	Martin Lippert				room 1.0.5
T15	**Secrets of object-oriented component-based Middleware - Patterns for Concurrent and Networked Objects**	Michael Stal			room 1.0.1	
T16	**Aspect Oriented Programming with AspectJ**	Craig Larman				room 1.0.1
T18	**Generative Programming: Methods, Techniques, and Applications**	Krzysztof Czarnecki				room 1.0.4
T19	**The UML-F Profile for Framework Architectures**	Markus Fontoura Wolfgang Pree Bernhard Rumpe			room 1.0.3	
T21	**Accomplishing Software Stability**	Mohamed E. Fayad			room 1.0.6	
T22	**Patterns for building Component Infrastructures**	Markus Voelter				room 1.0.6
T23	**Concurrency: How Can I get It Work**	Frank Buschmann			room 1.0.4	
T24	**Built-in Testing for Component-based Development**	Hans-Gerhard Gross Peter Lay				room 1.0.3

T01: The Agile Unified Process

Presenter	Craig Larman (Valtech, USA)
Duration	Full day
Day	Monday
Room	1.0.4
Level	Introductory - Intermediate

Abstract:

The Unified Process (UP) has emerged as a popular modern process for software development and with good reason, as it includes skillful best practices such as iterative development, and early attention to high-risks. Furthermore, the UP process framework encourages flexible adoption, providing a structure that can scale up and down.

Contrary to some misperception, the UP encourages an agile approach. In this tutorial you will learn the essential and most useful UP concepts and practices, the keys to its successful introduction in an organization, and how to apply the UP in an adaptive and agile spirit.

Further, most Extreme Programming (XP) and Scrum practices and values are either part of the UP, or specializations of more general UP guidelines. What are these, and which may be adopted consistent with and within a UP project? Nevertheless, this tutorial motivates the need for more than pure XP on many projects, and promotes the skillful capacity of the UP to provide a variety of optional practices and artifacts on larger or higher-ceremony projects. You will learn why a combination of primarily the UP with some XP practices is an excellent process approach.

Topics include:

- The key UP ideas to know and apply.
- The motivation and business case for the UP.
- What UP artifacts are really worth creating?
- UP anti-patterns: common worst-practices in adoption and use.
- How to adopt the UP within an organization.
- XP practices within the UP.
- Challenges of pure XP.
- How to plan an iterative UP project.
- Tips for good UP artifacts and models.
- UP models and the UML.
- How to define a UP development case.
- How to do architectural analysis and describe architectures in the UP.
- How to fail with the UP: You know you didn't understand it when…

Expected Audience:

Open to a broad audience of those interested in adopting an agile approach to the UP.

Required Experience:

None specific; general software development experience.

Presenter's profile:

Craig Larman is an internationally recognized leader, speaker, and coach in adopting agile, iterative development processes, and object technologies. He is the author of "Applying UML and Patterns--An Introduction to Object-oriented Analysis and Design and the Unified Process," the best-selling text on

OOA/D and iterative development, available in many languages, and used in business and universities worldwide.

Craig serves as Director of Process at Valtech, an international consulting group. He has been using object technologies since 1984, and for many years has assisted others in developing object systems, adopting practical iterative development processes, and in learning to apply object technologies. He holds a B.Sc. and M.Sc. in computer science.

T02: O2C: A Semantic Thread From Objects to Components

Presenters	Farhad Arbab (CWI, The Netherlands) Frank S. de Boer (Univ. Utrecht, The Netherlands) Marcello M. Bonsangue (Leiden University, The Netherlands)
Duration	Full day
Day	Monday
Room	1.0.6
Level	Intermediate

Abstract:

In this tutorial we present the basic concepts that underlie object oriented and component based software engineering and their semantic justifications. We start with the basic concepts such as abstract data types and inheritance, as used in the object oriented paradigm to enhance reuse, modularity, and maintenance of software. We show how the concept of component supports and generalizes similar concerns in the engineering of large, heterogeneous, loosely-coupled, distributed software systems. The current component technology regards components as extended objects. We describe an alternative approach that starts with the concept of components as abstract behavioral types. We show how this new interpretation makes components amenable to explicit exogenous coordination, supports compositionality, and provides a clear separation between computation and communication concerns. This leads to a new model of component composition that is based on a calculus of connectors for algebraic construction of component glue code.

Expected Audience:

This tutorial is intended for software engineers and programmers interested or involved in building large component based software systems, especially for distributed heterogeneous platforms. It is also intended for PhD students and researchers interested in formal methods and semantic foundations of component composition and component based systems. The audience will be exposed to semantic justifications for a variety of modern software engineering concepts that will enhance their appreciation of component based approaches.

Required Experience:

Familiarity with Object Oriented Programming concepts and some familiarity with formal methods.

Presenter's profile:

The presenters are acknowledged experts in formal methods, logics, and semantics of concurrency, object oriented programming, coordination, and component based software, with a track record of joint research in experimental and theoretical computer science.

T03: Foundations of object-oriented languages: Types and Language Design

Presenter	Kim Bruce (Williams College, USA)
Duration	Half day
Day	Monday - morning
Room	1.0.1
Level	Advanced

Abstract:

Static typing aids in earlier error detection, supports compiler optimizations, and provides information to programmers on the intended use of constructs. However, simple static-typing disciplines for object-oriented languages like C++ and Java are so restrictive that programmers are forced to by-pass the type system with type casts. Other languages allow more freedom, but require run-time checking to pick up the type errors that their more permissive systems missed. After surveying problems with existing type systems (illustrated by a series of sample programs), we discuss subtype and subclass restrictions, and suggest ways of improving the expressiveness of object-oriented languages while retaining static type safety. Advanced constructs discussed include `"MyType", "matching", and "bounded polymorphism". We include a brief discussion on how the type system and semantics ensure type safety. We apply the concepts in the tutorial to compare the strengths and weaknesses of proposals to extend Java to support genericity based on F-bounded polymorphism, "where" clauses, match-bounded polymorphism, and virtual types.

Expected Audience:

The tutorial is aimed at programmers who would like the flexibility of weakly or dynamically typed languages, but the error detection and efficiency of statically-typed languages, as well as language designers and others who would like to see the applications of type theory to language design.

Required Experience:

Attendees should be very comfortable with the concepts of and programming in class-based object-oriented programming languages and experienced with static typing systems. Frustration with the rigidities of statically-typed object-oriented languages will be a helpful motivator.

Presenter's profile:

Kim Bruce is Wells Professor of Computer Science at Williams College in Massachusetts, USA. He received his B.A. from Pomona College and Ph.D. from the University of Wisconsin. He taught at Princeton University before coming to Williams, and has been a visiting professor or scientist at M.I.T., Stanford, Ecole Normale Superieure, the University of Pisa, the Newton Institute for Mathematical Sciences at Cambridge University, and Princeton. He has worked on the types and semantics of object-oriented languages for over 15 years. He has served twice on the OOPSLA program committee, and served as chair of the organizing committee of the FOOL workshops on the Foundations of Object-Oriented Languages from 1993 to 2001. He has presented papers at the ECOOP, OOPSLA, POPL, MFPS, and LICS conferences. He is the author of the forthcoming book, Foundations of Object-Oriented Languages: Types and Semantics, to be published in early 2002 by MIT Press.

T04: Efficient Implementation of Object-Oriented Programming Languages

Presenter	Craig Chambers (Univ. Washington, USA)
Duration	Half day
Day	Monday - afternoon
Room	1.0.1
Level	Advanced

Abstract:

How are object-oriented languages implemented? What features of object-oriented languages are expensive? What compiler optimizations have been developed to make object-oriented languages more efficient? This tutorial addresses these questions. After identifying the main features of object-oriented languages that are challenging to implement efficiently, three classes of implementation techniques are presented. First, run-time system techniques such as virtual function dispatch tables (including complications due to multiple inheritance and virtual inheritance) and inline caches are described. Second, static intra- and interprocedural analyses are discussed that seek to identify at compile-time the possible classes of message receivers, in order to reduce or eliminate the overhead of dynamic binding. Third, ways in which dynamic execution profiles can be exploited to complement static analysis techniques are described. To assess the relative importance of the techniques, empirical measurements of the effectiveness of many of these techniques, as implemented in the Vortex optimizing compiler, are presented for large benchmarks written in Java, C++, and Cecil.

Expected Audience:

Attendees are expected to be interested in optimizing compiler strategies for OO languages. Attendees will become familiar with the issues and state-of-the-art techniques for implementing object-oriented languages efficiently.

Required Experience:

Attendees should be familiar with the features of object-oriented languages and also with traditional compiler techniques such as procedure inlining and data flow analysis.

Presenter's profile:

Craig Chambers has been researching object-oriented language design and implementation since 1987, with publications in OOPSLA, ECOOP, ISOTAS, PLDI, POPL, PEPM, and TOPLAS on the topic. For his Ph.D. thesis at Stanford, he developed the first efficient implementation of the Self language, using optimizing dynamic compilation. Chambers is currently an Associate Professor of Computer Science & Engineering at the University of Washington, where he designed the Cecil language and co-designed the MultiJava and ArchJava languages, heads the Vortex whole-program optimizing compiler project and the Whirlwind staged compiler project, and co-leads the DyC selective dynamic compilation project.

T05: .NET and/or Java? - Step-by-Step Comparison

Presenter	Michael Stal (Corporate Tech., Germany)
Duration	Half day
Day	Monday - morning
Room	1.0.3
Level	Intermediate

Abstract:

Both Java and Microsoft .NET offer complete frameworks to build and deploy desktop applications and enterprise applications without dependencies on the concrete platform used. Both of them now focus on the delivery of Web services to build Web-based applications in a much more sophisticated way. For the developer it is very difficult to decide which of these technologies to use. What are the benefits and liabilities of these solutions? What are their similarities and differences. It is the goal of the tutorial to reveal all of these issues.

Expected Audience:

Developers and software architects interested in J2EE and Microsoft .NET.

Required Experience:

Sound knowledge in Object-Oriented Programming Java or C# skills might be helpful

Presenter's profile:

Michael Stal is Senior Principal Engineer at Siemens responsible for research on Middleware and Application Integration. Michael focuses on Distributed Objects & Components, Software Architecture, and Web technologies. He is co-author of Pattern-Oriented Software Architecture Volume I&II, Siemens representative at the OMG, former member of the C++ X3J16 standardization working group, and editor-in-chief of Java Spektrum.

T06: Specifying and Achieving Non-Functional Requirements

Presenters	Lenn Bass (Carnegie Mellon Univ., USA) Felix Bachmann (Carnegie Mellon Univ. USA)
Duration	Half day
Day	Monday - afternoon
Room	1.0.2
Level	Introductory

Abstract:

The availability, modifiability, performance, and security requirements for a system are difficult both to capture and achieve. Use cases are well recognized as a means for capturing functional requirements but are not appropriate for the specification of non-functional requirements. Achieving a design that satisfies these requirements is even more difficult In this tutorial, we present recent work in both of these areas. We present a characterization of non-functional requirements for the four quality attributes listed. We also present architectural patterns that are used to achieve these qualities. Our characterization is based on the concept of "quality scenario" where each scenario includes a stimulus and an expected response to that stimulus. We present a set of quality scenarios that have been validated against scenarios seen in the "wild" and discuss the concepts behind these scenarios. Explanations of how particular architectural patterns achieve these requirements will also be given.

Expected Audience:

Those who have responsibilities that involve the specification or achievement of requirements for software systems should attend this tutorial. They will get a systematic explanation of how to specify non-functional qualities and the strategies and patterns that are used by designers to achieve them.

Required Experience:

This tutorial is oriented toward those who have some experience with software design and development.

Presenter's profile:

Len Bass and Felix Bachmann are well-known software architects at The Software Engineering Institute at Carnegie Mellon University. They jointly developed the Attribute Driven Design Method for designing architectures. In addition, Len was a developer of the first scenario based architecture evaluation method (SAAM - Software Architecture Analysis Method) and an author of the popular "Software Architecture in Practice". Both Felix and Len are co-authors of the upcoming book "Documenting Software Architecture".

T07: Design metrics for UML users: using OCL to formalize metrics definitions

Presenters	Fernando Brito e Abreu (Lisbon New Univ., Portugal) Miguel Goulão (Lisbon New Univ., Portugal)
Duration	Half day
Day	Monday - morning
Room	1.0.2
Level	Advanced

Abstract:

In this tutorial we will present an approach that solves the metrics ill-definition problem. The UML meta-model is used as context and OCL operations defined on it are used in a compositional fashion to express design metrics. The metrics applicability limitations are defined with OCL pre-conditions. The metrics result itself is formally defined with OCL post-conditions. The outcome is an elegant, precise and straightforward way to define metrics that may help to overcome several current problems. Besides, it is a natural approach since we are using object technology to define metrics on object technology itself. To illustrate this approach, we will present several examples taken from well-known metric sets and will work on hands-on examples with the participants. A demonstration of the validation of a design metrics set using this approach will be performed. A library of basic OCL functions for composition of OO metrics will be made available to all tutorial participants.

Expected Audience:

Required Experience:

Some knowledge on software metrics, namely design ones. Basic knowledge of UML and OCL syntax and semantics.

Presenter's profile:

Dr. Brito e Abreu is professor of object-oriented design and programming at the Lisbon New University (http://di.fct.unl.pt). He holds a PhD on Computer Science and a MSc on Telecommunications and Computer Engineering, both from the Lisbon Technical University (http://www.ist.utl.pt). He is a researcher in the Software Engineering Group at INESC ID where he leads a team on Experimental Software Engineering (http://www.esw.inesc.pt/mood). INESC is a private non-profit association dedicated to research, development and training in advanced technological areas, that acts as an interface between the Portuguese Telecommunications and Information Technology sectors and the University system (http://www.inesc-id.pt). Since 1998 he is also a member of the international team of professors that deliver the EMOOSE - European Master on Object Orientation and Software Engineering, created in the scope of the European Commission ALFA Program. There, he is co-responsible for the course "Software Quality" (http://www.emn.fr/emoose). Since December 1999, he presides the Portuguese Information Technologies and Telecommunications Quality Commission (CS03) of the National Council for Quality (CNQ). He is also a member of the Software Group (http://www.eoqsoftware.org) of EOQ - European Organization for Quality (http://www.eoq.org), since April 1999. Dr. Brito e Abreu is also member of the Editorial Board of the Software Quality Professional journal (http://sqp.asq.org), published by the American Society for Quality (http://www.asq.org), since September 2000. He is author or co-author of over 30 communications presented at national and international workshops, conferences and symposiums such as ECSQ, ICSM, ICSQ, CSMR, AQUIS, ESCOM, TOOLS or QUATIC. He also has more than 20 papers published on journals such as: Journal of Systems and Software, Object Expert, ERCIM News, Personal Computer World, L'Objet, Qualirama, Sistemas de Informação and Interface. He is a member of the steering committee of the CSMR conference. He was the general organizing chair of CSMR'2001 held in Lisbon (http://www.esw.inesc.pt/csmr2001) and is the program co-chair of CSMR'2002 to be held in Budapest (http://rgai.inf.u-szeged.hu/CSMR2002). Besides being a panel member and session chairman at several occasions, he served as member of the program committee at CAPSI'2001, ICSM'2001, ASSE'01, IWSM'2001, METRICS'2001, CSMR'2001, QUATIC'2001, TOOLS'2000 Pacific, CAPSI'2000, 2WCSQ, CSMR'2000, CSMR'1999, QUATIC'98, CSMR'1998, AQUIS'96, QUATIC'95 and QUATIT'94. Since 1995 he has also been organizing several workshops on quantitative approaches for object-oriented systems in conferences such as OOPSLA and ECOOP. Dr. Brito e Abreu is the author of the MOOD and MOOD2 (Metrics for Object Oriented Design) sets and of a technique for reengineering the modularity of object-oriented systems.

Miguel Goulão is currently with the Department of Computer Science, Faculty of Sciences and Technology, Lisbon New University, where he holds a position of teaching assistant (http://di.fct.unl.pt). He is also a researcher of the Software Engineering Group (http://www.esw.inesc.pt) of INESC-ID, a leading R&D organization in Portugal. He has participated in several joint projects of INESC and other organizations, mainly with the Portuguese Navy. He is also a member of the Portuguese Commission for Quality in Information Technologies. Miguel holds a MSc degree from the Lisbon Technical University (http://www.ist.utl.pt). His MSc thesis was about the usage of a quantitative approach to support software process improvement. His research interests are mainly focused in the realm of Empirical Software Engineering and in enabling technologies for the facilitation of software maintenance and reengineering. He has been working in this field for the past 6 years and is now starting his PhD work. He is author of several research articles presented at international conferences and published in peer-reviewed journals. He has been awarded as a co-author of the best paper on software metrics presented in the 6th European Conference on Software Metrics, in 1999. Miguel has also been a member of the organizing committee of two international conferences in 2001, CSMR'2001 (http://www.esw.inesc.pt/csmr2001) and QUATIC'2001 (http://www.esw.inesc.pt/quatic2001) and often participates as an invited reviewer in the paper evaluation process of several conferences.

T08: Advanced Aspect Composition: Obstacles and The Composition Filters Approach

Presenter	Mehmet Aksit (University of Twente, The Netherlands)
Duration	Half day
Day	Monday - afternoon
Room	1.0.7
Level	Advanced

Abstract:

The object-oriented paradigm has been successful because of its good modularity characteristics, which supports the separation of concerns from analysis to implementation phases. Composability problems may be experienced when constructing new objects from existing ones, for example when objects need to evolve due to new or changing requirements. This experience has triggered researchers in the past years to come up with enhancements of the OO model in an attempt to solve the composability problems. The most well-known examples in this area are: AOP/AspectJ, Subject-Oriented Programming & HyperJ, Adaptive Programming and Composition Filters. Especially, the so-called non-functional properties of software, such as access-control, history sensitivity, synchronization and real-time behavior, make it difficult to adapt and reuse software components. This tutorial first presents an illustrative example, which evolves due to the changing requirements. Various versions of this example have been implemented in a number of languages such as C++, Java and CORBA. The tutorial will clearly illustrate the limitations of these languages and design patterns in coping with the changing requirements. The origin of the problems will be analyzed in detail. As a solution to the obstacles various approaches will be discussed. The tutorial will further focus on the composition filters approach and will illustrate its advantages, limitations and practical applicability in real projects.

Expected Audience:

This tutorial is intended for software professionals and researchers who want to gain an understanding of the origins, issues and approaches for advanced software composition and the composition filters approach.

Required Experience:

Experience on object oriented languages is required.

Presenter's profile:

Mehmet Aksit and Lodewijk Bergmans have both been working on software composition for over a decade. Their work has included the analysis of inheritance anomaly for synchronization and real-time specifications (composability problems) and the composition filters approach to solve such issues. They have been involved in the organization of most workshops in this area. Both are experienced teachers who have together given over 100 professional (open/international/in-company) courses. Mehmet Aksit has given several tutorials during ECOOP and OOPSLA in the past. (see also http://trese.cs.utwente.nl/aksit, http://trese.cs.utwente.nl/bergmans)

T09: From Object-Oriented Programming to Model-Driven Software Production

Presenter	Jean Bézivin (Univ. Nantes, France)
Duration	Half day
Day	Monday - morning
Room	1.0.5
Level	Intermediate

Abstract:

This tutorial covers the new software development framework recently proposed by the Object Management Group. The Model-Driven Architecture departs from traditional code-centric approaches to explore new practical ways to separate platform-independent business models from platform-specific design and implementation models.

Expected Audience:

Practitioners and researchers wanting to understand the present paradigm shift from objects to models, as implemented in the OMG move from OMA (Object Management Architecture) to MDA (Model-Driven Architecture)

Required Experience:

Presenter's profile:

Jean Bézivin is professor of Computer Science at the University of Nantes, France. He got his Master degree from the University of Grenoble and PhD from the University of Rennes before spending several years, as an assistant professor, at the University of Brest. He also spent one year as a research fellow at the Queen's University of Belfast (Northern Ireland) and one year at the Concordia University of Montreal (Canada). He has been very active in Europe in the object-oriented community, starting the ECOOP series of conference (with P. Cointe), the TOOLS series of conferences (with B. Meyer), the OCM meetings (with S. Caussarieu and Y. Gallison) and more recently the <<UML>> series of conferences (with P.-A. Muller). He also organized several workshops at OOPSLA like in 1995 on "Use Case Technology", in 1998 on "Model engineering with CDIF" and more recently at ECOOP in 2000 on "Model Engineering". He started in 1979 at the University of Nantes, one of the first Master programs in Software Engineering entirely devoted to Object Technology (Data Bases, Concurrency, Languages and Programming, Analysis and Design, etc.). He has expertise in the fields of object technology, analysis and design, model engineering, etc. He published various papers on subjects related to concurrency, software engineering, simulation, object technology, etc. He also presented tutorials on related subjects at several international conferences. His present research interests include object-oriented analysis and design, product and process modeling.

T10: Java Security APIs: What is Going on Behind the Scenes?

Presenter	Rolf Oppliger (eSECURITY Technologies, Switzerland)
Duration	Half day
Day	Monday - afternoon
Room	1.0.3
Level	Introductory

Abstract:

The tutorial provides an overview about the Java security application programming interfaces (APIs) and the security technologies they implement. As such, the primary goal of the tutorial is to provide the background information that is required to understand what is going on behind the scenes and to properly make use of the Java security APIs.

Expected Audience:

This tutorial is indeed for researchers and practitioners who want to get a more thorough understanding of the security technologies that are implemented in the Java security APIs.

Required Experience:

None. The security technologies that are implemented in the Java security APIs are explained from scratch.

Presenter's profile:

Rolf Oppliger, Ph.D., is the founder and owner of eSECURITY Technologies (http://www.esecurity.ch). In addition, he works for the Swiss Federal Strategy Unit for Information Technology (FSUIT) and teaches at the University of Zürich. He has authored eight books, including, for example, Security Technologies for the World Wide Web (Artech House, 2000), and the second edition of Internet and Intranet Security (Artech House, 2002), frequently speaks at security-related conferences, and regularly publishes papers and articles in scientific magazines and journals. He is the current editor of Artech House's book series on computer security (http://www.esecurity.ch/serieseditor.html).

T12: Squeak: An Open Source Smalltalk for the 21st Century!

Presenter	Andrew P. Black (School of Science & Eng., USA)
Duration	Half day
Day	Monday - afternoon
Room	1.0.5
Level	Introductory

Abstract:

Squeak is an open source Smalltalk system that runs on any platform (including Windows, MacOS, Linux, and many PDAs). It is fun to use, has a large, talented and enthusiastic user community, and is constantly improving. Because all the source code is written in Squeak itself and is freely available, anything can be changed to suit the needs of the programmer. Squeak is ideal as an experimental, exploratory environment, as well as for teaching. This tutorial is intended for those familiar with object-oriented concepts and design, and who are keen to explore the richness of a 21st Century Smalltalk. The

tutorial will be "hands-on"; participants are encouraged to bring their own laptop computers loaded with Squeak. We will cover the essential aspects of conventional Smalltalk, such as the programming environment, debugging, and testing, and will emphasize Squeak's innovations, such as the Morphic graphic model, 3D, book-morphs, scripting, and Sound.

Expected Audience:

Professionals who have learned other O-O languages and are disappointed that they do not offer the hoped-for paradigm shift. Educators who are considering using Squeak as a medium for teaching OO concepts, UI-design, Multimedia or Graphics. Engineers and managers who are curious to know why a 20 year old language is generating so much excitement. Anyone looking for a productive programming environment with a vibrant and friendly user community in order to put the fun back into programming. After the tutorial, participants will understand the potential of Squeak, will be able to write simple graphical applications, and will know how to set about exploring the Squeak world more thoroughly.

Required Experience:

Participants should know about O-O concepts and O-O design. They are likely to have programmed in another O-O language, such as C++ or Java, or perhaps another Smalltalk. No previous experience with Squeak is assumed.

Presenter's profile:

Andrew Black is a Professor at the OGI School of Science & Engineering, Oregon Health & Science University, located just outside Portland, Oregon, USA, where he has taught MS and PhD students Object-Oriented Programming using Smalltalk and other languages since 1994. He has been involved in OO language and systems research since 1981.

T13: Architecture-centric Software Engineering

Presenter	Jan Bosch (Univ. Groningen, The Netherlands)
Duration	Full day
Day	Tuesday
Room	1.0.2
Level	Intermediate

Abstract:

Many software organizations are in the process of moving from project-centric to architecture centric engineering of software. The two typical reasons for this move are (1) the architecture allows for a clear break-down in parts whereas a project-centric approach easily leads to a monolithic system and (2) the organization is interested in exploiting the commonalities between its products or systems. This tutorial addresses this development by providing an overview and in depth treatment of the issues around architecture-centric engineering of software. Topics include software architecture design in the presence of existing components and infrastructure (top-down versus bottom-up), architecture evalua-tion and assessment, software artefact variability management, software product lines and the role of the software architect. These topics are, in addition to the technical perspective, discussed from process and organizational viewpoints. The topics are extensively illus-trated by examples and experiences from many industrial cases.

Expected Audience:

The expected audience can be divided into two categories. First, software engineers and technical managers considering the introduction of architecture-centric software develop-ment and evolution. Second, researchers interested in the experiences collected by the tutorial presenter and his research group and the reflections made based on the experiences.

Required Experience:

It is assumed that the participant has some experience with industrial software develop-ment.

Presenter's profile:

Prof. dr. ir. Jan Bosch is a professor of software engineering at the University of Groningen, The Netherlands, where he heads the software engineering research group. He received a MSc degree from the University of Twente, The Netherlands, and a PhD degree from Lund University, Sweden. His research activities include software architecture design, software product lines, object-oriented frameworks and component-oriented programming. He is the author of a book "Design and Use of Software Architectures: Adopting and Evolving a Product Line Approach" published by Pearson Education (Addison-Wesley & ACM Press), (co)editor of three volumes in the Springer LNCS series and has (co)authored more than 50 refereed journal and conference publications. He has organized numerous workshops, served on many programme committees, including the ICSR'6, CSMR'2000, ECBS'2000, GCSE, SPLC and TOOLS conferences and is member of the steering groups of the GCSE and WICSA conferences. He is the PC co-chair of the 3rd IFIP (IEEE) Working Conference on Software Architecture (WICSA-3).

T14: Using and Adapting Extreme Programming

Presenter	Martin Lippert (Univ. Hamburg, Germany)
Duration	Full day
Day	Tuesday
Room	1.0.5
Level	Intermediate

Abstract:

Extreme Programming (XP) is a hot topic. It promises high productive on-time software development with minimum risks. The XP techniques to achieve these goals sound both interesting and frightening: interesting from the programmer's viewpoint because XP seems to provide a relaxed, fun and free way to do the development, frightening for the management because XP looks chaotic, unplanned and unpredictable. We have done a number of successful XP projects in various application domains. The tutorial contains a condensed version of our experiences with XP in different settings. Taking as an example a long-term project within an insurance company, we discuss the benefits derived from XP and how we adapted the XP techniques. The participants get an idea of how we integrated the two opposing viewpoints of developers and managers within our projects by using lightweight additional techniques like project stages. We propose a set of best-practice methods we have used in a number of industrial high-risk projects for different application domains.

Expected Audience:

The tutorial is aimed at project leaders, software architects and experienced software developers who are interested in extreme programming but may be skeptical about using XP for their projects. Participants will get a good idea of how to adapt extreme programming to complex application domains and how to benefit from extreme programming within a development project. They will benefit from the XP experience of the presenters.

Required Experience:

The participants should have experience with object-oriented analysis and design. Extreme programming experiences are not expected.

Presenter's profile:

Martin Lippert is a research assistant at the University of Hamburg and a professional software architect and consultant at IT Workplace Solutions. Among his current research interests are framework design and implementation, tool support for framework specialization, refactoring, Java, and extreme programming. He is a senior architect of the JWAM framework and has gathered experience with extreme programming techniques over the past 3 years. He is a project coach for extreme programming and software architectures and has given a number of talks, tutorials and demonstrations on various topics of software engineering at international conferences including ICSE, ECOOP, eXtreme Programming, OOPSLA, HICSS, ESEC/FSE, ICSTest and OOP, especially XP tutorials at ECOOP 2001, OOP 2002 and ICSTest 2002. Among his publications are articles for conference proceedings, books (Extreme Programming Examined, Extreme Programming Perspectives, Software Quality and Software Testing in Internet Times) and journals. He is a member of the XP 2002 program committee. Stefan Roock is a research assistant at the University of Hamburg and professional software architect and consultant at APCON Worplace Solutions. Among his current research interests are framework design and implementation and extreme programming. He his a senior architect of the JWAM framework and has gained experience with extreme programming techniques over the past 3 years. He is a project coach for extreme programming and software architectures and has given a number of talks and demonstrations on various topics of software engineering (e.g. 3 day OO workshop (Roennby/Sweden), framework tutorial at OOP 2001 (Munich/Germany), eXtreme Programming 2000 (paper presentation, Cagliari/Italy), XP tutorial at ECOOP 2001, XP tutorial at OOP 2002).

T15: Secrets of object-oriented component-based Middleware - Patterns for Concurrent and Networked Objects

Presenter	Michael Stal (Corporate Tech., Germany)
Duration	Half day
Day	Tuesday - morning
Room	1.0.1
Level	Advanced - Intermediate

Abstract:

Nowadays, most software applications are developed to run on networks or on the Internet. However, to build sophisticated distributed systems in an efficient and an effective way a lot of experience and knowledge is required from developers. Patterns are the perfect means to document the experience of professional developers in this area. Hence, the goal of this tutorial is to introduce a web of important patterns that together explain how to build distributed, concurrent systems with quality in mind. The patterns introduced will not only help to build software applications but also to understand the "secrets" of object-oriented middleware and component-based from an architecture perspective.

Expected Audience:

Developers and software architects interested in building and understanding object-oriented and component-based distributed systems.

Required Experience:

Sound knowledge in Object-Oriented Programming Basic skills in distribution technologies Java or C++ skills might be helpful.

Presenter's profile:

Michael Stal is Senior Principal Engineer at Siemens responsible for research on Middleware and Application Integration. Michael focuses on Distributed Objects & Components, Software Architecture, and Web technologies. He is co-author of Pattern-Oriented Software Architecture Volume I&II, Siemens representative at the OMG, former member of the C++ X3J16 standardization working group, and editor-in-chief of Java Spektrum.

T16:Aspect-Oriented Programming with AspectJ

Presenter	Craig Larman (Valtech, USA)
Duration	Half day
Day	Tuesday - afternoon
Room	1.0.1
Level	Introductory - Intermediate

Abstract:

Aspect-oriented programming (AOP) is a technique for improving separation of concerns in software design and implementation, and an intriguing new approach to thinking about design and programming, falling broadly within the emerging trend of Generative Programming techniques.

AOP works by providing explicit mechanisms for capturing the structure of crosscutting concerns. AspectJ is a seamless aspect-oriented extension to Java™. It can be used to cleanly modularize the crosscutting structure of concerns such as exception handling, multi-object protocols, synchronization, performance optimizations, and resource sharing. When implemented in a non-aspect-oriented fashion, the code for these concerns typically becomes spread out across entire programs. AspectJ controls such code-tangling and makes the underlying concerns more apparent, making programs easier to develop and maintain.

This tutorial will introduce aspect-oriented programming and show how to use AspectJ to implement crosscutting concerns in a concise, modular way. It includes numerous examples to develop participants' understanding of AOP through AspectJ. It will also demonstrate AspectJ's integration with IDEs such as JBuilder.

AspectJ is freely available at http://www.aspectj.org.

Participants may optionally wish bring a laptop installed with AspectJ and integration with one of its supported IDEs, as there may be a few short programming experiments to add some fun.

Expected Audience:

Introductory-Intermediate.

Required Experience:

Understanding of object-oriented programming, and Java.

Presenter's profile:

Craig Larman is an internationally recognized leader, speaker, and coach in iterative processes, OOA/D, and object technologies. He is the author of "Applying UML and Patterns--An Introduction to Object-oriented Analysis and Design and the Unified Process," the best-selling text on OOA/D and iterative development, available in many languages, and used in business and universities worldwide. Craig serves as Director of Process at Valtech, an international consulting group. He has been using object technologies since 1984, and for many years has assisted others in developing object systems, adopting practical iterative development processes, and in learning to apply object technologies. He holds a B.Sc. and M.Sc. in computer science.

T18:Generative Programming: Methods, Techniques, and Applications

Presenter	Krzysztof Czarnecki (DaimlerChrysler Research & Technology, Germany)
Duration	Half day
Day	Tuesday - afternoon
Room	1.0.4
Level	Intermediate

Abstract:

Have you ever run into the question of how to scope a component to make it reusable? Or how to achieve flexible designs without paying the performance cost for abstraction? Or how to encapsulate abstract features such as performance requirements in the code? Generative Programming (GP) addresses these and many more issues in developing reusable software. Building on the system-family approach, it complements object-oriented methods with notations and techniques to perform domain scoping and feature modeling. It also provides techniques for deriving a common family architecture and components. Finally, it deploys generative technologies to automatically assemble components based on specifications that programmers can conveniently express in their application code. Participants will learn the basic concepts of GP, and how to perform feature modeling, derive components and architectures from feature models, design domain-specific languages, and implement generators using widely available techniques such as XML and Java technologies or C++ template metaprogramming. After presenting some basic concepts and a small, but complete example, each step of the GP process will be discussed in more depth, and the participants will have a chance to experience it in hands-on-exercises. We will round up the tutorial with an outlook on future, advanced generative technologies such as active libraries and active sources.

Expected Audience:

This tutorial is aimed at researchers and practitioners interested in cutting-edge approaches to achieve reusability and adaptability. Participants will gain an understanding of concepts and principles of Generative Programming and its connections to related approaches such as Product-Line Engineering and Aspect-Oriented Programming. They will also learn the effective use of modeling techniques for product lines and advanced generative Java and C++ programming techniques. The case study provides a ready-to-use, comprehensive approach to developing reusable software.

Required Experience:

Being able to read Java and C++ code would be helpful (but not required)

Presenter's profile:

Krzysztof Czarnecki is a researcher and consultant with the Software Technology Lab at DaimlerChrysler Research in Ulm, where he has been working on generative programming and its industrial application for over five years. He co-authored (together with Ulrich Eisenecker) the book "Generative Programming: Methods, Tools, and Applications", Addison-Wesley, 2000.

T19:The UML-F Profile for Framework Architectures

Presenters	Markus Fontoura (IBM Almaden Research Center, USA) Wolfgang Pree (Univ. of California, USA) Bernhard Rumpe (Munich Univ. of Techn., Germany)
Duration	Half day
Day	Tuesday - morning
Room	1.0.3
Level	Intermediate

Abstract:

The Unified Modeling Language (UML) community has started to define various profiles for the UML language family to better suit the needs of specific domains or settings. Object and component frameworks represent a special breed of object-oriented systems-they are extensible semi-finished pieces of software. Completing the semi-finished software leads to various software pieces, typically specific applications, that share the same core. However, UML does not provide adequate constructs to model frameworks.

The intention of the UML-F profile for framework architectures is the definition of a UML subset, enriched with a few UML-compliant extensions, that allows the annotation of such artifacts. The tutorial presents the UML-F profile for framework architectures, called UML-F. It supports framework modeling and annotations of its variation points. It offers a concise set of notational elements for a static and dynamic description of the structure and semantics of a framework's variation points. Case studies round out the tutorial by illustrating the usage of UML-F in the realm of state-of-the-art frameworks.

Expected Audience:

The tutorial's main audience are software developers, students, researchers and project managers interested in framework technology. The main goal of the tutorial is to help practitioners that want to develop and use frameworks, by describing UML-based techniques and exemplifying their usage by real world case studies. The audience will (1) understand how UML is adapted to the specific needs of framework development and (2) benefit from several examples and case studies of real-world frameworks.

Required Experience:

Intermediate (assumes the knowledge of OO concepts, basic UML, reading knowledge of a C-style OO language such as Java/C#/C++, and ideally an understanding of the design patterns published by Gamma et al.)

Presenter's profile:

Marcus Fontoura had led several framework projects and specializes in Web-based software development and service-oriented architectures in the realm of IBM's Almaden Research Center. Before that he has held research postitions at the Computer Systems Group of the University of Waterloo, Canada, and at Princeton University's Computer Science Department.

Wolfgang Pree is guest professor at the University of California, Berkeley, moving to the University of Salzburg, Austria, in 2002. He has worked for several years in various areas of software engineering, in particular focusing on object technology, software architectures, frameworks, and human-computer interaction. Wolfgang is the author of Design Patterns for Object-Oriented Software Development (Addison-Wesley/ACM Press, 1995).

Bernhard Rumpe is organizing research on UML and agile modeling processes at the Technische Universität München. He has worked as a technical lead and manager in projects covering object-technology, frameworks, modeling notations, fast and efficient software processes like eXtreme Programming, and interoperability techniques. Bernhard is involved in the organization of the UML conference series since 1999 and is editor-in-chief of the new Journal on Software and System Modeling (published by Springer).

The presenters are the authors of a book with the same title of this tutorial that has been published by Addison-Wesley in November 2001.

T21:Accomplishing software stability

Presenter	Mohamed E. Fayad (Univ. Nebraska, USA)
Duration	Half day
Day	Tuesday - morning
Room	1.0.6
Level	Intermediate

Abstract:

There is little doubt that software engineering, like all other engineering fields, has helped to make life what it is today. With software controlling more equipment and becoming an integral part of more of our lives, the field of software engineering is quickly becoming more and more important. Unlike many other engineering fields, however, the products produced through software engineering are largely intangible. Also, unlike the products of other engineering fields, software products are unlikely to remain stable over a long period of time.

In hardware areas, the failure rates of products often start high, then drop low, and then go high again. Early in a hardware product's lifecycle, there are some problems with the system. As these problems are fixed, the failure rate of the hardware products drops. However, as hardware gets old, physical deterioration causes the hardware to fail. In other words, the hardware wears out and the failure rate rises again.

Software, on the other hand, is not subject to the same wear and tear that hardware is. There are no environmental factors that cause software to break. Software is a set of instructions, or a recipe, for a piece of hardware to follow. There are no moving parts in software. There is nothing that can physically deteriorate. Software should not wear out. Unfortunately, it does. Countless authors in the field of

software engineering have identified this problem. However, the software engineering techniques outlined by many software-engineering authors have not achieved a good amount of stability in software projects.

This problem is more than just an inconvenience for software engineers and software users. The reengineering that is required for these software products do not come without a price. It is not uncommon to hear of these reengineering projects costing hundreds of thousands to millions of dollars. This does not take into account the time that is wasted by this continual reengineering process.

Software defects and "deterioration" are caused by changes in software. Many of these changes cannot be avoided. These changes can be minimized, however. Currently, when a change must be made to a software program, most of the time the entire program is reengineered. It does not matter if the change required is due to new technology or a change in clientele. This reengineering process is ridiculous. The core purpose of the software product has not changed. Why, then, must the entire project be reengineered to incorporate a change?

This tutorial will examine software stability with respect to three central themes: "How can we engineer software systems that are stable overtime?," "What are the approaches of making software systems stable over time?" and "What is the role of object-oriented technology in the issue of software stability over time?."

The tutorial will answer the following questions:

1. How can we achieve software stability over time and extend the life span of software products?
2. What are the relationships between software architecture and software that has been stable over time?
3. What are the relationships between software that has been stable over time and workflow management?
4. What are the relationships between software that has been stable over time and business objects?
5. What is the role of object-oriented techniques and technologies in making software stable over time?
6. What are the approaches to making software stable over time?

Expected Audience:

Participants should have a general familiarity with basic object-oriented concepts, software engineering principles, software modeling techniques, (OMT, UML, or any object-oriented method) and software architecture. This tutorial is intended for a broad community of computer and software engineering researchers and professionals involved in software engineering component engineering, software architecture research and development of object-oriented software projects. Software engineering researchers, software designers, software architects, software engineers, system engineers and application program developers will greatly benefit from this tutorial.

Required Experience:

Familiarity with basic notions of software engineering, software architecture and UML notation and models.

Presenter's profile:

MOHAMED FAYAD is a J.D. Edwards Professor, Computer Science & Engineering, at the University of Nebraska, Lincoln. He was an associate professor at the computer science and computer engineering faculty at the University of Nevada, from 1995 - 1999. He received an MS and a Ph.D. in computer science, from the University of Minnesota at Minneapolis. He has 15+ years of industrial experience. He has been actively involved in over 60 Object-Oriented projects in several companies. He is a Senior Member of the IEEE, a Senior Member of the IEEE Computer Society, a Member of the ACM, and has served on several conference program committees, including TOOLS USA '96 and Hong Kong QSD '96. In addition he is an IEEE Distinguished Speaker, a Sr. Referee for IEEE Computer, an Associate Editor, Editorial Advisor, and a Columnist for The Communications of the ACM, Al-Ahram Egyptians Newspaper, an Editor-In-Chief for IEEE Computer Society Press (1995-1997). He was a guest editor on six theme issues: CACM's OO Experiences, Oct. 1995, IEEE Computer's Managing OO Software Development Projects, Sept. 1996, CACM's Software Patterns, Oct. 1996, CACM's OO Application Frameworks, Oct. 1997, ACM Computing Surveys - OO Application Frameworks, March 2000, IEEE Software Software Engineering in the Small, Sept. 2000 and is currently working on two more: International Journal Software Practice and Experience - Enterprise Frameworks, April 2001, IEEE Transactions on Robotics and Automation - Object-Oriented Methods for Distributed Control Architecture 2001, and Annals of Software Engineering -- OO Web-Based Software Engineering, 2002. He has published articles in IEEE Software, IEEE Computer, JOOP, ACM Computing Surveys and CACM on OO software engineering methods, experiences, design patterns, and management. He has given tutorials and seminars on OO Technologies at ECOOP '94, TOOLS USA '96, OOPSLA '96, TRI-Ada '96, ECOOP '97, ESEC '97, and OOPSLA '99, among others.

He is the lead author of several books: Transition to OO Software Development, August 1998, Building Application Frameworks, Sept., 1999, Implementing Application Frameworks, Sept., 1999, Domain-Specific Application Frameworks, Oct., 1999, and he is currently working on 4-volume book on Software Architectures, Product-Line Architectures, Component-based Software Development and Enterprise Frameworks, Oct. 2001, published by John Wiley and Sons, Inc.

T22:Patterns for building Component Infrastructures

Presenter	Markus Voelter (MATHEMA AG, Germany)
Duration	Half day
Day	Tuesday - afternoon
Room	1.0.6
Level	Intermediate

Abstract:

Component infrastructures on the server have become wide-spread and well-known over the last couple of years, especially because of EJB and COM+. Also, CORBA will evolved in this direction with CORBA Components. In addition to these standard component infrastructures, there are many proprietary ones in use in many applications throughout the industry. When looking more closely at EJB, CCM or COM+, you will realize that they are based on the same basic patterns. This tutorial presents these basic patterns in the form of a concise pattern language, including examples of how the patterns are actually realized in EJB, CCM or COM+ Understanding these patterns will help you in designing efficient systems based on these infrastructures, they will allow you to compare the differences, and and you will be able to build your own proprietary component infrastructures using these patterns.

Expected Audience:

Enterprise architects and developers. They will be able to design efficient systems based on these infrastructures, compare the differences, and they will be able to build your own proprietary component infrastructures using these patterns.

Required Experience:

The participants should have experience in OO development and they should have some experience with CCM, COM+ or EJB.

Presenter's profile:

Markus Voelter works as a software engineer and consultant for MATHEMA AG, Germany. His interests and experience include distributed systems, components and patterns. Markus has published several papers in this area and he's a regular speaker at the respective conferences.

T23: Concurrency: How Can I get It Work

Presenter	Frank Buschmann (Siemens AG, Gernmany)
Duration	Half day
Day	Tuesday - morning
Room	1.0.4
Level	Intermediate

Abstract:

Motivated by the fact that concurrency, if not used properly and carefully, result in gigantic overhead and thus performance penalties compared to single-threaded systems, this tutorials examines how concurrency can be used effectively. We discuss concurrency issus at all levels of granularity, beginning from fundamental concurrency architectures, over desingning and implementing shared objects, to addressing low-level serialization and synchronization issues. In particular we present concrete patterns and techniques that let applications benefit from using multiple threads. Moreover, for every technique we discuss when it is most feasible, and when it should not be applied. To illustrate the patterns and techniques, we use concrete examples from the real-world. The tutorial concludes with a summary of our experiences in building concurrent systems. The tutorial covers most patterns of our book: Pattern-Oriented Software Architecture, Volume 2, Patterns for Concurrent and Networked Objects.

Expected Audience:

Software-Engineers (professionals, students) who are interested in designing and implementing concurrent systems.

Required Experience:

Reasonable knowledge in Object Technology.

Presenter's profile:

Frank Buschmann is senior principal engineer at Siemens Corporate Technology in Munich, Germany. His interests include Object Technology, Frameworks and Patterns. Frank has been involved in many software development projects. He is leading Siemens' pattern research activities. Frank is co-author of "Pattern-Oriented Software Architecture -- A System of Patterns" and "Pattern-Oriented Software Architecture -- Patterns for Concurrent and Networked Objects".

T24: Built-in Testing for Component-Based Development

Presenters	Hans-Gerhard Gross (Fraunhofer Institute, Germany) Peter Lay (Philips Lab., UK)
Duration	Half day
Day	Tuesday - afternoon
Room	1.0.3
Level	Intermediate

Abstract:

Software systems using Object-Oriented or COTS design methods need a different approach to testing on levels above unit testing. A key aspect of these methodologies is encapsulation, so the internals of the system components are hidden: only the interface and externally observable behaviour are visible. Objects and components often have a complex behaviour, which is dependent on the state information inside them. Some systems are built dynamically, where components can be added to or removed from a running system. Consequently additional mechanisms are needed so that the system can be tested as it is configured while executing, making use of encapsulated state information in the individual components. This tutorial will introduce a new technology that incorporates the advantages of component-based software development with the advantages of built-in test technology. The tutorial has two main parts which reflect the important aspects that the technology is considering: built-in contract testing and built-in Quality of Service testing. The first addresses the issues involved in furnishing individual components with the capabilities needed to check their deployment environment, these are other components as well as platforms, at run-time, while the second addresses the issues involved in furnishing a system of components with the capabilities needed to check that they collectively provide the expected quality of service. The enhanced model of component-based development, incorporating these forms of run-time testing, can be characterized by the phrase "plug, test and play".

Expected Audience:

People involved in development of software systems will learn a new way of analysing requirements on testing, how to fulfil them, and how to build testable software components.

Required Experience:

Attendees are expected to have working knowledge of object-oriented technology and of software verification and validation.

Presenter's profile:

Hans-Gerhard Gross is project manager at the Fraunhofer Institute for Experimental Software Engineering (IESE) in Kaiserslautern, Germany. He is responsible for the development of future software testing technologies within the institute and their transfer into industry. Here, his particular focus and research interest is on the development of verification methods and processes for object-oriented and component-based systems, and on the application of search- and optimisation techniques to software engineering problems. On behalf of the institute, Dr. Gross is also participating in a number of national and international research projects that focus on verification and validation of component-based architectures.

Peter Lay is a senior software architect at the Southampton Systems Laboratory (SLS) of Philips Semiconductors. He has a first class degree in Mathematics from Hull University, and after a brief spell in teaching entered the electronics industry 20 years ago. He has worked on submarine tracking systems, developed software for computer aided design systems and for the last ten years has been developing embedded software for consumer systems. Peter has been a member of various technical committees defining the software architectural strategy for Philips Semiconductors and is currently a member of the team driving the introduction of object-oriented software architectures. In addition he manages the home networking projects in SLS while also consulting to a number of projects in Philips Semiconductors on their software architecture.

Workshops

The workshops of ECOOP, Europe's foremost conference on object technology, are an excellent opportunity to bring together academics and researchers working on similar scientific areas in an atmosphere that fosters interaction, exchange, and problem solving. Workshops also provide the opportunity for representatives of a technical community to coordinate efforts and establish collective plans of action.

The ECOOP 2002 workshop committee have organized 19 high quality proposals for workshops which, as usual, are scheduled for the first two days of the conference. Your participation will contribute to making these two days a success. Enjoy them!

Each workshop lasts one day (except WS04 which lasts two days). Workshops take place on Monday and Tuesday at the Conference venue, from 9:30 to 13:00 and 14:30 to 18:00. Workshop rooms are scattered throughout the building of Informatics (see maps).

MONDAY								
WS02	WS04	WS06	WS07	WS08	WS12	WS18	WS19	WS20

TUESDAY										
WS01	WS03	WS04	WS09	WS10	WS11	WS13	WS14	WS15	WS16	WS17

ID	Title	Mon	Tue	See also
WS01	**Resource Management for Safe Languages** Godmar Back, Walter Binder, Greg Bollella, Laurent Daynes, Michael Hind, Doug Lea, George Necula, Niranjan Suri, Jan Vitek, Greg Czajkowski		room 2.0.4	WS18
WS02	**Generative Programming** Kasper Østerbye, Lutz Röder, Bedir Tekinerdogan, Markus Völter, Krzysztof Czarnecki	room 2.0.4		WS14
WS03	**Sixth Workshop on Pedagogies and Tools for Learning Object-Oriented Concepts** Kim B. Bruce, Jurgen Borstler, Isabel Michiels		room 3.0.3	
WS04	**The 12th Workshop for PhD Students in Object-Oriented Systems** Hermes Senger, Pierre Crescenzo, Michael Haupt, Ezz Hattab, Cyril Ray, Miguel A. Pérez	room 4.1.1		
WS06	**Second International Workshop on Web-Oriented Software Technology, IWWOST 2002** Daniel Schwabe, Luis Olsina, Gustavo Rossi, Oscar Pastor	room 3.0.3		
WS07	**Seventh International Workshop on Component-Oriented Programming (WCOP 2002)** Jan Bosch, Clemens Szyperski, Wolfgang Weck	room 2.0.3		WS10

ID	Title	Mon	Tue	See also
WS08	**Concrete Communication Abstractions of The Next 701 Distributed Object Systems** Andrew Black, Rachid Guerraoui, Eric Jul, Anne-Marie Kermarrec, Doug Lea, Nenad Medvidovic, Salah Sadou, Antoine Beugnard	room 3.0.4		WS10 WS16
WS09	**Unanticipated Software Evolution (USE)** Pascal Costanza, Mikhail Dmitriev, Günter Kniesel		room 4.1.4	
WS10	**Second International Workshop on Composition Languages (WCL 2002)** Bastiaan Schönhage, Jean-Guy Schneider, Thomas Genssler, Markus Lumpe		room 2.0.3	WS07 WS08
WS11	**The Inheritance Workshop** Andrew P. Black, Erik Ernst, Peter Grogono, Markku Sakkinen		room 4.0.1	
WS12	**Model-based Software Reuse** Andreas Speck, Elke Pulvermueller, Ragnhild Van Der Straeten, Ralf Reussner Matthias Clauss	room 2.0.5		WS14
WS13	**6th ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering** Fernando Brito e Abreu, Geert Poels, Houari A. Sahraoui, Mario Piattini		room 3.0.6	
WS14	**Multiparadigm Programming with OO languages** Kei Davis, Yannis Smaragdakis, Jörg Striegnitz		room 3.0.5	WS02 WS12
WS15	**Knowledge-Based Object-Oriented Software Engineering** Kim Mens, Ellen Van Paesschen, Nelson Medinilla, Maja D'Hondt		room 2.0.5	
WS16	**Fifth ECOOP Workshop on Object-Orientation and Operating Systems (ECOOP-OOOSWS'2002)** Olaf Spinczyk, Andreas Gal, Paniti Netinant, Dario Alvarez		room 3.0.4	WS08
WS17	**Integration and Transformation of UML models** Jon Whittle, Ambrosio Toval, Robert France, Joao Araújo		room 2.0.6	
WS18	**The 8th ECOOP Workshop on Mobile Object Systems: New Frontiers** Ciaran Bryce	room 3.0.5		
WS19	**Feyerabend - Redefining Computing** Pascal Costanza, Martine Devos, Dave Thomas, Wolfgang De Meuter	room 4.0.1		
WS20	**Formal Techniques for Java-like Programs (FTfJP)** Susan Eisenbach, Gary T. Leavens, Peter Mueller, Arnd Poetzsch-Heffter, Erik Poll	room 2.0.6		

Workshop Reader

This year we shall once again propose a Workshop Reader, edited by Springer Verlag and containing reports on the workshops, panels and posters. If you have not yet ordered the Workshop Reader at registration time you can still do it during the conference.

WS01: Resource Management for Safe Languages

Organizers	Godmar Back (Univ. Utah/Stanford) Walter Binder (CoCo Software) Greg Bollella (Sun Micros.) Greg Czajkowski (Sun Microsystems) (**co-chair**, **primary contact**) Laurent Daynes (Sun Microsystems) Michael Hind (IBM Research) Doug Lea (SUNY Oswego) George Necula (Berkeley) Niranjan Suri (Univ.of West Florida) Jan Vitek (Purdue University) (**co-chair**)
Day	Tuesday
Room	2.0.4
Home Page	http://www.ovmj.org/workshops/resman/

Abstract:

Safe programming language, such as the Java programming language, offer safety and security features making them attractive for developing extensible environments on a wide variety of platforms, ranging from large servers all the way down to hand-held devices. Extensible environments facilitate dynamic hosting of a variety of potentially untrusted code. This requires mechanisms to guarantee isolation among hosted applications and to control their usage of resources. While certain isolation properties are provided by safe languages, typically resource management is difficult with the current standard APIs and existing virtual machines.

The goal of this one-day workshop is to bring together practitioners and researchers working on various approaches to these problems to share ideas and experience.

The scope of the workshop includes, but is not limited to:

- runtime extensions to control resources
- resource management through code transformations
- tradeoffs between precision and the cost of resource control
- conflicts and synergies between resource management done by the virtual machine and resource management done by the OS
- executing multiple applications in a single virtual machine
- scaling up to large environments
- scaling down to embedded devices
- practical experience

WS02: Generative programming

Organizers	Kasper Østerbye (IT University Copenhagen) Lutz Röder (Microsoft Corporation) Bedir Tekinerdogan (University of Twente) Markus Völter (MATHEMA AG, Germany) Greg Butler (Concordia University) Krzysztof Czarnecki (DaimlerChrysler Research and Technology, Germany) (**primary contact**)
Day	Monday
Room	2.0.4
Home Page	http:// www.generative-programming.org/ecoop2002-workshop.html

Abstract:

Object-oriented technology indisputably provided us with a better handle on complexity than previous technologies. Nevertheless, several issues remain and generative techniques - automated generation of software artifacts - may help us address them. Such issues include the performance and complexity overheads of highly flexible OO designs, and the inability to implement aspectual or even more abstract features such as performance properties in a localized way.

The workshop aims to bring together practitioners, researchers, academics, and students to discuss the state-of-the-art of generative techniques and their role in object-oriented development. The goal is to share experience, assess the state-of-the-art and the state-of-the-practice, consolidate successful techniques, and identify the most promising application areas and open issues for future work. Topics of interest include

- synergy between object-oriented technology, components and generative techniques
- styles of generators (application generators, generators based on XML technologies, template languages (e.g., JSP), template metaprogramming, transformational systems, intentional languages, aspects, subjects, etc), particularly their uses and limitations;
- design of APIs that supports generative techniques
- generation of code artifacts, such as application logic, UIs, database schemas, and middleware integration;
- generation of non-code artifacts such as test cases, documentation, tutorials, and help systems;
- capturing configuration knowledge, for example, in DSLs, and extensible languages;
- influence of generative techniques on software architecture (e.g., building and customizing frameworks and applying patterns);
- testing generic and generative models; and
- industrial applications of generative technology.

Potential participants are asked to submit a two-page (or longer) position paper detailing their experience with generative techniques, their perspective on one or more of the above topics, and their planned contribution to the workshop. We seek concrete case studies, and potential topics of discussion in order to ground the workshop in real-world issues. Participants are expected to read all accepted position papers before the workshop.

WS03: Sixth Workshop on Pedagogies and Tools for Learning Object-Oriented Concepts

Organizers	Isabel Michiels (Vrije Universiteit Brussel) (**primary contact**) Kim B. Bruce (Williams College) Jurgen Borstler (Umea University)
Day	Tuesday
Room	3.0.3
Home Page	http://prog.vub.ac.be/ecoop2002/ws03/

Abstract:

The primary goal of learning and teaching object-oriented concepts is to enable people to successfully participate in an object-oriented development project.

Successfully using object-oriented technology requires a thorough understanding of basic OO concepts. However, learning these techniques, as well as lecturing about these concepts has proven to be very difficult in the past. Misconceptions can occur during the learning cycle and the needed guidance can not always be directly provided.

The goal of this workshop is to share ideas about innovative teaching approaches and tools to improve the teaching and learning of the basic concepts of object technology rather than teaching a specific programming language. Teaching tools can be either tools used in environments or specific environments for learning OO, as well as any kind of support for developing OO learning applications themselves.

WS04: The 12th Workshop for PhD Students in Object-Oriented Systems

Organizers	Hermes Senger (SENAC College of Computer Science and Technology) Pierre Crescenzo (Univ. of Nice-Sophia Antipolis) Michael Haupt (Technical Univ. Darmstadt) Miguel A. Pérez (Univ. Extremadura) (**primary contact**) Ezz Hattab (National Technical University of Athens) Cyril Ray (French Naval Academy Research Institute - IRENav)
Day	Monday and Tuesday
Room	4.1.1
Home Page	http://www.softlab.ece.ntua.gr/facilities/public/AD/phdoos02/index.htm

Abstract:

This workshop is slightly unusual, because the participants are PhD students and the topics are derived from the areas of interest of the participants. The workshop is divided into plenary sessions with a number of pre-screened presentations, and "discussion" sessions, holding small subgroups composed of PhD students working on similar topics. For each participant, this is an opportunity to present his/her research to a knowledgeable audience working in a similar context, and to share his/her ideas on hot-topics or new trends. In this way, the participants may receive insightful comments on their research, learn about related work, and initiate future research collaborations.

The participants are divided into three categories. First, it is possible to submit a (3-8 page) extended abstract on a specific topic, and give a 30 minutes presentation at the workshop. Second, a PhD student may submit a one-page abstract of their PhD project, and give a 15 minutes talk. Finally, last-minute participants may contribute with a short and informal oral presentation of their research.

The program committee is essentially composed of senior and young researchers with a strong background on some area of object-orientation. The review process is not designed to select the few very best papers, but to ensure that every participant is able to present some relevant material, and is well prepared.

WS06: Second International Workshop on Web-Oriented Software Technology, IWWOST 2002

Organizers	Daniel Schwabe (Organizer Chair) (Catholic University in Rio de Janeiro) Luis Olsina (Univ. Nacional de La Pampa, Argentina) Oscar Pastor (Univ. Politécnica de València) (**primary contact**) Gustavo Rossi (Univ. Nacional de La Plata)
Day	Monday
Room	3.0.3
Home Page	http://www.dsic.upv.es/~west/iwwost02

Abstract:

We are entering a new era where dozens of web development application methods are being presented, mostly using object-oriented technology. More than ever it is essential to look for common basic abstractions related to web technology, in order to avoid the "yet another method" syndrome.

The workshop’s intention is to put together research groups worldwide working on methods for web application design and implementation, in order to learn about all methodological approaches, especially those having the object-oriented paradigm as the development kernel. The main goal is to discuss the basic and necessary abstractions and concepts to have a web application conceptual modelling support. Moreover, steps towards a “unified object-oriented web application development framework” to develop web applications would be examined.

WS07: Seventh International Workshop on Component-Oriented Programming (WCOP 2002)

Organizers	Jan Bosch (University of Groningen) Clemens Szyperski (Microsoft) Wolfgang Weck (Oberon microsystems Inc.) (**primary contact**)
Day	Monday
Room	2.0.3
Home Page	http://www.research.microsoft.com/~cszypers/events/wcop2002/

Abstract:

WCOP 2002 seeks position papers on the important field of component-oriented programming (COP). WCOP 2002 is the seventh 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 CORBA/CCM, COM/COM+, JavaBeans/EJB, and most recently .NET. After WCOP'96 focused on the fundamental terminology of COP, the subsequent workshops expanded into the many related facets of component software.

WCOP 2002 shall emphasize dynamic reconfiguration of component systems, that is, the overlap between COP and dynamic architectures. Also, submissions reporting on experience with component-oriented software systems in practice are strongly encouraged, where the emphasis is on interesting lessons learned, whether the actual project was a success or a failure.

To enable lively and productive discussions, the workshop will be limited to 25 participants. Depending on the submitted position papers, the workshop will be organized into three or four subsequent mini-sessions, each initiated by a presentation of two or three selected positions and followed by discussions. Instead of splitting the workshop into task forces, it is intended to provoke lively discussion by preparing lists of critical questions and some, perhaps provocative, statements (to be used on demand).

Position papers will be formally reviewed, each by at least two independent reviewers. As an incentive for submission of high quality statements, the best position statements will be combined with transcripts of workshop results and published.

WS08: Concrete Communication Abstractions of the next 701 Distributed Object Systems

Organizers	Antoine Beugnard (ENST-Bretagne, France) (**primary contact**) Nenad Medvidovic (Univ. of Southern California) Eric Jul (University of Copenhagen) Andrew Black (OGI School of Science & Engineering at OHSU) Rachid Guerraoui (Ecole Polytechnique Federale de Lausanne) Anne-Marie Kermarrec (Microsoft Research) Doug Lea (State University of New York at Oswego) Salah Sadou (Universiete de Bretagne Sud)
Day	Monday
Room	3.0.4
Home Page	http://perso-info.enst-bretagne.fr/~beugnard/ecoop/ws8-2002.html

Abstract:

As applications become increasingly distributed and networks provide more and more connection facilities, applications require more and more interconnections, thus communication takes a central part of modern systems. To tackle the communication issue a lot of techniques and concepts have been developed in different research fields and some industrial solutions have been proposed. Over the last 15 years, the basic building blocks for distributed object systems have emerged: distributed objects, communicating with Remote Message Send (RMS), also known as Remote Method Invocation (RMI) or Location-Independent Invocation (LII). However, it has also become clear that while such abstractions are by themselves sufficient to expose the hard problems of distributed computing, they do not solve them.

Hence, since large applications parts have been underlined like databases systems or graphical user interface, the goal is to wonder, if can we say the same for the communication part of applications?

At last year's ECOOP workshop on The Net 700 Distributed Object Systems, we identified some of these problems (Security, Partial Failure, Guaranteeing Quality of Service, Run-time evolution, Meta-Object protocols, and Ordering of events) that are important concerns of any communication abstraction. The goal of this workshop is to work on the definition of new and good communication abstractions and on the distributed-specific features mentioned above.

WS09: Unanticipated Software Evolution (USE)

Organizers	Günter Kniesel (University of Bonn) (**primary contact**) Pascal Costanza (University of Bonn) Mikhail Dmitriev (Sun Microsystems)
Day	Tuesday
Room	4.1.4
Home Page	http://www.cs.uni-bonn.de/~gk/use2002/

Abstract:

Many studies of complex software systems have shown that more than 80% of the total cost of software development is devoted to software maintenance. This is mainly due to the need for software systems to evolve in the face of changing requirements. In some cases, software evolution may need to be dynamic, with changes being performed on running systems.

Despite the importance of software evolution, techniques and technologies that offer support for software evolution are far from ideal. In particular, unanticipated requirement changes are not well supported, although they account for most of the technical complications and related costs of evolving software.

By definition, unanticipated software evolution (USE) is not something for which we can prepare during the design of a software system. Therefore, support for such evolution in programming languages, and component models and related runtime infrastructures becomes a key issue. Without it, unanticipated changes often force software engineers to perform extensive invasive modification of existing designs and code.

This one-day workshop will address the issues inherent in static and dynamic evolution of object-oriented and component based systems. The goal of the workshop is to discuss new approaches and technologies for building large-scale evolvable software systems.

WS10: Second International Workshop on Composition Languages (WCL 2002)

Organizers	Markus Lumpe (Iowa State Univ) (**primary contact**) Bastiaan Schönhage (OTI, Netherlands) Thomas Genssler (FZI, University Karlsruhe) Jean-Guy Schneider (Swinburne Univ. of Technology)
Day	Tuesday
Room	2.0.3
Home Page	http://www.cs.iastate.edu/~lumpe/WCL2002

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 the first workshop on composition languages (WCL 2001), which was held in conjunction with ESEC/FSE 2001.

The main focus of the workshop will be on language issues for composing components into applications, 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 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.

All submissions will be peer-reviewed by at least two members of the workshop paper selection committee. Based on the quality and originality, a selection of the best position statements will be presented at the workshop. The workshop will be organized in several sessions, each dedicated to a

particular subject of common interest. Instead of splitting the workshop into task forces, we intend to provoke lively discussion by preparing lists of critical questions and topics, which will be published and distributed to the participants before the workshop.

The main expected results of the workshop would be an outline of collaborative research topics and a list of areas for further exploration. Additionally, the results collected during the workshop will be presented to the rest of the ECOOP community in the form of a poster at the same conference.

WS11: The Inheritance Workshop

Organizers	Markku Sakkinen (Univ. of Jyväskyla) (**primary contact**) Andrew P. Black (Oregon Health & Science University) Erik Ernst (Univ. of Aalborg) Peter Grogono (Concordia University)
Day	Tuesday
Room	4.0.1
Home Page	http://www.cs.auc.dk/~eernst/inhws/

Abstract:

Inheritance has been a cornerstone of Object-Oriented programming at least since Simula 67. But it is complex and error-prone: witness the "fragile base class" problem, the con ict between inheritance and encapsulation, and the confusion between inheritance and subtyping. This activity demonstrates that inheritance is both hard to avoid, and hard to get right.

The purpose of this workshop is to open a dialog about the functions and goals that inheritance serves, and how inheritance might best be constrained, changed or replaced to meet these goals. We seek evaluations of and experience reports on inheritance and its known alternatives, and descriptions of languages using alternative mechanisms. Possible subtopics include:

- Flaws and anomalies in the way inheritance is currently implemented or used. Examples that are hard to express.
- Inheritance mechanisms: single, multiple, mixin based, composing, rst-class or second-class, aspect-oriented.
- Interaction between inheritance and other language features: inheritance and parameterization, inheritance and typing, inheritance and encapsulation, inheritance and behaviour.
- Inheritance and software maintenance: inheritance at analysis, design or implementation time; inheritance and program understanding; refactoring tools and inheritance.
- Conceptual views of inheritance: inheritance a conceptual modeling tool, as a dependency management tool, and as a code reuse mechanism.

WS12: Model-based Software Reuse

Organizers	Andreas Speck (Intershop Research) Germany Elke Pulvermueller (Universitaet Karlsruhe) Matthias Clauss (Intershop Research) (**primary contact**) Ragnhild Van Der Straeten (Vrije Universiteit Brussel) Ralf Reussner (Monash University)
Day	Monday
Room	2.0.5
Home Page	http://research.intershop.com/workshops/ECOOP2002

Abstract:

The demand for software reuse is rather old and different approaches to support software reuse have been developed: patterns, modules, frameworks and component (architectures). Aspect-orientation deals with cross-cutting concerns and generative software engineering supports automatic system generation out of basic building blocks.

One problem, common to all of the aforementioned approaches, is how to model the different reusable assets (such as components, features or aspects) and the composition of assets. This problem partially occurs when dealing with, e.g., component interoperability, aspect weaving, feature interaction and (on a more abstract level) traceability between different views or models. Conventional object-oriented modelling languages (like the UML) are used for communicating users requirements and system designs, but fail to support the composition of separate parts or views.

One approach to deal with the reuse and related composition problem is to use models allowing to (a) check their interoperability, (b) support the configuration of assets, and (c) to predict properties of the assembled system (especially compliance with user requirements).

The workshop concentrates on this model-based approach to software reuse. As a result of the workshop the organisers like to see a common understanding on which properties such asset-models must have to ensure reuse and composition.

WS13: 6th ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QAOOSE 2002)

Organizers	Mario Piattini (Universidad de Castilla-La Mancha) (**primary contact**) Fernando Brito e Abreu (Universidade Nova de Lisboa) Geert Poels (Katholieke Universiteit Leuven) Houari A. Sahraoui (Université de Montréal)
Day	Tuesday
Room	3.0.6
Home Page	http://alarcos.inf-cr.uclm.es/qaoose2002

Abstract:

The proposed workshop is a direct continuation of five successful workshops, held at previous editions of ECOOP in Budapest (2001), Cannes (2000), Lisbon (1999), Brussels (1998) and Aarhus (1995). The

QAOOSE series of workshops has attracted participants from both academia and industry that are involved / interested in the application of quantitative methods in object oriented software engineering research and practice. Quantitative approaches in the OO field is a broad but active research area that aims at the development and/or evaluation of methods, techniques, tools and practical guidelines to improve the quality of software products and the efficiency and effectiveness of software processes. The relevant research topics are diverse, but always include a strong focus on applying a scientific methodology based on data collection (either by objective measurements or subjective assessments) and analysis (by statistical or artificial intelligence techniques). Like in previous years, submissions of position papers are invited, but not limited, to the areas of metrics collection, quality assessment, metrics validation, and process management.

The 2002 edition of the workshop aims to shed some light on recent research results and to point out future research directions that might interest not only the academic community but also industry. Over the years the QAOOSE workshop series has built an active research community working towards special topics of interest that were identified as open issues during previous workshop editions. This year we explicitly solicit contributions of new research on

- automatic support to share research hypotheses, data, and results
- measures for non-functional requirements
- measures for component models
- meta-metrics
- the application of quantitative methods in industrial software processes
- OO project estimation models

The workshop series will result in a forthcoming Addison-Wesley book (to appear in 2002) and a special issue of the journal L'Objet on quantitative approaches in OO software design. Also for the 2002 edition of the workshop we are contacting an academic journal to publish a selection of thoroughly revised position papers in a special issue devoted to the workshop.

During the workshop there will be some sessions for presenting position papers and a plenary 'working' session for summarizing, evaluating and assembling the new research results and for identifying future research opportunities. Participation in the workshop is by invitation only, i.e. based on the submission of a position paper. All submitted position papers will be formally reviewed by the workshop organizers for originality, relevance, quality and clarity. We especially encourage the submission of new ideas, even if not supported by validated research. Like in last year's workshop, all invited participants will be asked to summarize the contributions and limitations of the research presented.

WS14: Multiparadigm Programming with OO Languages

Organizers	Jörg Striegnitz (Central Institute for Applied Mathematics) (**primary contact**) Kei Davis (Los Alamos National Laboratory) Yannis Smaragdakis (Georgia Institute of Technology)
Day	Tuesday
Room	3.0.5
Home Page	http://www.multiparadigm.org/mpool2002

Abstract:

While OO has become ubiquitously employed for design, implementation and even conceptualization, many practitioners recognize the concomitant need for other programming paradigms according to

problem domain. Nevertheless, the choice of a programming paradigm is strongly in uenced by the supporting programming language facilities. In turn, choice of programming language is usually a practical matter; one cannot generally afford to use a language not in the mainstream. We seek answers to the question of how to address the need for other programming paradigms in the general context of OO languages.

Can OO programming languages effectively support other programming paradigms? The tentative answer seems to be affirmative, at least for some paradigms; for example, significant progress has been made for the case of (higher order, polymorphic) functional programming in C++.

This workshop seeks to bring together practitioners and researchers in this emerging field to 'compare notes' on their work–describe existing, developing, or proposed techniques, idioms, methodologies, language extensions, or software for expressing non-OO paradigms in OO languages; or theoretical work supporting or defining the same. Work-in-progress reports are welcomed.

WS15: Knowledge-Based Object-Oriented Software Engineering (KOSE)

Organizers	Maja D'Hondt (Vrije Universieit Brussel (**primary contact**) Kim Mens (Université de Louvain-la-Neuve) Ellen Van Paesschen (Vrije Universiteit Brussel Nelson Medinilla (Univ. Politécnica de Madrid)
Day	Tuesday
Room	2.0.5
Home Page	http://infoweb.vub.ac.be/~mjdhondt/KBOOSE/

Abstract:

The complexity of software domains – such as the financial industry, television and radio broadcasting, hospital management and rental business – is steadily increasing and knowledge management of businesses is becoming more important with the demand for capturing business processes. On the other hand the volatility of software development expertise needs to be reduced. These are symptoms of a very significant tendency towards making knowledge of different kinds explicit: knowledge about the domain or the business, knowledge about developing software, and even meta-knowledge about these kinds of knowledge.

Examples of approaches that are directly related to this tendency or could contribute to it are knowledge engineering, ontologies, conceptual modeling, domain analysis and domain engineering, business rules, work ow management and research presented at conferences and journals on Software Engineering and Knowledge Engineering and Automated Software Engineering, formerly known as Knowledge-Based Software Engineering. Whereas this community already contributed for years to research in knowledge engineering applied to software engineering but also vice versa, this workshop intends to focus on approaches for using explicit knowledge in various ways and in any of the tasks involved in object-oriented programming and software engineering. Another goal of this workshop is to bridge the gap between the aforementioned community and the ECOOP community.

On the one hand, this workshop is a platform for researchers interested in the symbiosis of knowledge-based or related methods and technologies with object-oriented programming or software development.

On the other hand it welcomes practitioners confronted with the problems in developing knowledge-intensive software and their approach to tackling them.

WS16: Fifth ECOOP Workshop on Object-Orientation and Operating Systems

Organizers	Dario Alvarez (Universidad de Oviedo) (**primary contact**) Olaf Spinczyk (University of Magdeburg) Andreas Gal (University of California, Irvine) Paniti Netinant (Bangkok University)
Day	Tuesday
Room	3.0.4
Home Page	http://ooosws.cs.uni-magdeburg.de/

Abstract:

The first and the second ECOOP Workshops on Object-Orientation and Operating Systems were held at ECOOP'97 in Jyväskyla and at ECOOP'99 in Lisbon. They successfully provided a forum for lively and interesting discussions. In the Lisbon workshop, the participants proposed to repeat the workshop yearly, leading to a third workshop that took place at ECOOP'2000 in Cannes, and the ECOOP'2001 in Budapest. Over the past years the number of participants at the ECOOP OOOS workshop grew constantly and many high quality papers have been presented. Several of the papers submitted to ECOOP OOOS workshops have been later published as part of Communications of the ACM and other relevant journals.

The ECOOP workshop series aims to bring together researchers and developers working on object-oriented operating systems and to provide a platform for discussing problems arising from the application of object-orientation to operating systems and solutions for them. Suggested topics for papers and discussion include, but are not restricted to:

- adaptable and adaptive OOOS, frameworks for OOOS, architecture of OOOS
- distributed OOOS and middleware, aspect orientation and OOOS design
- what are the penalties of OO in OS and how to avoid them, reflective OOOS
- OOOS tools, reusability and interoperability of OOOS components
- OOOS configurability, maintenance, tuning and optimization
- OOOS for embedded systems, real-time OOOS, and Java-based systems.

Applicants for the workshop are asked to submit a position paper of 6 pages maximum. Submitted papers will be reviewed. Papers clearly not within the scope of the workshop will have to be rejected. In contrast to the last years we intend to name a formal program committee consisting of the workshop organizers and additional external members.

WS17: Workshop on Integration and Transformation of UML models

Organizers	Joao Araújo (Universidade Nova de Lisboa) (**primary contact**) Jon Whittle (QSS Inc. / NASA Ames Research Center) Ambrosio Toval (Universidad de Murcia) Robert France (Colorado State University)
Day	Tuesday
Room	2.0.6
Home Page	http://ctp.di.fct.unl.pt/~ja/wituml02.htm

Abstract:

This workshop aims to understand more fully the relationship between different UML models. Each model in UML represents a particular viewpoint of the system under development. However, it is not clear from the UML specification how these models should be related. The relationships between UML models can be understood in terms of transformation – in which models are translated into other models (e.g., sequence diagrams transformed into statecharts) – or integration – in which models are somehow linked to contribute additional meaning (e.g., a statechart is attached to a class to describe the behaviour of that class). The result of the workshop will be an increased understanding of the possible transformation/integration approaches and how they can best be used in particular contexts.

This is a follow-on workshop to the highly successful Workshop on Transformations of UML models (WTUML), held as a satellite event to the ETAPS conference (Genoa, April 2001).

A non-exhaustive list of topics includes:

- transformation and integration between UML notations;
- transformations to other notations (e.g., formal languages, other modelling languages, code);
- rigorous processes for transformation and integration;
- formal and informal transformations;
- experience in industry using transformation and integration in UML;
- identification of specific transformations needed in industry;
- semantic issues related to UML model transformation and integration;
- types of transformations (e.g. refinements, enhancements, refactoring);
- role of transformations and integrations in UML;
- transformations for development, validation, verification;
- frameworks for implementing transformations;
- tool support for transformations and integration.

WS18: The 8th ECOOP WORKSHOP ON MOBILE OBJECT SYSTEMS: New Frontiers

Organizers	Ciaran Bryce (University of Geneva)
Day	Monday
Room	3.0.5
Home Page	http://cui.unige.ch/~ecoopws/ws02/

Abstract:

Mobile agent and object technology has been the subject of much research and development over the past few years. The object-oriented community has been to the forefront of this field, with language and system support, e.g., Emerald and Java.

Despite this success, hard questions remain, notably with respect to security, efficient operating system support and mobile object application design and programming techniques. In the other hand, the appearance of new networking and system infrastructures like peer-to-peer computing and wireless networks is posing problems for system designers that perhaps mobile objects might best solve.

The ECOOP Workshop On Mobile Object Systems was first held in 1995, and has been held every year since. Its goal has been to bring together researchers and practitioners working on mobility. Though the primary aim of the workshops has been a discussion forum, two books have come from their proceedings: Mobile Object Systems (LNCS 1222) and Secure Internet Programming (LNCS 1603).

WS19: Feyerabend - Redefining Computing

Organizers	Wolfgang De Meuter (Vrije Universiteit Brussel) (**primary contact**) Pascal Constanza (University of Bonn) Martine Devos (personally) Dave Thomas (Bedarra Corp, Carleton University and University of Queensland)
Day	Monday
Room	4.0.1
Home Page	http://www.dreamsongs.com/Feyerabend/ECOOP02/

Abstract:

Fifty years into the First Computing Era some of us in the computing arena have come to realize we have made a false start that cannot be fixed, and for us to finally be able to produce lasting, correct, beautiful, usable, scalable, enjoyable software that stands the tests of time and moral human endeaveor, we need to start over. Perhaps we will be able to salvage some of what we have learned from the First Era, but we expect almost everything except the most mathematical and philosophical fundamentals to be brushed aside.

This workshop is one in a series (see www.dreamsongs.com for a complete list of previous workshops) leading up to an event to reinvent computing. For that event, a most diverse group of 75 people will be put together. The result of the two-week event will be the first step toward a roadmap for massive rebuilding of computing both as a theoretical endeavor as a practice and toward a plan to accomplish it.

Just like the OOPSLA2001 Feyerabend workshop, this workshop will focus on the very basics of OO since the basic philosophy behind OO might definitely serve as a fertile soil of thoughts to be injected into the Feyerabend project.

WS20: Formal Techniques for Java-like Programs (FTfJP)

Organizers	Erik Poll (Univ. of Nijmegen) (**primary contact**) Susan Eisenbach (Imperial College) Gary T. Leavens (Iowa State University) Peter Mueller (Deutsche Bank) Arnd Poetzsch-Heffter (FernUniversitaet Hagen)
Day	Monday
Room	2.0.6
Home Page	http://www.cs.kun.nl/~erikpoll/ftfjp/

Abstract:

At ECOOP'99, ECOOP'2000, and ECOOP'2001, workshops have been organized under the title "Formal Techniques for Java Programs". These all attracted in the order of 30 participants, and many asked us to arrange a fourth event on this topic.

The topic is formal techniques to analyze programs, to precisely describe program behavior, and to verify program properties. Whereas the earlier workshops were restricted to Java, we have chosen to widen the scope to include closely related languages such as C#.

Languages like Java and C# provide a good platform to bridge the gap between formal techniques and practical program development, because of their reasonably clear semantics and standardized libraries. Moreover, these languages are an interesting target for formal techniques, because the novel paradigm for program deployment introduced with Java, with its improved portability and mobility, opens up new possibilities for abuse and causes concern about security.

Possible topics include:

- specification techniques and interface specification languages
- specification of software components and library packages
- automated checking and verification of program properties
- verification logics
- language semantics
- type systems
- dynamic linking and loading
- security issues

Technical Programme

The main conference runs from Wednesday 12 to Friday 14, and hosts the technical program, consisting of 24 high-quality technical papers, two invited talks given by José Meseguer and Clemens Szyperski, and one panel ("Agility Fragility or Process Duress: Which is Worse?") chaired by Ole Lehrmann Madsen. In this occasion the technical programme features two attractive birds-of-a-feather (BoF) meetings on Thursday afternoon. Demonstrations, posters, and exhibits will happen in parallel with the technical programme. We are also proud of having Prof. Krysten Nygaard deliver the keynote speech at the conference banquet, one of the recipients of this year's IEEE John von Neumann Medal and ACM Turing Award.

Technical Sessions

Wednesday, June 12

9:30 – 10:00 **WELCOME**

10:00 – 11:00 **INVITED TALK 1**

Semantic Models for Distributed Object Reflection

J. Meseguer, University of Illinois at Urbana-Champaign

11:00 – 11:30 *Coffee break*

11:30 – 13:00 **SESSION 1: Aspect Oriented Software Development**

AOP: Does it Make Sense? The Case of Concurrency and Failures

Jörg Kienzle, Swiss Federal Institute of Technology Lausanne
Rachid Guerraoui, Swiss Federal Institute of Technology Lausanne

Difference-Based Modules: A Class-Independent Module Mechanism

Yuuji Ichisugi, National Inst. of Advanced Industrial Science and Techn.
Akira Tanaka, National Inst. of Advanced Industrial Science and Techn.

Dynamically Composable Collaborations with Delegation Layers

Klaus Ostermann, Siemens AG Corporate Technology

13:00 – 14:30 *Lunch*

14:30 – 16:00 **SESSION 2: Java Virtual Machines**

Space- and Time-Efficient Implementation of the Java Object Model

David Bacon, IBM T.J. Watson Research Center
Stephen Fink, IBM T.J. Watson Research Center
David Grove, IBM T.J. Watson Research Center

Atomic Instructions in Java

David Hovemeyer, University of Maryland
William Pugh, University of Maryland
Jaime Spacco, University of Maryland

Code Sharing Among Virtual Machines

Grzegorz Czajkowski, Sun Microsystems
Laurent Daynes, Sun Microsystems
Nathaniel Nystrom, Sun Microsystems/Cornell University

16:00 – 16:30 *Coffee break*

16:30 – 18:00 **SESSION 3: Miscellaneous**

J-Orchestra: Automatic Java Application Partitioning

Eli Tilevich, Georgia Tech
Yannis Smaragdakis, Georgia Tech

Supporting Unanticipated Dynamic Adaptation of Application Behaviour

Barry Redmond, Trinity College Dublin
Vinny Cahill, Trinity College Dublin

A Simple and Practical Approach to Unit Testing: The JML and JUnit Way

Yoonsik Cheon, Iowa State University
Gary T. Leavens, Iowa State University

Thursday, June 13

9:30 – 11:00 **PANEL**

Agility Fragility or Process Duress: Which Is Worse?
Chaired by Ole Lehrmann Madsen, Aarhus University, Denmark

11:00 – 11:30 *Coffee break*

11:30 – 13:00 **SESSION 4: Distributed Systems**

Modular Internet Programming with Cells

Ran Rinat, Johns Hopkins University
Scott Smith, Johns Hopkins University

Lana: An Approach to Programming Autonomous Systems

Ciaran Bryce, University of Geneva
Chrislain Razafimahefa, University of Geneva
Michel Pawlak, University of Geneva

Engineering Event-Based Systems with Scopes

Ludger Fiege, TU Darmstadt
Mira Mezini, TU Darmstadt
Gero Muehl, TU Darmstadt
Alejandro Buchmann, TU Darmstadt

13:00 – 14:30 *Lunch*

14:30 – 16:00 **SESSION 5: Patterns and Architecture**

Architectural Reasoning in ArchJava

Jonathan Aldrich, University of Washington
Craig Chambers, University of Washington
David Notkin, University of Washington

Patterns as Signs

James Noble, Victoria University of Wellington
Robert Biddle, Victoria University of Wellington

Pattern Based Design and Implementation of a XML and RDF

Parser and Interpreter: A Case Study

Gustaf Neumann, Vienna University of Economics and BA
Uwe Zdun, University of Essen

16:00 – 16:30 *Coffee break*

16:30 – 18:00 **SESSION 6: Languages**

Modern Concurrency Abstractions for C#

Nick Benton, Microsoft Research
Luca Cardelli, Microsoft Research
Cedric Fournet, Microsoft Research

On Variance-Based Subtyping for Parametric Types

Mirko Viroli, Universita' degli Studi di Bologna
Atsushi Igarashi, University of Tokyo

Type-Safe Prototype-based Component Evolution

Matthias Zenger, Swiss Federal Institute of Technology, Lausanne

18:00 – **BOFs**

- The code-in and refactoring party
- The eclipse BOF

21:30 – **CONFERENCE BANQUET**

Keynote Speech (K. Nygaard)

Friday, June 14

10:00 – 11:00 **INVITED TALK 2**

Objectively: Components versus Web Services

Clemens Szyperski, Microsoft Research

11:00 – 11:30 *Coffee break*

11:30 – 13:00 **SESSION 7: Optimization**

Thin Guards: A Simple and Effective Technique for Reducing the Penalty of Dynamic Class Loading

Matthew Arnold, Rutgers University/IBM T.J. Watson Research Center
Barbara Ryder, Rutgers University

Type-Safe Method Inlining

Neal Glew, California
Jens Palsberg, Purdue University

Polychotomic Encoding: A Better Quasi-Optimal Bit-Vector Encoding of Tree Hierarchies

Robert E. Filman, Research Institute for Advanced Computer Science

13:00 – 14:30 *Lunch*

14:30 – 16:00 **SESSION 8: Theory and Formal Techniques**

A Formal Framework for Java Separate Compilation

Davide Ancona, University of Genova

Giovanni Lagorio, University of Genova
Elena Zucca, University of Genova

Behavioral Compatibility of Self-typed Theories

Suad Alagic, University of Southern Maine
Svetlana Kouznetsova, Wichita State University

Semantics-Based Composition of Class Hierarchies

Gregor Snelting, Universitity of Passau
Frank Tip, IBM T.J. Watson Research Center

16:00 – 16:30 **CONFERENCE FAREWELL**

Invited Talks

Invited Talk I: *Semantic Models for Distributed Object Reflection*

Presenter: **José Meseguer**
University of Illinois at Urbana-Champaign

Abstract:

A generic formal model of distributed object reflection is proposed, that combines logical reflection with a structuring of distributed objects as nested configurations of metaobject that can control subobjects under them. The model provides mathematical models for a good number of existing models of distributed reflection and of reflective middleware. To illustrate the ideas, we show in some detail how two important models of distributed actor reflection can be naturally obtained as special cases of our generic model, and discuss how several recent models of reflective middleware can be likewise formalized as instances of our model.

[co-authored with Carolyn Talcott].

Brief biography:

Jose Meseguer received his doctorate in Mathematics from the University of Zaragoza, Spain, in 1975. He is currently Professor of Computer Science at the University of Illinois at Urbana-Champaign. Formerly he was a Principal Scientist at SRI International in Menlo Park, California, and held postdoctoral positions at UC Berkeley, UCLA, and the University of Santiago, Spain. Dr. Meseguer's research interests include: (1) Formal Executable Specification, Programming, and Verification; (2) Concurrent, Distributed, and Mobile Computing; and (3) Logical and Semantic Foundations of Computing. He is particularly well-known for his work on algebraic specification languages, semantic models of distributed object systems, rewriting logic, concurrency theory, general logics, and on the OBJ and Maude languages.

Invited Talk II: *Objectively: Components versus Web Services*

Presenter: **Clemens Szyperski**
Microsoft Research

Abstract:

We are observing a dramatic confluence of several different aspects: software components, software as a service, and an ever growing space of Internet and Web standards. Over the past year all major players in the software industry have announced their support of XML Web Services in one form or another. So, are services here to displace components? And what about our good old objects? Drawing boundaries that help to understand the key concepts without obstructing the path towards future development is important but challenging. Concepts such as contracts, specifications, and perhaps even the very notion of correctness need to be rethought. Or are they? A strange feeling of deja vue spreads as we see computer science and software engineering rediscovered - this time at your service.

Brief biography:

After years of both academic and entrepreneurial experience, Clemens Szyperski has joined Microsoft Research in Redmond, Washington in early 1999, where he works on furthering the principles, technologies, and methods supporting component software. He is the author of the award-winning book "Component Software: Beyond Object-Oriented programming" (Addison Wesley), now in its second edition, and of numerous other publications. He has served on program committees for major international conferences, including ECOOP, ICSE, and OOPSLA and he is a frequent speaker at events of both academic and industrial nature.

Clemens received his Masters in Electrical Engineering in 1987 from the Aachen Institute of Technology, in Germany. He received his PhD in Computer Science in 1992 from ETH Zurich under the guidance of Niklaus Wirth. After a postdoctoral fellowship at the International Computer Science Institute at UC Berkeley, he was tenured as associate professor at the Queensland University of Technology, Australia, where he continues to hold an adjunct professorship. He is a cofounder of Oberon Microsystems, Inc., Zurich, with its recent spinoff, esmertec inc, also Zurich.

Keynote Speaker

Presenter: Kristen Nygaard

Prof Kristen Nygaard will deliver this year's keynote speech in the banquet. Recently, both the IEEE John von Neumann Medal and ACM Turing Award have been awarded to Ole-Johan Dahl and Kristen Nygaard for the pioneering work they did in the 60's when designing the programming language SIMULA67:

19 November 2001 – IEEE: Ole-Johan Dahl and Kristen Nygaard have been awarded the IEEE's 2002 John von Neumann Medal "For the introduction of the concepts underlying object-oriented programming through the design and implementation of SIMULA67."

New York, February 6, 2002 – ACM has presented the 2001 A.M. Turing Award, considered the "Nobel Prize of Computing", to Ole-Johan Dahl and Kristen Nygaard of Norway for their role in the invention of object-oriented programming, the most widely used programming model today. Their work has led to a fundamental change in how software systems are designed and programmed, resulting in reusable, reliable, scalable applications that have streamlined the process of writing software code and facilitated software programming.

2002 will thus be remembered as an exceptional year in the history of Object-Orientation and it is an honour to have Kristen Nygaard as this year's banquet speaker.

Panel

Agility Fragility or Process Duress: Which Is Worse?

Moderator: Ole Lehrmann Madsen, Aarhus University, Denmark

Room: Conference theater

Demonstrations

Live demonstrations offer an excellent occasion for discussing technical aspects of object-oriented applications, tools and systems. Technical members of their implementation team give these demonstrations twice on Wednesday 12, Thursday 13 and Friday 14 June, in parallel with the technical programme.

	Wed 12/6		Thu 13/6		Fri 14/6	
	Coffee break					
	Room A	Room B	Room A	Room B	Room A	Room B
11:30	D04	D10	D02	D06	D01	D11
12:15		D05		D03	D02	
	Lunch					
14:30	D07	D06	D07	D11	D04	D09
15:15	D08	D13	D08	D13	D05	
	Coffee break					
16:30	D01	D10	D09	D12		
17:15		D12		D03		

ID	Title	Contact Person	Schedule
D01	**Developing Tooling with Eclipse for Embedded Java Development**	Aldo Eisma and Susan Iwai (Object Technology International AG)	Wed 12, 16:30 Fri 14, 11:30
D02	**Experiences in Developing an Aspect-Oriented Development Tool in AspectJ and Eclipse**	Adrian M. Colyer and Susan Iwai (Object Technology International AG)	Thu 13, 11:30 Fri 14, 12:15
D03	**Separating Computation from Coordination for Java Applications**	Joao Gouveia, Georgios Koutsoukos, Michel Wermelinger, Luis Andrade and José Luiz Fiadeiro (Oblog Software SA)	Thu 13, 12:15 Thu 13, 17:15
D04	**Squeak: the Open-Source Smalltalk**	Stephane Ducasse (SCG-IAM, University of Berne)	Wed 12, 11:30 Fri 14, 14:30
D05	**SOUL: Supporting Object-Oriented Software Development with Logic Meta Programming**	Tom Tourwe, Johan Brichau (Universiteit Brussel) and Kim Mens (Université catholique de Louvain)	Wed 12, 12:15 Fri 14, 15:15
D06	**Typed Concurrent Objects**	Vasco T. Vasconcelos (University of Lisboa) and Luis Lopes (University of Porto)	Wed 12, 14:30 Thu 13, 11:30
D07	**LAVA – Object-based Dynamic Inheritance for Java**	Günter Kniesel, Uwe Bardey,Tobias Windeln and Jörg Gonska (University of Bonn)	Wed 12, 14:30 Thu 13, 14:30
D08	**A Threaded Code Interpreter for Rotor, Microsoft's Shared Source Implementation of the Common Language Infrastructure (CLI)**	Yahya H. Mirza (Aurora Borealis Software)	Wed 12, 15:15 Thu 13, 15:15

ID	Title	Contact Person	Schedule
D09	**GME: A Reflective Environment for Domain-Specific Modeling**	Akos Ledeczi (Institute for Software Integrated Systems Vanderbilt University)	Thu 13, 16:30 Fri 14, 14:30
D10	**Using the Montages Language Prototyping Tool to add Object-Orientedness to Xasm**	Philipp W. Kutter (Applied Formal Methods Institute AG and ETH Zurich)	Wed 12, 11:30 Wed 12, 16:30
D11	**SoftwarePot: Encapsulated File Spaces for Secure Software Circulation**	Yoshihiro Oyama, Kazuhiko Kato (Japan Science and Technology Corporation) and Katsunori Kanda (University of Tsukuba)	Thu 13, 14:30 Fri 14, 11:30
D12	**LES Objects: A Model-based Code Generation Environment for Object-Oriented Conceptual Modeling of Web Application Interfaces**	Jaime Gómez, Cristina Cachero and Antonio Párraga (Web Engineering Group, University of Alicante)	Wed 12, 17:15 Thu 13, 16:30
D13	**Octopus: A PostWIMP Framework for New Interaction Techniques**	Mads Brøgger Enevoldsen, and Michael Lassen (Computer Science Dept., University of Aarhus)	Wed 12, 15:15 Thu 13, 15:15

D01: Developing Tooling with Eclipse for Embedded Java Development

Presenters: Aldo Eisma and Susan Iwai, Object Technology International AG

Eclipse is an open source, extensible, tooling platform (see http://www.eclipse.org/). The platform was built using its own Java IDE hat includes Java development support features such as refactoring, version control, compiling and debugging. The Eclipse platform is designed to be easily extended by plugins to create tools for other languages and applications.

For the Eclipse-based IBM WebSphere Device Developer product we developed a set of plugins to support building and launching of embedded applications (with support for various platforms, such as J2ME/MIDP, PocketPC and PalmOS). This demonstration presents the results of our work as an illustration of the possibilities that the Eclipse platform gives to tool developers. The demonstration wil focus on the integration of the SmartLinker(tm) Java optimization and packaging tool in Eclipse. The presentation will also show how Eclipse integration facilitates collaboration among distributed labs when we illustrate how we set up and run experiments collaboratively between OTI's labs and our research partners at the IBM TJ Watson Research Center in New York.

Schedule:

- Wednesday, June 12, 16:30, room A
- Friday, June 14, 11:30, room A

D02: Experiences in Developing an Aspect-Oriented Development Tool in AspectJ and Eclipse

Presenters: Adrian M. Colyer and Susan Iwai, Object Technology International AG

Eclipse is an open source, extensible, tooling platform (see http://www.eclipse.org/). The platform was built using its own Java IDE that includes Java development support features such as refactoring, version control, compiling and debugging. The Eclipse platform is designed to be easily extended by plugins to create tools for other languages and applications.

The AspectJ development tools plug-in provides support for aspect-oriented software development in the Eclipse environment using the AspectJ programming language. AspectJ is an extension to the Java programming language that supports modularization of cross-cutting concerns. The demonstration will show how the Eclipse Java support is being extended to provide an AspectJ aware editor, an outline view allowing navigation of cross-cutting structure, support for multiple build configuration files within a project (giving selective application of aspects), an AspectJ builder for compilation, task-list integration and several other features.

Schedule:

- Thursday, June 13, 11:30, room A
- Friday, June 14, 12:15, room A

D03: Separating Computation from Coordination for Java Applications

Presenters: Joao Gouveia, Georgios Koutsoukos, Michel Wermelinger, Luis Andrade and José Luiz Fiadeiro, Oblog Software SA

Coordination contracts are a modelling primitive, based on methodological and mathematical principles, that facilitates the evolution of software systems. The use of coordination contracts encourages the separation of computation from coordination aspects, and the analysis of which are the ``stable'' and ``unstable'' entities of the system regarding evolution. Coordination contracts encapsulate the coordination aspects, i.e., the way components interact, and as such may capture the business rules that govern interactions within the application and between the application and its environment.

System evolution consists in adding and removing contracts (i.e., changing the business rules) between given components (the participants of the contract). As a result of an addition, the interactions specified by the contract are superposed on the functionalities provided by the participants without having to modify the computations that implement them. In fact, the components are completely unaware they are being coordinated. The contracts specify a set of rules, each with a triggering condition (e.g., a call to a method of one participant), and a rule body stating what to do in that case. Contracts are also unaware of the existence of other contracts. This facilitates enormously incremental system evolution, because explicit dependencies between the different parts of the system are kept to a minimum, and new contracts can be defined and added to the system at any time (even run-time), thus coping with changes that were not predicted at system design time.

For this approach to be usable in real applications, it requires a tool to support system development and evolution using coordination contracts. Capitalising on the expertise of Oblog Software in building development tools, the current version of the Coordination Development Environment (CDE) helps programmers to develop Java applications using coordination contracts. More precisely, it allows to write contracts, to translate them into Java, and to register Java classes (components) for coordination. The code for adapting those components and for implementing the contract semantics is generated based on a micro-architecture that is based on the Proxy and Chain of Responsibility design patterns. The CDE also includes an animation tool, with some reconfiguration capabilities, in which the run-time behaviour of contracts and their participants can be observed using sequence diagrams, thus allowing testing of the deployed application.

CDE is written in Java and requires JDK 1.2. It is freely available for download from www.atxsoftware.com/agility.html.

Schedule:

- Thursday, June 13, 12:15, room B
- Thursday, June 13, 17:15, room B

D04: Squeak: the Open-Source Smalltalk

Presenters: Stephane Ducasse, Software Composition Group (SCG-IAM, University of Berne)

Squeak is the open source Smalltalk being developed by Dr. Alan Kay and a part of the original team that invented Smalltalk the first object-oriented language, multi-windowing systems, bitmap manipulation at Xerox Lab. There also were pioneers in the use of GC, virtual machines, mouse, bitmap devices. Macintosh with the Lisa copied them. However, Squeak is more than just redoing the past. Squeak has strong multimedia and network capabilities. Moreover, Squeak is a platform used for experimenting all kinds of new topics related object-oriented programming from new object model such as mixin to new ways of teaching OO.

The demo will show the following aspects of Squeak:

- New graphics support (Morphic from Self, WarpBlt new bit manipulation)
- Net support (pop, html, flash reader)
- Web server (Comanche a Apache like web server written in Squeak, SeaSide for dynamic html page generation)
- 3D engine
- Music and sound support (sample, midi, digitalizer)
- Voice generation
- eToy: an environment for teaching kids to program.
- Handwriting recognition
- Current projects developed in Squeak

Moreover Squeak runs on all the possible hardware and OS. Indeed the VM is also open source and written in a subset of Squeak and translated to C.

WebPages: http://www.squeak.org/

Schedule:

- Wednesday, June 12, 11:30, room A
- Friday, June 14, 14:30, room A

D05: SOUL: Supporting Object-Oriented Software Development with Logic Meta Programming

Presenters: Tom Tourwe, Johan Brichau, Universiteit Brussel
Kim Mens, Université catholique de Louvain

At the Programming Technology Lab of the Vrije Universiteit Brussel, research is being conducted on how the technique of logic meta-programming can be used to build state-of-the-art software development support tools. By capturing and expressing the interaction between the higher levels of software development (design, analysis, etc...) and the actual implementation level, we can co-evolve the source code and the higher-level software artifacts. The goal is to keep design, documentation and source code synchronised when the software evolves.

The interaction between source code and higher-level artifacts is expressed using a logic meta-programming language. This approach allows us to express these artifacts in a declarative way as meta-information of the program itself. This information can be actively used by the programming environment to guide the software development process in a variety of ways: to generate design from implementation and vice-versa; to provide support for refactoring, software reuse and evolution, aspect-oriented programming, generative programming, etc...

During this presentation, we will demonstrate SOUL (Smalltalk Open Unification Language), an open, reflective logic programming environment written in Smalltalk VisualWorks 5i4 and ported to various other Smalltalk dialects (such as Squeak). In the first part of the presentation we show how the environment and associated tools can be used in practice to:

- Reason about object-oriented programs (including naming and programming conventions, design pattern extraction, type inferencing, UML model extraction, architectural views,...)
- Declaratively generate code (including source code templates, aspect-oriented programming, design pattern code generation, ...)
- Support software evolution (including refactoring, software merging, ...)

During the second part of the presentation, we take a closer look at the technical details. We show how high-level software artifacts can be expressed in terms of logic facts and rules. For this, we require that the logic meta programming language lives in symbiosis with the object-oriented base language, allowing base level programs to be manipulated as terms, facts or rules in the meta level. In the case of Smalltalk, we use the reflection facilities to implement a logic meta layer on top of the Smalltalk meta-object protocol.

Schedule:

- Wednesday, June 12, 12:15, room B
- Friday, June 14, 15:15, room A

D06: Typed Concurrent Objects

Presenters: Vasco T. Vasconcelos, University of Lisboa
Luis Lopes, University of Porto

TyCO stands for “Typed Concurrent Objects". A few basic constructors in the lan-guage provide for a form of concurrent object-based programming (that is, objects but no inheritance), without requiring primitive objects. The language is quite simple: the basic syntax reduces to half-a-dozen constructors, and can be thought of as an assembly language for concurrent, object-based, distributed programming. To help in writing common programming patterns, a few derived constructors are available. TyCO is implicitly typed: a type inference system assigns monomorphic types to variables, and (_ a la ML) polymorphic types to procedures. TyCO provides for a simple model of concurrent object-based programming that combines the benefits of the formal framework of process calculi with the characteristics of Agha and Hewitt's actor systems.

The Virtual Machine runs byte-code files generated by a compiler. The compiler, incorporating a type-inference system, gathers static information that is used to generate optimized code. The design and implementation of the virtual machine focuses on performance, compactness, and architecture independence with a view to mobile computing. The byte code has a simple layout and is a compromise between size and performance. For specially crafted benchmarks, the current implementation performs close or better than languages such as Pict, Oz, and JoCaml.

The TyCO distribution is available at http://www.ncc.up.pt/~lblopes/tyco/, and includes the TyCO compiler (tycoc), the emulator (tyco), example applications, and docu-mentation (tutorial, getting started, language de_nition).

Schedule:

- Wednesday, June 12, 14:30, room B
- Thursday, June 13, 11:30, room B

D07: LAVA -Object-based Dynamic Inheritance for Java

Presenters: Günter Kniesel, Uwe Bardey,Tobias Windeln and Jörg Gonska, University of Bonn

The traditional notion of inheritance, which dates back to Simula 1967 and Smalltalk 78,has often been criticized as being too rigid to express dynamic evolution of structure and behaviour and too coarse-grained to express object-specific sharing of state and behaviour. A more expressive alternative, object-based dynamic inheritance, also known as delegation, was proposed in 1987 by Henry Liebermann. However, it has not yet made its way into mainstream object-oriented languages on the premise of its assumed inefficiency and incompatibility with static typing.

LAVA.LAVA is an extension of Java that refutes these assumptions. It supports statically type safe delegation and its new implementation shows that the code for dynamic inheritance generated by a compiler does not need to be less efficient than manual encoding of the same functionality.

In LAVA, object-based inheritance can be specified simply by adding the keyword delegated to a variable declaration, e.g. “delegated Financial Component parent. Objects referenced by such variables are called parents .The effect of inheriting methods and state is achieved by automatically forwarding locally undefined method and variable accesses to parents. When an applicable method is found in a parent it is executed after binding this to the initial message receiver. This binding of this is the defining characteristic of delegation. It lets methods of delegating objects override methods of their parents, just like subclass methods override superclass methods in class-based languages.

In LAVA, determination of the "inherited" code and of part of the inherited interface can take place at object instantiation time or can be fully dynamic. In the first case we talk about fixed delegation and use the usual Java keyword final to indicate that after initialisation parent objects cannot be exchanged. Fixed delegation is particularly interesting because it can be fully unanticipated –objects instantiated from any class can be used as parents. The implementation of parent classes does not need to be aware of delegation and does not need to include any hooks to enable delegation. So delegation can be used as a mechanism for adaptation of existing objects to unanticipated changes.

LAVA shows that fully dynamic (mutable) delegation has to be anticipated in order to make it compatible with static typing. The main idea is that classes whose subclass instances may be used as dynamically exchangeable parents must be annotated with a keyword that indicates this intended use.

The demo. The session will consist of two parts, a slide-based presentation and a practical demonstration. The presentation will introduce object-based dynamic inheritance, the related language extensions of LAVA and the reasons why they are designed exactly this way. The practical part will show how typical scenarios for representing objects with evolving or context-dependent behaviour (e.g. the wrapper, decorator, state and strategy pattern)can be encoded with minimal effort and how delegation can be used for unanticipated adaptation. The run-time and space costs of the LAVA –based implementations will be compared to implementations of the same functionality in pure Java.

Schedule:

- Wednesday, June 12, 14:30, room A
- Thursday, June 13, 14:30, room A

D08: A Threaded Code Interpreter for Rotor, Microsoft's Shared Source Implementation of the Common Language Infrastructure (CLI)

Presenters: Yahya H. Mirza, Aurora Borealis Software

In 1994, as an architect Mr. Mirza provided consulting support for Spatial Technologies on their core ACIS Geometry Libraries and it's integration with COM. Additionally, Mr. Mirza has been researching compositional frameworks since he was a visiting member of Ralph Johnson's Smalltalk Programming Group at the University of Illinois in 1995. Mr. Mirza also consulted for SAFECO Life Insurance during the early phases of design for their distributed intranet architecture. Finally in 1999, Mr. Mirza through his company Aurora Borealis Software was a Software Design Engineer in the Microsoft .NET Web Objects and Messaging Team. Currently Mr. Mirza is conducting component technology research utilizing Microsoft's Shared Source Rotor implementation of the CLI.

Language interoperability and distributed objects are areas that have been of interest for quite some time now. Commercial implementations have varied from a standardized meta-object system based on inheritance, i.e. IBM's SOM, to a binary standard with standard compositional idioms, i.e. Microsoft's COM. With the wide scale adoption of the Java platform, the importance and mind-share of language agnostic solutions was put on the backburners. Recently, Microsoft has introduced their next major evolution of their computing platform, the .NET initiative. The core of this initiative is a language agnostic runtime system, which is being standardized by ECMA. This effort is called the Common Language Infrastructure or the CLI. Finally, the language interoperability and integration solutions promised in the 90's are now becoming a pervasive commercial reality. Furthermore, the multi-vendor adoption of Microsoft's .NET initiative provides a great opportunity for language researchers. This opportunity allows a language researcher to innovate in their particular domain, while interoperating with existing commercial and research oriented language based solutions.

Recently, Microsoft released a "Shared Source" implementation of the CLI available on both BSD UNIX as well as the Windows platform codenamed "Rotor". A key technical advantage of the CLI is the ability of its intermediate language to support multiple language paradigms, as opposed to a single rigid object model. For language designers, Rotor can serve as an effective runtime core for experimentation at the language feature level. For compiler and virtual machine researchers, Rotor provides a context for applied research into alternative object representations, method dispatch, garbage collectors, JIT compilers, etc. My goal is to provide an in depth exploration into Rotor using my threaded code interpreter as an example Rotor extension.

The objective of this demonstration is to enable the language or virtual machine researcher to use the Rotor implementation, as a host environment for their research. Not only will the demonstration attendees learn about the inner workings of the Rotor implementation, they will also learn about alternative design options and their ramifications which led to the current architecture of the CLI. The approach to describing Rotor as well as my threaded code interpreter will be based on identifying the patterns and idioms embodied within Rotor, as well as describing their implementation details. Additionally, I plan to illustrate similarities and differences with other runtime systems including the Squeak and Java virtual machines.

Schedule:

- Wednesday, June 12, 15:15, room A
- Thursday, June 13, 15:15, room A

D09: GME: A Reflective Environment for Domain-Specific Modeling

Presenters: Akos Ledeczi, Institute for Software Integrated Systems Vanderbilt University

The Generic Modeling Environment (GME 2000) is a metaprogrammable graphical editor supporting the design, analysis and synthesis of complex, software-intensive systems in diverse engineering fields. It is closely related to such metaCASE tools as MetaEdit+ or Dome. GME 2000 is the result of over a decade long research at Vanderbilt University in model integrated computing. The toolset has been applied to several real world applications throughout the US.

GME 2000 has a component-based architecture using MS COM technology and is implemented in C++. The Core component exposes the domain-specific language specification through a set of COM interfaces. Another set of interfaces for bi-directional model access and also exposes all the model modification events. All the other tool components, (diagram editor, decorators, browser, event-based OCL constraint manager, software generators, etc.) are built independently around the Core. Model persistence is supported via standard database technology and XML import/export functionality. The standard technologies applied throughout the architecture (UML, OCL, COM, XML) make GME 2000 easily applicable and extensible to a wide variety of domains.

After introducing the basic concepts, the demonstration will start by building a domain-specific graphical modeling environment for hierarchical Finite State Machines fromscratch. Then we'll demonstrate the scalability of the approach by focusing on a complex real-world example domain, an integrated simulation framework for embedded systems. We'll show the UML and OCL based metamodels specifying the domain-specific visual modeling language. We'll reveal how the environment supports metamodel composition. We'll demonstrate how the domain-specific environment is generated automatically from the metamodels. We'll emphasize how the target visual modeling language supports such OO concepts as type inheritance and multiple aspects. We'll show an example application, an Automatic Target Recognition system, including its complex models and the automatically generated Matlab simulation and C implementation of it.

Schedule:

- o Thursday, June 13, 16:30, room A
- o Friday, June 14, 14:30, room B

D10: Using the Montages Language Prototyping Tool to add Object-Orientedness to Xasm

Presenters: Philipp W. Kutter, Applied Formal Methods Institute AG and ETH Zurich

We have developed the Montages method and tool for specifying and implementing domain-specific languages. Montages is based on EBNF, Attribute Grammar, finite state machines (FSMs), and a small imperative language called eXtensible Abstract State Machines (Xasm), which is a generalization of Gurevich’s Abstract State Machines (ASMs). Imperative features of a described language such as destructive updates and jumps can be modeled in a direct way. Xasm is not only used to give actions associated with states of the FSMs and control-guards associated with the transitions of the FSMs, but as well for giving semantics to Montages and as basis for implementing Montages. The Xasm language itself is specified and implemented using the Montages tools, leading to different possibilities of bootstrapping. In this tool demo we show how the Montages tool is currently used to design the object-oriented version of Xasm and we sketch how the new design is bootstrapped.

Schedule:

- o Wednesday, June 12, 11:30, room B
- o Wednesday, June 12, 16:30, room B

D11: SoftwarePot:Encapsulated File Spaces for Secure Software Circulation

Presenters: Yoshihiro Oyama, Kazuhiko Kato, Japan Science and Technology Corporation

Katsunori Kanda, University of Tsukuba

We will demonstrate our object-oriented system SoftwarePot, which provides a sandboxed execution environment called pot .Pots are different from traditional sandboxes in that each pot contains its own virtual file space. SoftwarePot is useful for preventing malicious or vulnerable software from accessing or damaging real crucial files. Our demonstration will show SoftwarePot running on a PC (Linux/x86).Visitors to our booth can see the execution of widely used programs (e.g. bash, Emacs and GIMP)being confined in a pot. They can also experience operating SoftwarePot by themselves.

Technical Merits:

- o Users can flexibly control programs running in a pot by specifying security policies for controlling socket communication and file mappings between a pot and real files.
- o A file space created in a pot is independent of the real file space. Users can hide a /etc/passwd .file by supplying a fake /etc/passwd file to a pot.
- o Pots can be serialized and transferred; SoftwarePot can create a plain file that represents the snapshot of a pot.The plain .le is transferred between sites as software combined with its execution state.

- SoftwarePot is a user-level middleware located between applications and operating systems. Executing SoftwarePot does not require a root privilege. In addition, SoftwarePot does not force programs to be written in a specific language.

Novelty:

- SoftwarePot can create a software package that includes a virtual file space for sandboxing the software; few systems are able to do that.
- SoftwarePot can encapsulate file spaces by means of a flexible access-control mechanism, which is similar to ones seen in object-oriented systems. There is little research on encapsulating file spaces in an object-oriented manner.

Relevance to Object-Oriented Technology. Object-oriented technologies have had a strong influence on the design of SoftwarePot. SoftwarePot manages software packages in an object-oriented manner. Users can deal with encapsulated file spaces (pots)as objects and files as object members. Some files in a pot are public; they are shared with other pots. Others are private ;they are not accessible from outside the pot. Files in a pot cannot only be mapped to real files but can also be mapped to processes. This feature is similar to the property mechanism in C#.

SoftwarePot web page:http://www.osss.is.tsukuba.ac.jp/pot/

Schedule:

- Thursday, June 13, 14:30, room B
- Friday, June 14, 11:30, room B

D12: LES Objects: A Model-based Code Generation Environment for Object-Oriented Conceptual Modeling of Web Application Interfaces

Presenters: Jaime Gómez, Cristina Cachero, Antonio Párraga, Web Engineering Group, University of Alicante

Existing tools intended to build and deploy engaging complex Web sites (including functionality) have shown to be inadequate to face the software production process in an unified and systematic way, from a precise system specification to the corresponding final implementation. In this context, where new technologies are continuously emerging, new approaches are required for the web developer to address the entire software lifecycle, from design to development to deployment to maintenance, in a way that is consistent, efficient and lets developers write language-independent applications.

To address this concern we report the Object-Oriented Hypermedia (OO-H) methodological proposal and their associate precompetive CASE Tool that, using an Object-Oriented approach, captures all the relevant properties involved in the modeling and implementation of Web Application Interfaces. The design process involves the construction of two additional views, complementary to those captured in traditional, UML-compliant, conceptual modeling approaches. These are, namely, the Navigation View, which extends a class diagram with hypermedia navigation features, and the Presentation View, in which the different elements regarding interface appearance and behavior are modeled by a series of interconnected template structures, expressed in XML. As a result, a language-independent front-end specification is obtained. From there a web interface, easily integrable with preexistent logic modules, can be generated in an automated way.

The CASE tool provides an operational environment that supports all the methodological aspects of OO-H Method. It simplifies the design and implementation of web-based Information Systems from an object-oriented perspective, providing a comfortable and friendly interface for elaborating the OO-H Method models. The most interesting contribution of this CASE environment is its ability to generate the web application front-end in well-known industrial software development environments. This CASE Tool is being used at this moment for the resolution of real web applications, in the context of a R&D projects carried out jointly by the Web Engineering Group of the University of Alicante and a set of industrial partners.

Schedule:

- Wednesday, June 12, 17:15, room B
- Thursday, June 13, 16:30, room B

D13: Octopus: A PostWIMP Framework for New Interaction Techniques

Presenters: Mads Brøgger Enevoldsen, Michael Lassen, Computer Science Dept., University of Aarhus

The Octopus project integrates advanced interaction techniques based on a novel user interface architecture to support new interaction methods in complex graphical tools. Examples of such tools could be UML editors, state charts, Petri nets, simulations etc.

The system is developed in the object-oriented programming language BETA (http://www.daimi.au.dk/~beta).

Graphics are implemented with the OpenGL graphics library to create visual effects such as transparency and shadows and to take full advantage of the growing capabilities of new strong graphics cards in current and future generations of personal computers. The choice of BETA/OpenGL was made due to the platform independence of these. Also Octopus is a good example of a framework utilizing BETA's powerful abstraction mechanisms including virtual classes and nested classes.

The system supports simultaneous inputs from multiple pointing devices based on the widespread USB standard. This allows for the two-handed interaction forms known from everyday life.

One such use of two-handed input supported by the framework is the use of semi-transparent "tool glasses" controlled by one hand. Through this toolglass actions can be applied to objects on the canvas with the use of the other hand. This is similar to the work form used when holding a nail in one hand and applying a hammer with the other. Other two-handed interactions are: Zooming by stretching and scrolling by panning (i.e. no need for zoom menu and scrollbars).

Another interaction form supported by the framework is the so-called "marking menus". These are context dependant circular shaped menus in which items are selected by dragging the mouse in a direction (as opposed to dragging the mouse a specified distance as known from traditional linear menus). By having integrated a gesture recognition algorithm the user can select a menu item with a quick "stroke" even without waiting for the menu to appear. Measurements have proven up to 40% faster interaction by using this principle.

The demonstration will show applications developed on top of this framework.

One such example is the CPN/Tools application (http://www.daimi.au.dk/CPNtools) for editing and simulating Petri Nets.

We expect to have a version of Octopus ready for free download at the time of the conference.

If time allows, we will also give a short overview of the status of some of the other research projects going on at our institute, e.g.

1. BETA.Eclipse: Integration of BETA in the Eclipse framework (http://www.eclipse.org)
2. BETA.java/BETA.Net: Porting of the BETA language to the Java VM and the Microsoft .NET platform.
3. Mjølner System. Current status of the system used for developing e.g. the Octopus framework.

Schedule:

- Wednesday, June 12, 15:15, room B
- Thursday, June 13, 15:15, room B

Posters

Poster sessions are an alternative forum for viewing results of object-oriented work, with the opportunity for one-on-one interaction with presenters. Poster themes cover the breadth of object-oriented technology from theory to experiences in applications. They provide an easy and informal means for presenting ongoing research to the ECOOP 2002 audience. This may be especially useful for new ideas that have not yet been developed to the point of a regular paper.

Posters will be on display during the entire conference.

ID	Title
P01	**Component Redundancy for Adaptive Software Applications** Ada Diaconescu, John Murphy
P02	**Understanding Performance Issues in Component-Oriented Distributed Applications: The COMPAS Framework** Adrian Mos, John Murphy
P03	**Design Compiler and Optimiser** Lucian Gabor, John Murphy
P04	**Enforcing Business Policies Through Automated System Reconfiguration** Luís Andrade, José Luiz Fiadeiro, Michel Wermelinger, Georgios Koutsoukos, João Gouveia
P05	**Extensible Java Pre-Processor EPP** Yuuji Ichisugi
P06	**A Framework for Checking UML Models – The UBB OCL Evaluator** Dan Chiorean, Adrian Cârcu, Mihai Pasca, Cristian Botiza, Horia Chiorean, Sorin Moldovan, Ilinca Ciupa
P07	**UniFrame: Framework for Seamless Integration of Heterogeneous Distributed Software Components** Barrett R. Bryant, Rajeev R. Raje, Andrew M. Olson, Girish J. Brahnmath, Zhisheng Huang, Changlin Sun, Nanditha N. Siram, Carol C. Burt, Fei Cao, Chunmin Yang and Wei Zhao, Mikhail Auguston
P08	**Darwin and Lava - Object-based Dynamic Inheritance in Java** Guenter Kniesel

PO1: Component Redundancy for Adaptive Software Applications

Authors:

Ada Diaconescu (Performance Engineering Laboratory, Dublin City University)
John Murphy (Performance Engineering Laboratory, Dublin City University)

Abstract:

The growing complexity of computer systems has led to more complicated and expensive system design, testing and management processes, and decreased systems flexibility. In the last years, research initiatives have started to address these issues, as a great challenge of modern software engineering and in general of the whole I/T industry. Self-evolving and self-adaptive system technologies, autonomic computing and redundancy as a basis for system robustness are some of these initiatives.

With respect to this, we are trying to address the inter-related issues of self-optimisation and self-'healing' of software applications. We focus on the component-based software development (CBSD) approach as a new solution that promises to increase the reliability, maintainability and overall quality of large-scale, complex software applications.

Our goal is to propose a new component technology that adds on the existing component technologies (e.g. EJB, .Net, or CCM) to support new functionalities such as adaptability, self-optimisation, or self-'healing'. We concentrate our solution around the concept of (software) component redundancy. By this concept we understand that a number of component implementation versions, providing the same services are available, at run-time. In case one component version instance fails, or performs poorly in certain conditions, it can be replaced with another component version that offers the equivalent functionalities.

The component technology must accommodate the presence of the redundant component implementation versions and be able to use them for achieving optimal performance and robustness.

As a new requirement on component implementations, each such component has to support and provide a formal description of itself. This description includes component functionality information and performance related parameters corresponding to various environmental conditions (e.g. available resources and number of concurrent client requests).

We identified the following entities needed for utilisation and management of redundant components:

- Application Monitor - monitors and evaluates the performance of active component variants. Identifies 'problem' component(s) in application transaction chains.
- Environment Monitor - keeps track of the current available resources (e.g. memory, storage, communications bandwidth, or processing) and number of client requests.
- Component Evaluator - determines the optimal component version, based on component descriptions, current environmental conditions and decision policies.
- Component Swapping Mechanism - swaps instances of different component versions, while preserving state and references consistency.

These entities operate at run-time in an automated, feedback loop manner. The application performance is monitored and evaluated, the optimal component version is identified and activated for each component and the resulting application is monitored and evaluated.

The aforementioned entities are part of the component technology we propose. Therefore, they do not have to be (re-)implemented for each component based software application that uses this component technology. Application servers are already providing some common services such as security or transactions. The component redundancy based optimisation and adaptability would be another such service.

We intend to concentrate on specifying the formal component description, Component Evaluator, and Component Swapping Mechanism and identify existent application and environment monitors.

Our prime goal will be to demonstrate our approach for an existent component technology (e.g. EJB) by integrating the aforementioned entities with an open-source application server and simulating environmental conditions variations.

Additional information: http://www.eeng.dcu.ie/~diacones/workCurrent.html

PO2: Understanding Performance Issues in Component-Oriented Distributed Applications: The COMPAS Framework

Authors:

Adrian Mos (Performance Engineering Laboratory, Dublin City University)
John Murphy (Performance Engineering Laboratory, Dublin City University)

Abstract:

We propose the Component Performance Assurance Solutions (COMPAS) framework that can help developers of distributed component-oriented applications find and correct performance issues during as well as after development.

The motivation for our work on this framework was derived from the following considerations:

- Performance can be critical for large-scale component oriented applications.
- A poor architecture, a bad choice of Commercial-Off-The-Shelf (COTS) components or a combination of both can prevent achieving the application performance goals.
- Performance problems are more often caused by bad design rather than bad implementations.
- Current performance modelling and prediction techniques, mainly require developers to build performance models based on assumptions about the run-time characteristics of the software application, which often is difficult to do, especially for complex middleware-based applications.
- Often, performance is "a function of the frequency and nature of inter-component communication, in addition to the performance characteristics of the components themselves" [P.C. Clements].

To address these issues, the presented framework is structured into three main functional parts or modules, that are interrelated:

- Monitoring: obtains real-time performance information from a running application without interfering with the application code or the application run-time infrastructure (i.e. the application server implementation). We are currently concerned with monitoring of EJB systems only.
- Modelling: creates UML models of the target application using information from the monitoring module. The models are augmented with performance indicators and can be presented at different abstraction levels to improve the understanding of the application from a performance perspective.
- Performance Prediction: the generated models of the application are simulated with different workloads (e.g. corresponding to different business scenarios); simulation results can be used to identify design problems or poor performing COTS components.

There is a logical feedback loop connecting the monitoring and modelling modules. It refines the monitoring process by focusing the instrumentation on those parts of the system where the performance problems originate.

The intent of the framework presented is not to suggest a development process that prevents the occurrence of performance issues in the design, but rather to enable early discovery of such issues and suggest corrections.

Models are represented in UML, which many enterprise-scale application developers are familiar with. The use of Model Driven Architecture and Enterprise Distributed Object Computing (EDOC) concepts facilitates navigation between different abstraction layers. The top-level models are represented using a technology independent profile, the Enterprise Collaboration Architecture from EDOC, in order to benefit from a standardized form of representation for business modelling concepts. Lower level models are represented using UML profiles such as the UML Profile for EJB which provide means to illustrate technology specific details. Regardless of the level of abstraction, each model is augmented with performance information presented using the UML Profile for Schedulability, Performance, and Time Specification.

The Performance Prediction Module uses runnable versions of the generated modules and simulates them with different workloads as inputs displaying performance information in the same manner as in the modelling phase.

Additional information: http://www.eeng.dcu.ie/~mosa/projects/compas.html

PO3: Design Compiler and Optimiser

Authors:

Lucian Gabor (Dublin City University)
John Murphy (Dublin City University)

Abstract:

A major part of software systems are not designed for performance. Very often there are flows in the system that appear late in the life cycle of the system. Even if there are profiling and monitoring tools that can detect various problems, they don't provide a solution. It takes a very experienced designer or developer to deal with such problems efficiently. Solutions for a good design are the use of design patterns, refactorings and antipatterns.

Our framework provides a way to automatically correct design problems preserving the functionality of the system and improving its performance. The approach is to provide a system of patterns, antipatterns and refactorings that will provide the instruments for automation of design optimisation. The idea is similar to the work that a compiler does when it optimise it's output. The tool will use these instruments mainly to change the design of a model. Another task is to comprehend it. It will use patterns to understand how a system is modelled. It will divide the system into subsystems and components making it easier to process. It will gather information about participants to existing patterns.

An important feature of the process is that it will take a non-intrusive approach when redesigning. The functionality of the system will be maintained. Each pattern that solves a problem will be defined as a sequence of refactorings and/or it will have aspect-oriented strategies.

Other aspects are reuse of the experience and interoperability.

Finding similarities between different models, in order to better recognize patterns that are implemented into the new model, manifests the reuse of experience. Additional problems can be detected by finding

similar scenarios between two models. Also, since the definition of the context a pattern address isn't usually comprehensive or complete, it can learn new situations in which a pattern may be used.

The interoperability is addressed by the use of UML and MOF standards for representing knowledge.

In conclusion the main targets are the system of patterns and the expert system that has the ability to compare models, find similarities and localise the context of a pattern or a performance problem.

As stated by the POSA books system of patterns should comprise a sufficient base of patterns, it should describe all its patterns uniformly, it should expose the various relations between patterns, it should organise its constituent patterns, it should support the construction of software systems and it should support its own evolution.

Additional information: http://www.eeng.dcu.ie/~gaborl/unofficial/gabor-it&t.pdf
http://www.eeng.dcu.ie/~gaborl/unofficial/gabor-ecoop-2002.examples.src.zip

PO4: Enforcing Business Policies Through Automated System Reconfiguration

Authors:

Luís Andrade (ATX Software SA, Oblog Software SA)
José Luiz Fiadeiro (ATX Software SA, Dep. de Informática, Fac. de Ciências, Univ. de Lisboa)
Michel Wermelinger (ATX Software SA, Dep. de Informática, Univ. Nova de Lisboa)
Georgios Koutsoukos (Oblog Software SA)
João Gouveia (Oblog Software SA)

Abstract:

The engineering of Business Systems is under the increasing pressure to come up with software solutions that allow companies to face very volatile and turbulent environments (as in the telecommunications domain). This means that the complexity of software has definitely shifted from construction to evolution, and that new methods and technologies are required.

Most often, the nature of changes that occur in the business are not at the level of the components that model business entities, but at the level of the business rules that regulate the interactions between the entities. Therefore, we believe that successful methodologies and technologies will have to provide abstractions that reflect the architecture of such systems by supporting a clear separation between computation, as performed by the core business components, and coordination, as prescribed by business rules. This separation should help in localising change in the system, both in terms of identifying what needs to be changed in the system and circumscribing the effects of those changes.

In our opinion, the lack of abstractions for supporting the modelling of interactions and architectures explains why component-based and object-oriented approaches have not been able to deliver their full promise regarding system evolution. Usually, interactions are coded in the way messages are passed, features are called, and objects are composed, leading to intricate spaghetti-like structures that are difficult to understand, let alone change. Moreover, new behaviour is often introduced through new subclasses which do not derive from the "logic" of the business domain, widening the gap between specification and design.

The approach we have been developing builds on previous work on coordination models and languages, software architecture, and parallel program design languages. Instead of delegation we use explicit architectural connectors that encapsulate coordination aspects: this makes a clear separation between computations and interactions and externalises the architecture of the system. Instead of subclassing we

advocate superposition as a structuring principle: interactions are superposed on components in a non-intrusive and incremental way, allowing evolution through reconfiguration, even at run-time.

The main advantages of our approach are adequacy and flexibility. The former is achieved by having a strict separation of computation, coordination, and configuration, with one primitive for each concept, stating clearly the pre-conditions for each coordination and reconfiguration rule. As for flexibility, interactions among components can be easily altered at run-time through (un)plugging of coordination rules, and it is possible to state exactly which coordination rules are in effect for which components, and which configuration policies apply to which parts of the system.

In the following sections we briefly summarise our approach for different phases of software development. More details are provided by the publications available at the ATX website by tutorial 11 at this conference.

Additional information: www.atxsoftware.com/agility.html

PO5: Extensible Java Pre-Processor EPP

Authors:

Yuuji Ichisugi (Information Technology Institute, National Institute of Advanced Industrial Science and Technology (AIST))

Abstract:

We are developing an extensible Java pre-processor, EPP. EPP is a framework of Java source code processing systems. The behavior of EPP can be extended by adding extension modules, EPP plug-ins. EPP has an extensible recursive descent parser and an extensible type checking mechanism. A great variety of new language features can be easily introduced by implementing macro expansion functions for new language constructs. Independently developed plug-ins can be used in a program simultaneously. EPP can be used also as a platform for source code processing systems such as metrics tools and refactoring tools.

A wide variety of language extensions have been implemented, including a data-parallel language, thread migration, parameterized types, metrics tool and the difference-based module mechanism. (The paper describing difference-based modules are accepted by ECOOP2002.)

EPP plug-ins are written in an extended Java language. The extensions for EPP plug-ins themselves are implemented using EPP. The language extensions support a kind of mixin-based programming, which enhances extensibility and composability of applications. In addition, some useful features such as symbols and backquote macros as in Lisp have been introduced into Java language in order to support easy implementation of macros.

The most important application of EPP is the MixJuice language, which is an enhancement of Java language with difference-based module mechanism. The module mechanism enables separation of crosscutting concerns. We are currently rewriting EPP itself using the MixJuice language, in order to enhance extensibility, performance and safety.

EPP is distributed from the web page with source-code and sample plug-ins.

Additional information: http://staff.aist.go.jp/y-ichisugi/epp/

PO6: A Framework for Checking UML Models – The UBB OCL Evaluator

Authors:

Dan Chiorean
Adrian Cârcu
Mihai Pasca
Cristian Botiza
Horia Chiorean
Sorin Moldovan
Ilinca Ciupa ("Babes-Bolyai" University - Computer Science Research Laboratory)

Abstract:

The main objective of the checking process
Obtaining correct UML models.
This implies:
- Reduced time & costs with system development
- Better system quality

UML model correctness = completeness + correctness (against UML Specification)
- Precondition – a correct and complete set of WFR

State of the art
Today, checking of UML models is done using:
- Checking operations implemented in UML CASE Tools
- Scripts
- Other Add-Ins tools (like Rose Checker)

Drawbacks of the above techniques:
- Only a part of the WFR are implemented
- The equivalence between the WFR and their implementation has to be proved
- Each CASE tool has its own Repository interface and script language
- A part of the WFR cannot be implemented due to the lack of information

OCL Evaluator
Technique used:
- Check UML Models by means of WFR
- The UML models are expressed in XMI 1.1 format
- The user has full access to the metamodel information by means of UML AO

Consequently the user can extend the checks at different profile levels (including checks specific to the target language)

All the rules expressed at the metamodel (M2) level. The constraints can be fully evaluated. Apart from the UML model, no additional information is necessary. The rules are stored in a text file, conforming to the OCL standard.

The OCL support is fully compliant with the OCL 1.4 Specification.

The BCR specified by the user at the M1 level can also be evaluated provided that all the required information is available.

The results obtained
Using NEPTUNE OCL Evaluator, most of the UML 1.4 WFR were evaluated.

- Apart from the errors discovered in case of similar projects (see http://www.db.informatik.uni-bremen.de/projects/USE) a lot of conceptual errors have been identified in the AO and WFR Specification
- The main conclusions are:

The WFR have to be clearly explained in natural language and if needed, exemplified using pieces of models. Providing examples of correct models and of modes breaking the WFR gives the user the possibility to better understand the meaning and importance of each rule.

The UML WFR do not have to forbid legal constructions in object oriented programming languages.

The UML metamodel has pure object oriented architecture. Given that the AO and the WFR specify the UML metamodel classes behavior, their specification has to satisfy the rules specific to OOP and to the design and programming by contract domains. (The naming rules, the redefinition rules, the rules adopted for solving the ambiguities in multiple inheritance, etc.)

The AO and the WFR specifications have many solutions. It is very important to find the simplest and clearest one.

We are now trying to identify all the errors in the AO and the WFR and propose solutions for them

Most of this research work has been done in the framework of NEPTUNE IST 1999-20017 Research European Project

Additional information: http://lci.cs.ubbcluj.ro

PO7: UniFrame: Framework for Seamless Integration of Heterogeneous Distributed Software Components

Authors:

Barrett R. Bryant (Department of Computer and Information Sciences, University of Alabama at Birmingham)

Rajeev R. Raje
Andrew M. Olson
Girish J. Brahnmath
Zhisheng Huang
Changlin Sun
Nanditha N. Siram (Department of Computer and Information Science, Indiana University Purdue University Indianapolis)

Carol C. Burt
Fei Cao
Chunmin Yang
Wei Zhao (Department of Computer and Information Sciences, University of Alabama at Birmingham)

Mikhail Auguston (Department of Computer Science, New Mexico State University)

Abstract:

A framework is proposed for assembling software systems from distributed heterogeneous components. For the successful deployment of such a software system, it is necessary that its realization not only meet the functional requirements, but also non-functional requirements such as Quality of Service (QoS) criteria. The approach involves: a) the creation of a meta-model for components, called the Unified Meta

Model (UMM), and an associated hierarchical format for indicating the contracts and constraints of the components, b) resource discovery components, called "headhunters," that locate components according to a software developer's request, c) automatic generation of glue and wrappers, based on a designer's specifications, for achieving interoperability, d) guidelines for specifying and verifying the quality of components and their compositions, e) a formal mechanism for precisely describing the meta-model, f) a methodology for component-based software design, and g) validation of this framework by creating proof-of-concept prototypes. A formal specification based on Two-Level Grammar is used to represent these notions in a tightly integrated way so that QoS becomes a part of generative domain models.

Additional information: http://www.cs.iupui.edu/uniFrame

PO8: Darwin and Lava - Object-based Dynamic Inheritance in Java

Authors:

Guenter Kniesel (Computer Science Department III University of Bonn)

Abstract:

The traditional notion of static class-based inheritance is too rigid to express dynamic evolution of structure and behaviour and too coarse-grained to express object-specific sharing of state and behaviour.

Back in 1987 Henry Liebermann proposed a complementary approach: object-based dynamic inheritance, also known as delegation. In spite of its superior expressive power, object-based inheritance has not yet made its way into mainstream object-oriented languages on the premise of its (assumed) inefficiency and incompatibility with static typing.

These assumptions are refuted by the Darwin model and its proof of concept implementation, Lava. Darwin is a general model for statically typed class-based object-oriented systems with object-based dynamic inheritance [1]. Lava is a corresponding extension of Java.

Like in previous approaches, delegating objects in Darwin can "inherit" variables and methods from other objects, which they reference in special "parent variables". The inheriting objects are called "children" and the object from which they inherit are their "parents". The effect of heritance is achieved through a message forwarding mechanism that includes a specific treatment of the "self" (or "this") pseudovariable. Messages for which a message receiver has no applicable method are automatically forwarded to its parents. When an applicable method is found in a parent it is executed after binding its implicit self parameter to the message receiver. This specific binding of self is the defining characteristic delegation. It has the effect that methods of children override methods of parents, just like subclass methods override superclass methods in class-based languages. Delegation can be "fixed" or "mutable":

- "fixed" delegation means that parent objects are never exchanged (after initialisation);
- "mutable" delegation allows exchange of parents at any time.

Note that even in the first case, delegation is still dynamic. The determination of the "inherited" code and of part of the inherited interface is deferred to object instantiation time.

Unlike previous approaches, Darwin shows that object-based inheritance can be supported without sacrifincing static type safety, even in the presence of subtyping.

Furthermore it shows that fixed delegation can be fully unanticipated. That means that objects instantiated from _any_ class can be used as parents. The implementation of parent classes does not need to be aware of delegation and does not need to include any hooks to enable delegation. So delegation can be used as a mechanism for adaptation of existing objects to unanticipted changes in their environment [2].

Finally, Darwin shows that mutable delegation has to be anticipated in order to make it compatible with static typing. The main idea is that classes whose subclass instances may be used as dynamically exchangeable parents must be annotated with a keyword (dynamic root).

This annotation "freezes" the type of "this" in the dynamic root and all its subclasses. Due to the restricted type of "this", messages that are specific to one of these subclasses cannot be sent to "this". Doing so would be unsafe because "this" might replace its current parent by an instance of a dynamic root subclass that does not contain the specific method.

The poster tries to convey the essential ideas of Darwin by taking advantage of the fact that "a picture says more than a thousand words".

Additional information: http://javalab.cs.uni-bonn.de/research/darwin/
http://www.cs.uni-bonn.de/~gk/

BOF’s

"Birds-of-a-feather" (BOF) meetings provide an easy and informal means for letting people interested in a given topic join and dicuss about it.

This year, two BOF meetings have been organized together with the conference:

- The Code-in and Refactoring Party
- The Eclipse BOF

Both will happen on Thursday June 13 at the conference site, from 18:00 (ie. right after the technical sessions) onwards. Beverages and snacks will be served during these BOFs.

After the meetings, buses will take BOFs participants to the hotels (departing from the School of Computer Science at 20:10) and/or to the conference banquet (departing at 21:00).

Apart from these two pre-arranged meetings, rooms will be available during the conference for anyone interested in organizing any other BOF. They will be advertised at the registration's notice board.

The Code-in and Refactoring Party (Lab. 3.3.7)

Somewhat in the spirit of the annual Camp Smalltalk at OOPSLA, the Code-In is a communal BOF event to share our love and fun with programming and tools. In essence, the Code-In involves developers pair programming, sharing their skills and tools.

Developers are sometimes unfairly characterized as not being social-not true. We like to talk about programming and tools, and the Code-In is a hands-on place to do that.

In addition to participating as a student, the Code-In is a place where you may play the role of a coach; teaching another interested developer about something or some tool that would like to share. We're all potential coaches and students at the Code-In.

In addition to any and all playing the role of coach at the Code-In, we'll be working to invite various thought-leaders that may be attending the conference to play the role of coach in methods such as refactoring, test-first programming, code reviews, and tools and languages, but no promises on who will show up.

Rules of the Game

We can play all three roles at the party. Anyone may be a coach; anyone a student. In each, humor and gentleness is appreciated. So is beer.

This is fun with computers, not serious stuff.

1. Method-Oriented Coach - for example, let's pair program for 30 minutes while I coach you in how to refactor (using the code you brought on your laptop), or in test-first programming with JUnit.
2. Technology-Oriented Coach - for example, let's pair program while I coach you in AOP using AspectJ, or applying the refactoring features in the IDEA IDE.
3. Student - to a coach; usually does the hands-on typing.

Coach - Student games involve pair programming, usually with the Student at the keyboard - not the Coach, with the exception of code review games, or when otherwise appropriate. Coaches will have signs at their tables advertising what they can coach.

Equipment

If you are coaching, please bring your laptop, and tools and samples set up for ease of learning.

There will be tables, chairs, and power outlets.

If you would like to learn more about refactoring or have a code review, please bring your laptop and your code.

Refreshments: Drinks and snacks will be served during the party.

Examples

Share or learn what intrigues you: Refactoring, test-first programming, code review to polish Java, the new 1.4 java.nio library, ASP.NET, a performance analysis tool, XUnit, C#, AspectJ, Ruby, Python, XEmacs, creating web services, …

Expectations

Some Code-Ins have 5 people; some have 25. It's always a surprise.

Facilitator

Craig Larman - Craig will volunteer as a coach for at least AOP with AspectJ. He would like to be a student of many things. Craig is the author of Applying UML and Patterns, and the Java 2 Performance and Idiom Guide. He serves as Director of Process at Valtech, an international consulting company.

Sponsorship

The material and drinks for The Code-In and Refactoring Party have been sponsored by CORITEL BPM

The Eclipse BOF (room: Sala de Grados A)

Come and join eclipse developers, users, and tool builders at ECOOP. We'll start the evening with a tour of the new features of Eclipse 2.0 and several experience reports from researchers working with Eclipse. Then we'll open the floor to you for a roundtable discussion with our panel of Eclipse users and developers. Everyone is welcome, whether you're an experienced plug-in writer or have just heard about Eclipse!

You can find out more about Eclipse by visiting our website at www.eclipse.org.

Plan of the BoF

The BoF will follow right after the sessions at the University on Thursday June 13th. We'll be in Salon Grados A.

18:00 Welcome - Brian Barry, OTI
18:15 A Tour of Eclipse 2.0 - Erich Gamma, OTI
18:55 Experience Reports
1.Eclipse IDE for Beta - Ole Lehrmann Madsen, Aarhus University
2.Eclipse Aspect J Plug-in - Adrian Colyer, IBM
3.ArchJava - Jonathan Aldrich / Craig Chambers, U. of Washington
4.Perspectives - Andrew Black, OGI
19:45 Roundtable Discussion with Eclipse User/Developer Panel
20:30Close

Transportation

The ECOOP banquet follows right after the BoF and the organizers are making sure that we'll all be there on time! Buses will be leaving from the University as follows:

20.10 To the hotels (BoF will be ongoing)
20:30 Directly to the banquet

Sponsorship

IBM is supporting the Eclipse BOF.

Social Programme

Social programme		
Sunday 9	Guided tour to the city	17:00
Monday 10	Get-together party	18:00
Tuesday 11	Tapas' evening	Free tour - Málaga downtown
Wednesday 12	Welcome reception	18:00 - Gibralfaro Roman Castle
Thursday 13	Banquet dinner	21:00 - Señorio de Lepanto
Friday 14	Farewell drinks	16:30
Accompanying persons' special programme		
Wednesday 12	Guided Tour	12:15

Social Programme for Conference Participants

In addition to the technical and scientific programme, an extensive social programme has been arranged during the conference for making people enjoy the visit to Málaga as much as possible.

Guided tour to the city (Sunday 9, 17:00)

On Sunday afternoon, a guided tour to Málaga will show all ECOOP early-comers the city and its monuments.

Guests will be transferred by bus at 17:00 from the Conference venue located at the Teatinos campus to the city center. Then, we will enjoy a guided walk around this area, one of the most beautiful parts of the city. Málaga has a lot of monuments, among which the cathedral, the Alcazaba (a traditional morish castle) and Picasso's birthplace, are perhaps the most relevant ones.

Get-together party (Monday 10, 18:00)

After the technical activities of the first day, ECOOP 2002 welcomes both Tutorial and Workshop as well as Conference participants to a get-together party with drinks and snacks at the Conference venue. This will be an excellent opportunity for visiting the exhibition and making contacts with the exhibitors.

Tapas' evening (Tuesday 11, free tour – Málaga downtown)

The City of Málaga invites all ECOOP 2002 participants to enjoy the traditional tapas (small portions of food) at the city center. Participants wil be provided with 15 "verdiales" (a special currency only valid for eating out in some places in Málaga, 1 "verdial" = 1 €) and a list of bars and restaurants where they can enjoy the tapas and pay for them with verdiales. In addition, ECOOP participants will be usually invited by the bartenders of these tapas bars to a glass of wine with their specialities if paying with "verdiales".

Welcome reception (Wednesday 12, 18:00 – Gibralfaro Castle)

The City of Málaga will host a welcome reception at the Gibralfaro Castle, an impressive roman fortress overlooking the city and the Málaga bay.

Buses will depart at 18:30 from the Conference venue, taking participants through a tour of the city first and then to the Castle, where approx. at 20:00 they will be offered a cocktail reception.

A flamenco show will be featured during the reception.

Banquet dinner (Thursday 13, 21:00 – Señorío de Lepanto)

On Thursday 15 evening the banquet will be served at 21:30 at the Señorío de Lepanto, a beautiful Spanish villa located within the Natural Park of the Málaga Mountains.

Participants should show the invitation card to gain entrance to the banquet. The card will be given to all participants that have explicitly registered to the banquet when registering for the conference. Additional banquet cards can be bought along the conference at the registration desk.

Buses will transport participants from the hotels to the Señorío de Lepanto at 21:00, and back to the hotels once the banquet is over. Special buses will also run from the conference venue for those participants attending any of the BOF meetings happening on Thursday, departing as well at 21:00.

Our banquet speaker will be Kristen Nygaard, who has recently received--together with Ole-Johan Dahl-- the IEEE von Newmann Medal and ACM Turing awards for their contributions to Object-Oriented programming.

Farewell drinks (Friday 14, 16:30)

Right after the last technical session and the closing of the conference, some farewell drinks will be served at the Conference venue.

Accompanying person's programme

All registered accompanying persons are cordially welcome to join the conference social programme and evening activities. In addition, the following programme has been specially prepared for them.

Guided tour (Wednesday 12, 12:15)

This visit will depart at 12:15 from the Tourist Office located in Málaga's city park. This building –also known as "La Casa del Jardinero" (the gardener's house)–was built in 1908 and it is a very nice and small construction surrounded by plants and leafy vegetation. You will have the opportunity to visit the Historical Botanical garden "La Concepción", the second most important tropical garden in Europe.

The visit will also include a special lunch with traditional dishes of Málaga and Andalucía, in a typical restaurant sited on the Málaga's hills.

The visit and the lunch are free for all registered accompanying persons.

Useful to Know

Conference venue

ECOOP 2002 will be hosted by the School of Informatics (ETSI Informática) of the University of Málaga. All events will happen at the School premises.

Located in the "Campus de Teatinos", it has an easy access from Málaga downtown by bus or taxi (see the map page for details on how to get here). In addition, buses will be provided by the organization in the morning and evenings to take ECOOP participants to and from the hotels to the conference place.

Registration office

A registration office will be set up in the conference site. The office will be opened during all the conference from 8:30 to 17:30. It will also open on Sunday June 9, from 15:00 to 19:30.

Badges

You are kindly requested to wear your name badge at all times, as admittance to the sessions, coffee breaks, lunches, and the evening programme is not allowed without your badge. The conference assistants will be wearing specific badges for easy recognition.

Electronic mail

Electronic mail facilities and Internet connection will be available for ECOOP participants during the conference.

Meals

Meals and coffee-breaks are included for those who handle a valid pass to the Conference. Special food (vegetarian and pork-free food) has also been arranged.

Social and accompanying programme

An extensive social program has been arranged during the conference for making people enjoy the visit to Málaga as much as possible. Activities for accompanying persons are available as well, since we think that this is an excellent opportunity to bring your partner or family and enjoy the trip to Spain. Should you need further assistance on this matter, apart from the excursions and trips already organized, do not hesitate to contact the reception office.

Weather

Málaga is the capital of the Costa del Sol (Sun Coast). So, warm and nice temperatures are expected during the conference, ranging from 25C to 30C (77F-86F). Bring your swimming suit with you (for your spare days!), and do not forget some protection for the sun.

Health regulations

No vaccinations are required when entering Spain from any country.

Insurance

The conference organizers will not be able to take out any kind of insurance for participants or accompanying persons. Participants are required to make their own arrangements concerning insurance.

Currency

The Euro is the official currency in Spain.

Problems?

Organizational details during the conference will be dealt with at the registration desk. Should you need further assistance, do not hesitate to contact any person of the organization, that will try to help in everything they can.

Outside the conference hours, or if you are away from the conference venue, please dial **619385603** for contacting somebody from the ECOOP organization. In addition, the usual emergency and help numbers are always available:

- Local police: 092
- National police: 091
- Fire brigade: 080
- Health service: 061
- Red Cross: 95 222 222
- Toxicology: 91 620 420

Please do not hesitate to ring them in case of trouble. Málaga is a tourist resort used to deal with tourists and their problems, and normally very friendly to foreigners.