TUTORIALS

Chair: Craig Larman, *Valtech*

The OOPSLA conference is well known for its high quality tutorials that span a wide range of relevant and timely topics. OOPSLA 2001 has worked hard to continue this tradition by offering 70 carefully selected tutorials. The complete tutorial program covers a broad spectrum of topics selected to appeal to the interests of a wide range of participants, including practitioners, managers, and researchers. The tutorials are categorized below for ease of selection; individual tutorials may be found in multiple categories.

All tutorials include a lunch provided by OOPSLA.

TUTORIALS AT A GLANCE

Tutorials by Date

Sunday, 14 October 2001

Full Day – 8:30 am – 5:00 pm

1	A Brief Tour of Responsibility-Driven Design	Convention Ctr — Room 20
2	Testing Object-Oriented Software Systems	Convention Ctr — Room 13
3	Usage-Centered Design: An Agile Model-Driven Process for Object-Oriented User Interface Design	Convention Ctr — Room 25
4	Concepts of Object-Oriented Programming	Convention Ctr — Room 22
5	Lo-Fi Design Strategies for Creating Highly Usable Object-Oriented User Interfaces	Convention Ctr — Room 24

Half Day – Morning – 8:30 am – 12:00 noon

6	Inside High-Quality Software Architectures	Convention Ctr — Room 18
7	Dungeons and Patterns!	Marriott Hotel — Meeting Room 11
8	Introduction to Writing Use Cases	Marriott Hotel — Florida Salon V
9	Object-Oriented Design of Human-Computer Interaction	Convention Ctr — Room 15
10	Introducing Patterns (or Any New Idea) into Organizations	Convention Ctr — Room 14
11	Introduction to Concurrent Object-Oriented Programming in Java	Convention Ctr — Room 16
12	Agile Methodologies	Convention Ctr — Room 19

Sunday Morning (continued)

13	How to Manage the Change from COBOL to OOP	Marriott Hotel — Salon A
14	Component and Service Architecture Modeling with UML	Convention Ctr — Room 21
15	XML, XSD, and SOAP as a Better Component Model	Marriott Hotel — Florida Salon VI
16	An Introduction to Design Patterns	Marriott Hotel — Florida Salon IV
17	Producing GUIs with Java	Marriott Hotel — Meeting Room 12

Half Day – Afternoon – 1:30 pm – 5:00 pm

18	Designing Concurrent Object-Oriented Programs in Java	Convention Ctr — Room 16
19	Building Parsers with Java	Marriott Hotel — Meeting Room 11
20	Daily Builds Are for Wimps	Convention Ctr — Room 15
21	Designing with Patterns	Marriott Hotel — Florida Salon IV
22	The .NET Framework: The Common Language Runtime and C#	Marriott Hotel — Florida Salon VI
23	Garbage Collection	Marriott Hotel — Meeting Room 12
24	Advanced Use Case Writing	Marriott Hotel — Florida Salon V
25	Fractal Patterns and Frameworks in UML — Towards UML 2.0?	Convention Ctr — Room 21

Monday, 15 October 2001

Full Day – 8:30 am – 5:00 pm

26	Aspect-Oriented Programming with AspectJ™	Convention Ctr — Room 20
27	Software Architecture: It’s What’s Missing From OO Methodologies	Convention Ctr — Room 13
28	Improving Your Use Cases	Convention Ctr — Room 22
29	Pattern-Oriented Software Architecture: Patterns for Concurrent and Networked Objects	Convention Ctr — Room 15
30	Object-Oriented Reengineering	Marriott Hotel — Meeting Room 12

Half Day – Morning – 8:30 am – 12:00 noon

31	Patterns at Work	Marriott Hotel — Florida Salon VI
32	Designing an Agile Methodology	Convention Ctr — Room 19
33	Exposing and Consuming Web Services with .NET	Convention Ctr — Room 21
34	Efficient Architectures for Object-Oriented Component-Based Middleware	Marriott Hotel — Salon A
35	Extreme Programming Live!	Marriott Hotel — Florida Salon IV
36	Patterns and Architectures for J2EE Systems	Marriott Hotel — Florida Salon V
37	Refactoring: Improving the Design of Existing Code	Convention Ctr — Room 16

Half Day – Afternoon – 1:30 pm – 5:00 pm

38	Efficient Implementation of Object-Oriented Programming Languages	Marriott Hotel — Salon A
39	Making the Software Process Transparent by Using Intelligent Agents	Marriott Hotel — Florida Salon V
40	Surviving Object-Oriented Projects	Marriott Hotel — Florida Salon VI
41	Refactoring to Patterns	Convention Ctr — Room 21
42	How to Really Fail at Software Architecture	Convention Ctr — Room 19

Tuesday, 16 October 2001

Full Day – 10:00 am – 5:00 pm (short lunch break)

43	The Art of Writing Use Cases	Marriott Hotel — Meeting Room 12
44	Architecting Large Business Systems	Convention Ctr — Room 25

Half Day – Afternoon – 1:30 pm – 5:00 pm

45	The UML's Object Constraint Language (OCL) — Specifying Components	Marriott Hotel — Meeting Room 11
46	J2ME Design and Development Considerations	Convention Ctr — Room 16
47	Embedded Systems in C++ — C++ Idioms, Patterns, and Architecture for Constrained Systems	Convention Ctr — Room 5
48	Adaptive Object-Model Architecture: How to Build Systems That Can Dynamically Adapt to New Business Requirements	Convention Ctr — Room 15
49	No Stone Unturned: An Introduction to Test-First Programming	Convention Ctr — Room 3
50	Designing Software Architecture for Quality: The ADD Method	Convention Ctr — Room 22
51	Creating Responsive, Scalable Systems	Convention Ctr — Room 13
52	Leading Retrospectives on OO Projects: Looking Back to Move Forward	Convention Ctr — Room 14
53	Business Modeling with the UML	Marriott Hotel — Meeting Room 1
54	XP Meets UML: Development Processes for eTechnology	Convention Ctr — Room 23
55	Component-Based Design: A Complete Worked Example	Convention Ctr — Room 24
56	Developing Java Applications for Small Spaces	Convention Ctr — Room 6

Wednesday, 17 October 2001

Half Day – Afternoon – 1:30 pm – 5:00 pm

57	Patterns for Making Your Business Objects Persistent in a Relational Database World	Convention Ctr — Room 3
58	Creativity in Software Development	Marriott Hotel — Meeting Room 11
59	Architectures for Integrating Business Logic into J2EE	Convention Ctr — Room 22
60	Planning Agile Projects	Convention Ctr — Room 16
61	Designing Small Memory Software: Development Patterns for Systems with Limited Memory	Convention Ctr — Room 25
62	Reflection in Java	Convention Ctr — Room 5
63	Ruby for the Impatient	Marriott Hotel — Meeting Room 12
64	Realizing Extreme Programming as a Strategic Weapon for Innovation	Convention Ctr — Room 23
65	Advanced Extreme Programming Testing Techniques	Convention Ctr — Room 24
66	C++ Idioms	Convention Ctr — Room 13
67	Patterns and Techniques for Developing Performance Effective Enterprise Java Beans	Marriott Hotel — Meeting Room 1
68	Pair Programming: Experience the Difference	Convention Ctr — Room 15
69	Objects vs. The Web	Convention Ctr — Room 14
70	OPEN: A Flexible OO/CBD Process for Software-Intensive Systems Development	Marriott Hotel — Meeting Room 2

TUTORIALS BY TRACK

Fundamentals

1 A Brief Tour of Responsibility-Driven Design (Sun. Full Day)

2 Testing Object-Oriented Software Systems (Sun. Full Day)

4 Concepts of Object-Oriented Programming (Sun. Full Day)

8 Introduction to Writing Use Cases (Sun. AM)

12 Agile Methodologies (Sun. AM)

13 How to Manage the Change from COBOL to OOP (Sun. AM)

16 An Introduction to Design Patterns (Sun. AM)

.NET Technologies

15 XML, XSD, and SOAP as a Better Component Model (Sun. AM)

22 The .NET Framework: The Common Language Runtime and C# (Sun. PM)

33 Exposing and Consuming Web Services with .NET (Mon. AM)

Agile Methods and Extreme Programming

12 Agile Methodologies (Sun. AM)

20 Daily Builds Are for Wimps (Sun. PM)

32 Designing an Agile Methodology (Mon. AM)

35 Extreme Programming Live! (Mon. AM)

37 Refactoring: Improving the Design of Existing Code (Mon. AM)

49 No Stone Unturned: An Introduction to Test-First Programming (Tue. PM)

54 XP Meets UML: Development Processes for eTechnology (Tue. PM)

60 Planning Agile Projects (Wed. PM)

64 Realizing Extreme Programming as a Strategic Weapon for Innovation (Wed. PM)

65 Advanced Extreme Programming Testing Techniques (Wed. PM)

68 Pair Programming: Experience the Difference (Wed. PM)

Architecture

6 Inside High-Quality Software Architectures (Sun. AM)

14 Component and Service Architecture Modeling with UML (Sun. AM)

27 Software Architecture: It’s What’s Missing From OO Methodologies (Mon. Full Day)

29 Pattern-Oriented Software Architecture: Patterns for Concurrent and Networked Objects (Mon. Full Day)

34 Efficient Architectures for Object-Oriented Component-Based Middleware (Mon. AM)

Internet Technologies

15 XML, XSD, and SOAP as a Better Component Model (Sun. AM)

22 The .NET Framework: The Common Language Runtime and C# (Sun. PM)

33 Exposing and Consuming Web Services with .NET (Mon. AM)

36 Patterns and Architectures for J2EE Systems (Mon. AM)

55 Component-Based Design: A Complete Worked Example (Tue. PM)

59 Architectures for Integrating Business Logic into J2EE (Wed. PM)

69 Objects vs. The Web (Wed. PM)

Java Technologies

11 Introduction to Concurrent Object-Oriented Programming in Java (Sun. AM)

17 Producing GUIs with Java (Sun. AM)

18 Designing Concurrent Object-Oriented Programs in Java (Sun. PM)

19 Building Parsers with Java (Sun. PM)

26 Aspect-Oriented Programming with AspectJ™ (Mon. Full Day)

36 Patterns and Architectures for J2EE Systems (Mon. AM)

46 J2ME Design and Development Considerations (Tue. PM)

56 Developing Java Applications for Small Spaces (Tue. PM)

59 Architectures for Integrating Business Logic into J2EE (Wed. PM)

62 Reflection in Java (Wed. PM)

67 Patterns and Techniques for Developing Performance Effective Enterprise Java Beans (Wed. PM)

Languages (except Java)

15 XML, XSD, and SOAP as a Better Component Model (Sun. AM)

23 Garbage Collection (Sun. PM)

38 Efficient Implementation of Object-Oriented Programming Languages (Mon. PM)

47 Embedded Systems in C++ — C++ Idioms, Patterns, and Architecture for Constrained Systems (Tue. PM)

63 Ruby for the Impatient (Wed. PM)

66 C++ Idioms (Wed. PM)

Meta-level and Reflective

26 Aspect-Oriented Programming with AspectJ™ (Mon. Full Day)

48 Adaptive Object-Model Architecture: How to Build Systems That Can Dynamically Adapt to New Business Requirements (Tue. PM)

62 Reflection in Java (Wed. PM)

Middleware

34 Efficient Architectures for Object-Oriented Component-Based Middleware (Mon. AM)

36 Patterns and Architectures for J2EE Systems (Mon. AM)

59 Architectures for Integrating Business Logic into J2EE (Wed. PM)

67 Patterns and Techniques for Developing Performance Effective Enterprise Java Beans (Wed. PM)

Patterns

7 Dungeons and Patterns! (Sun. AM)

10 Introducing Patterns (or Any New Idea) into Organizations (Sun. AM)

16 An Introduction to Design Patterns (Sun. AM)

21 Designing with Patterns (Sun. PM)

25 Fractal Patterns and Frameworks in UML — Towards UML 2.0? (Sun. PM)

29 Pattern-Oriented Software Architecture: Patterns for Concurrent and Networked Objects (Mon. Full Day)

31 Patterns at Work (Mon. AM)

36 Patterns and Architectures for J2EE Systems (Mon. AM)

41 Refactoring to Patterns (Mon. PM)

57 Patterns for Making Your Business Objects Persistent in a Relational Database World (Weds. PM)

67 Patterns and Techniques for Developing Performance Effective Enterprise Java Beans (Wed. PM)

People, Process, and Project Management

10 Introducing Patterns (or Any New Idea) into Organizations (Sun. AM)

39 Making the Software Process Transparent by Using Intelligent Agents (Mon. PM)

40 Surviving Object-Oriented Projects (Mon. PM)

52 Leading Retrospectives on OO Projects: Looking Back to Move Forward (Tue. PM)

58 Creativity in Software Development (Wed. PM)

60 Planning Agile Projects (Wed. PM)

70 OPEN: A Flexible OO/CBD Process for Software-Intensive Systems Development (Wed. PM)

Refactoring and Reengineering

30 Object-Oriented Reengineering (Mon. Full Day)

37 Refactoring: Improving the Design of Existing Code (Mon. AM)

41 Refactoring to Patterns (Mon. PM)

Requirements Analysis

8 Introduction to Writing Use Cases (Sun. AM)

24 Advanced Use Case Writing (Sun. PM)

28 Improving Your Use Cases (Mon. Full Day)

43 The Art of Writing Use Cases (Tue. Full Day)

53 Business Modeling with the UML (Tue. PM)

Small or Mobile

46 J2ME Design and Development Considerations (Tue. PM)

47 Embedded Systems in C++ — C++ Idioms, Patterns, and Architecture for Constrained Systems (Tue. PM)

56 Developing Java Applications for Small Spaces (Tue. PM)

61 Designing Small Memory Software: Development Patterns for Systems with Limited Memory (Wed. PM)

Testing

2 Testing Object-Oriented Software Systems (Sun. AM)

20 Daily Builds Are for Wimps (Sun. PM)

49 No Stone Unturned: An Introduction to Test-First Programming (Tue. PM)

65 Advanced Extreme Programming Testing Techniques (Wed. PM)

UML

4 Concepts of Object-Oriented Programming (Sun. Full Day)

14 Component and Service Architecture Modeling with UML (Sun. AM)

25 Fractal Patterns and Frameworks in UML — Towards UML 2.0? (Sun. PM)

45 The UML's Object Constraint Language (OCL) — Specifying Components (Tue. PM)

53 Business Modeling with the UML (Tue. PM)

54 XP Meets UML: Development Processes for eTechnology (Tue. PM)

Usability and UI

3 Usage-Centered Design: An Agile Model-Driven Process for Object-Oriented User Interface Design (Sun. Full Day)

5 Lo-Fi Design Strategies for Creating Highly Usable Object-Oriented User Interfaces (Sun. Full Day)

9 Object-Oriented Design of Human-Computer Interaction (Sun. AM)

17 Producing GUIs with Java (Sun. AM)

1 A Brief Tour of Responsibility-Driven Design

Sunday, Full Day
Convention Ctr — Room 20

Rebecca Wirfs-Brock,
Wirfs-Brock Associates

Alan McKean,
Wirfs-Brock Associates

Responsibility-Driven Design offers practical advice for designing, implementing, and redesigning with responsibilities. In a responsibility-based model, objects play specific roles and occupy well-known positions in the application architecture. Each object is accountable for a specific portion of the work. They collaborate in clearly defined ways, contracting with each other to fulfill the larger goals of the application. By creating a “community of objects,” assigning specific responsibilities to each, we build a collaborative model of our application. Objects are more than simple bundles of logic and data ... they are service-providers, information-holders, structurers, coordinators, controllers, and interfacers to the outside world! Each must know and do its part! Thinking in these terms enables you to build powerful, flexible applications.

This tutorial, which includes new material from our forthcoming book, will be an example-based tour of Responsibility-Driven Design. It presents our latest innovations and practical techniques. Topics include: finding and evaluating the qualities of candidate design objects, mapping roles to classes and interfaces, strategies for assigning object responsibilities, deciding on the control style of an application, effective ways to describe collaborations, how to organize a design by specifying contractual relations and obligations, and techniques for increasing a design’s flexibility and clarity.

Attendee Background: Participants should be familiar with object concepts and be looking for practical techniques, guidelines and a design process that emphasizes modeling the behavioral aspects of a software system.

Presenters: *Rebecca Wirfs-Brock is president of Wirfs-Brock Associates, a firm specializing in the transfer of object analysis and design expertise to organizations and individuals through training, mentoring, and consulting. Rebecca has been involved with object technology since its infancy. She is the inventor of the set of development practices known as Responsibility-Driven Design. From development on the Tektronix implementation of Smalltalk in the early 1980s, through years of development and training experience, she is recognized as one of only a few knowledgeable and influential practitioners of object-oriented design. She spent 17 years as a Software Engineer at Tektronix, where she managed the first commercial Smalltalk effort and was the technical lead for the development of Color Smalltalk. Recently, she has authored use cases for a telecommunications framework and an online banking system and has mentored teams in use case writing, design, architecture and managing incremental, iterative object-technology projects. She practices what she teaches!*

Alan McKean is Vice President and Director of Educational Services at Wirfs-Brock Associates. Alan McKean has devoted most of his career applying principles of design and adult learning to find better ways to communicate technical and design information. A student of R. Buckminster Fuller and a graduate of the University of Oregon with a Masters in Computer Science, he specializes in system architecture and object-oriented design and programming. Alan has delivered over a hundred workshops on designing and programming object-oriented software during his 10+ years at Instantiations, Digitalk, and Wirfs-Brock Associates. Alan was a keynote speaker at the OOPSLA Educator’s Symposium in 1995 and has been invited to speak at this year’s Educators’ Symposium. Prior to his training experience, Alan was a Director at Dynamix, Inc., a computer game company, where he invented and developed a toolset for synchronizing animated images with actors’ recorded voices and a suite of Smalltalk-based tools for managing computer game sound effects and music.

2 Testing Object-Oriented Software Systems

Sunday, Full Day
Convention Ctr — Room 13

John McGregor, *Clemson University*

The use of object-oriented software construction techniques and iterative, incremental processes influence the organization, structure, and execution of testing activities in a project. The techniques presented in the tutorial are intended to provide a scalable process that can be tailored to the size of a project and the criticality of the type of application. The comprehensive test plan, presented in the tutorial, integrates the construction process and the testing process to produce an efficient and complete development process.

This tutorial is divided into three parts: (1) specific techniques supported by small examples to illustrate specific testing algorithms, (2) techniques for testing system level models using enhanced inspection and review procedures and (3) a process for system testing presented within the context of a complete testing process for object-oriented systems.

Instructional objectives: The participant will be able to define test cases from use cases. The participant will be able to build test suites that reuse test cases from related uses. The participant will be able to adapt a generic testing process to his/her corporate and project environments. The participant will be able to prioritize tests based on information in the use cases.
Lecture/discussion: 70% Exercises: 30%

Attendee Background: Attendees should be familiar with object-oriented concepts and at least one object-oriented programming language. It will be helpful if the attendee is familiar with basic software testing techniques to the level gained by practical experience.

Presenter: *Dr. John D. McGregor is a senior partner in Korson-McGregor and an associate professor of computer science at Clemson University. Dr. McGregor has conducted funded research for organizations such as the National Science Foundation, DARPA, IBM, and AT&T. Dr. McGregor has developed testing techniques for object-oriented software and developed custom testing processes for a variety of companies. Dr. McGregor is co-author of* Object-oriented Software Development: Engineering Software for Reuse *(Van Nostrand Reinhold) and is co-author of* A Practical Guide to Testing Object-Oriented Software *(Addison-Wesley, 2001). He writes a column on Testing and Quality for the Journal of Object-oriented Programming (JOOP) published by SIGS Publishing. He has published numerous articles on software development focusing on design and quality issues. Dr. McGregor's research interests include software engineering specifically in the areas of process definition, design quality, testing and measurement. Dr. McGregor has given tutorials for several years at OOPSLA and ECOOP. He presents to 10 - 12 conferences per year as well as offering industrial courses to demanding technical audiences.*

3 Usage-Centered Design: An Agile Model-Driven Process for Object-Oriented User Interface Design

Sunday, Full Day
Convention Ctr — Room 25

Larry Constantine,
University of Technology, Sydney; Constantine & Lockwood, Ltd.

James Noble,
Victoria University of Wellington

Agile processes and lightweight methods are increasingly popular but share with their ponderous predecessors an inattention to usability and user interface design. Usage-centered design is a proven industrial-strength process for designing highly usable and innovative solutions to interaction-intensive problems. It has been applied with marked success to projects ranging from automation programming tools to classroom information systems to e-commerce Web sites. Through alternating lectures, applied exercises, and discussions, this tutorial introduces a streamlined process for quickly and efficiently designing improved user interfaces supported by robust internal software. Essential use cases—a simplified, generalized, and abstract improvement on conventional use cases—are used to model tasks and to guide the design of user interfaces that effectively support the real needs of users. Through actual application to a compressed but representative case study problem, participants will learn, how to employ ordinary index cards and accelerated modeling sessions to quickly understand and prioritize user roles and user tasks and to organize the needed user interface contents. The emphasis will be on modeling techniques that yield the greatest payoff from the least effort in design, techniques that are well-suited to producing world-class designs through iterative, time-boxed development within compressed release cycles.

Attendee Background: Some experience with use cases and familiarity with the basic concepts and techniques of object-orientation are assumed. Understanding of the basic principles of usability and user interface design would be helpful but is not mandatory.

Presenters: *Larry Constantine is Adjunct Professor of Computing Sciences, University of Technology, Sydney, and Director of Research and Development for Constantine & Lockwood, Ltd., the international design and consulting firm he co-founded. A pioneer of modern software engineering practice and a recognized authority on the human side of software, he is the co-inventor of essential use cases and usage-centered design. He has conducted hundreds of seminars and tutorials in nineteen countries and his publications include sixteen books and nearly 150 papers.*

Dr. James Noble is a lecturer at the Victoria University of Wellington, New Zealand, and a Consulting Associate with Constantine & Lockwood, Ltd. He is the co-author of Small Memory Software: Patterns for Systems with Limited Memory *(Addison-Wesley 2000), and numerous published papers on software design, user interface design, and design patterns. He has extensive lecturing and teaching experience, including tutorials at OOPSLA, TOOLS Pacific, and OzCHI.*

4 Concepts of Object-Oriented Programming

Sunday, Full Day
Convention Ctr — Room 22

Raimund Ege,
Florida International University

This tutorial defines and teaches the basic object-oriented concepts, illustrates their advantages, and introduces the components and features of object-oriented programming languages and development environments. The tutorial enables an attendee to make an informed decision about what language/environment will best serve his/her software development needs. The tutorial has two major parts: Part 1 discusses in detail all object-oriented concepts and uses UML and Java to illustrate them. The focus will be on a precise non-confusing definition of the core concepts and terminology, such as object, instance, class, interface, attribute, service, message passing, hierarchy, inheritance, polymorphism, late binding, memory management, access specification, and packaging. Part 2 then compares the major object-oriented programming languages: C++, Java, Smalltalk, and others. The comparison is done with a double focus: (1) how does the language support and enforce the concepts, and (2) how does the language help software development (to that effect, I have a small case study program, that will be solved in all languages). Whether and how each language supports advanced concepts, like multiple and repeated inheritance, genericity, interfaces, is discussed in detail.

Attendee Background: Attendees are software professionals who are interested in learning the fundamental concepts and advantages of object- oriented programming and how to apply them in a modern software development environment. No previous knowledge of object-oriented concepts is assumed. The attendees should have a fundamental background in computer science and/or computer programming.

Presenter: *Raimund K. Ege is an Associate Professor of Computer Science at the Florida International University, Miami. He is author of* Programming in an Object-Oriented Environment *(Academic Press, 1992), and* Object-Oriented Programming with C++ *(Academic Press, 1994). He is an active researcher in the area of object-oriented concepts, and their application to programming, user interfaces, databases, simulation, and software engineering. He has presented numerous successful tutorials at major conferences (OOPSLA, ECOOP, TOOLS). The tutorials were consistently rated highest and won praise from organizers and attendees.*

5 Lo-Fi Design Strategies for Creating Highly Usable Object-Oriented User Interfaces

Sunday, Full Day
Convention Ctr — Room 24

Luke Hohmann, *Independent Consultant*

Even if a software development project creates extensive and complete object-oriented analysis and design models, it will still be perceived as a failure if the user interface is poorly constructed. To be perceived as truly successful, the system must meet the needs of the user. This is best done by designing a user interface that is effective, appealing, intuitive, and easy to learn. In other words, you must create a highly usable object-oriented user interface. Participants of this tutorial will learn how to design highly usable object-oriented user interfaces using the latest in lo-fi prototyping techniques by creating such designs in small groups. Upon completion of this tutorial, participants will be able to: define usability and its relation to object technology; define the role of lo-fi and hi-fi prototyping; design and test lo-fi prototypes according to timeless principles of usability; and implement lo-fi prototypes in a manner that is consistent with the underlying domain model.

Attendee Background: Participants should have a basic knowledge of object-oriented analysis and design, use cases, and scenarios; and be involved in the design and implementation of a project utilizing a graphical user interface. Knowledge of a specific object-oriented programming language is not required.

Presenter: *Luke Hohmann is an independent consultant, committed to coaching his clients to greater levels of performance. Mr. Hohmann is author of* Journey of the Software Professional: A Sociology of Software Development *(Prentice-Hall), as well as numerous articles on software engineering management. A skilled instructor and speaker, Mr. Hohmann has been invited to many conferences. Mr. Hohmann can be contacted through e-mail at LukeHohmann@yahoo.com.*

6 Inside High-Quality Software Architectures

Sunday, Morning
Convention Ctr — Room 18

Frank Buschmann,
Siemens AG, Germany

In this tutorial we examine the secrets of high-quality software architectures: how are they specified, what are their properties, and how are they implemented. The result is a set of principles consisting of methodological steps, concrete design goals that help constructing and implementing software architectures successfully as well as a set of properties that such architectures expose. To illustrate these steps, goals, and properties we use a running example from the industrial automation domain.

Attendee Background: Sound knowledge in Object Technology.

Presenter: *Frank Buschmann is a software engineer at Siemens Corporate Technology in Munich, Germany. His research interests include Object Technology, Application Frameworks, and specifically Patterns. Frank has been involved in several concrete industrial software development projects. Frank is co-author of* Pattern-Oriented Software Architecture — A System of Patterns.

7 Dungeons and Patterns!

Sunday, Morning
Marriott Hotel — Meeting Room 11

Steve Metsker, *Capital One*
William Wake, *Capital One*

“Dungeons and Patterns” is a hands-on tutorial for exploring and learning about design patterns. Learning design patterns will help you become a more powerful object-oriented developer. Unfortunately, a single reading of *Design Patterns* won’t magically implant design pattern recognition skills in your brain. You have to learn patterns by doing, which means you have to start applying patterns before you can apply them—a monstrous dilemma! The solution is to practice patterns in a playful setting where slip-ups are profitable and painless. In this tutorial you will join an adventure with others at your table, seeking the treasure of patterns hidden within a dungeon replete with structural traps, motivational pitfalls, and implementation monsters. These barriers will succumb to strong collaboration with your table-mates and will yield to effective application of the patterns in *Design Patterns*. Dungeons and Patterns will deepen your understanding and strengthen your skills at recognizing and applying design patterns.

Attendee Background: Attendees should have tried reading *Design Patterns* at least once. No experience with role-playing games is required.

Presenters: *Steve Metsker is a researcher and author who explores and writes about ways to expand the abilities of developers. Steve’s articles have explained how to maintain relational integrity in object models, how to solve logic puzzles in Java, and how the conception of “object” differs between Plato and the OO languages. Steve’s most recent publication is the book,* Building Parsers with Java.

William Wake is interested in XP, patterns, human-computer interaction, and information retrieval. He is the author of, Extreme Programming Explored, *and the inventor of the Test-First Stoplight and the XP Programmer’s Cube.*

8 Introduction to Writing Use Cases

Sunday, Morning
Marriott Hotel — Florida Salon V
Alistair Cockburn,
Humans and Technology

A use case is a way of describing the required behavior of a system, centered around what the system offers its users. Use cases are easy to read, and simple in concept, but surprisingly tricky to write. This tutorial is for the beginning use case writer, to see what a use case looks like, the basics of how to write one, and how to organize people to write, review and use them. The tutorial will be part lecture, and part hands-on exercises. Attendees will brainstorm a list of use cases for a system, write a main scenario, and uncover failure scenarios. The exercises are designed to allow the attendees to practice the writing skills, and discover where use cases get difficult. At the end of the tutorial, the attendee will have the basic vocabulary of use cases, will have seen examples of good and bad ones, and will have experienced the variations in writing that will show up in real use cases.

Attendee Background: This tutorial is for people just beginning to write or consider use cases. No particular background is required.

Presenter: *Alistair Cockburn is a highly regarded instructor and is known as one of the premier experts on use cases. His book,* Writing Effective Use Cases, *set the standard in the area and was nominated for Software Development's Jolt book award in 2001. Alistair has taught use case writing since 1994, and has also acted as consultant on project management, object-oriented design, and methodology to the Central Bank of Norway, the IBM Consulting Group, and the First Rand Bank of South Africa. Materials that support his workshops can be found at http://members.aol.com/acockburn, http://crystalmethodologies.org and http://usecases.org.*

9 Object-Oriented Design of Human-Computer Interaction

Sunday, Morning
Convention Ctr — Room 15

Mary Beth Rosson, *Virginia Tech*

Object-oriented design methods are claimed to reduce the gap between the problem domain and the software system. This has important implications for the design of human-computer interaction: A software model that mirrors the real world should reduce the cognitive distance between how a system works and the mental models that users build to use and understand the software. This tutorial explores how to apply object-oriented thinking to the design of human-computer interaction. The methods discussed are part of a general scenario-based framework for usability engineering. In this framework, a scenario is a narrative of the goals, actions, and reactions of actors pursuing goals with an interactive system. The tutorial presents scenario-based techniques for developing and integrating object-oriented views of requirements, activity design, user interface design, and usability evaluation. Throughout, design rationale is captured, serving to raise and discuss the implications that object-oriented concepts will have for the user experience. Concepts and techniques are introduced briefly, then illustrated with examples. The format will be lecture interspersed with presentation and discussion of the examples.

Attendee Background: General knowledge of object-oriented concepts, interest in use-centered design of interactive systems.

Presenter: *Mary Beth Rosson is an associate professor of computer science at Virginia Tech. She is an expert in human-computer interaction (HCI), and the author of numerous research papers on the relationship between HCI and object-oriented design. Rosson has given research papers and tutorials at the ACM SIGCHI, OOPSLA, and ECOOP conferences and has served in many leadership positions in SIGCHI and SIGPLAN. Most recently, she was General Chair of OOPSLA 2000.*

10 Introducing Patterns (or Any New Idea) into Organizations

Sunday, Morning
Convention Ctr — Room 14

Mary Lynn Manns,
University of North Carolina at Asheville

Linda Rising, *Independent Consultant*

Many people who have attended OOPSLA or other conferences find new ideas that they wish to take back to their organizations, but then struggle to make something happen. This tutorial will help participants understand what successful change agents have learned while attempting to introduce new ideas into their organizations. The lessons learned have been documented in an evolving pattern language titled Introducing Patterns (or any new idea) into Organizations. This session will examine the problems and solutions documented in this language through the simulation of attempts to introduce a new idea, such as patterns, into an organization.

Attendee Background: Anyone in the software business who is trying to introduce patterns (or any new idea) into an organization will find this tutorial useful. We assume that attendees are familiar with the notion of patterns.

Presenters: *Mary Lynn Manns is on the faculty at the University of North Carolina at Asheville. During the past three years, she has studied the issues in introducing patterns into organizations. She has also taught patterns in industry and done numerous other presentations on the topic.*

Linda Rising has a Ph.D. from Arizona State University in the area of object-based design metrics. Her background includes university teaching experience as well as work in industry in the areas of telecommunications, avionics, and strategic weapons systems. She has been working with object technologies since 1983. She is the editor of A Patterns Handbook, The Pattern Almanac 2000, and Design Patterns in Communication Software.

11 Introduction to Concurrent Object-Oriented Programming in Java

Sunday, Morning
Convention Ctr — Room 16

David Holmes, *DSTC Pty Ltd.*

Doug Lea,
State University of New York (SUNY) at Oswego

Concurrent programming has mostly been the domain of systems programmers rather than application developers, but Java's support of concurrency has enticed many to try their hand at concurrent applications. However concurrent programming poses many traps for the unwary. This tutorial demonstrates various design patterns and techniques for constructing concurrent applications in Java and for managing that concurrency. On the language side we look at Java's mechanisms to support concurrent programming. On the design side we look at object structures and design rules that can successfully resolve the competing forces (safety, liveness, efficiency, coordination, reusability) present in concurrent software design problems. Participants will acquire comprehensive knowledge of the concurrency support provided by the Java language and core classes, as well as insight into some threading issues within the Java libraries. They will be exposed to a range of design approaches to assist them in developing safe, concurrent applications in Java and other object-oriented languages.

Attendee Background: This tutorial targets anyone involved, or planning to get involved, in the development of concurrent object-oriented applications. It is assumed that the attendee is familiar with basic OO concepts and has a working knowledge of the Java programming language.

Presenters: *David Holmes is a Senior Research Scientist at the Cooperative Research Centre for Enterprise Distributed Systems Technology (DSTC Pty, Ltd.), in Brisbane, Australia. He completed his Ph.D. in the area of synchronization within object-oriented systems and has been involved in concurrent programming for a number of years. He is a co-author of the third edition of the Java Series book,* The Java Programming Language.

Doug Lea is a professor of Computer Science at the State University of New York at Oswego. He is author of the Java Series book, Concurrent Programming in Java: Design Principles and Patterns, *co-author of the book,* Object-Oriented System Development, *and the author of several widely used software packages, as well as articles and reports on object-oriented software development.*

12 Agile Methodologies

Sunday, Morning
Convention Ctr — Room 19

Jim Highsmith,
Information Architects, Inc.

In the past two years, a wide range of publications (Software Development, IEEE Software, Cutter IT Journal, Software Testing and Quality Engineering, and even the Economist) have published articles on what Martin Fowler calls the New Methodologies. There has been a rapidly rising interest in these new approaches to software development such as Extreme Programming, Scrum, Adaptive Software Development, Feature-Driven Development, and Dynamic Systems Development Methodology. Furthermore, scores of organizations have developed their own "lighter" approach to building software. Recently, representatives from each of the New Methodologies met, formed the Agile Alliance, and developed common purpose and principles to help others think about software development, methodologies, and organizations, in new "more agile" ways. This workshop, given by Jim Highsmith, developer of one of the Agile Methodologies (Adaptive Software Development), and one of the authors of the *Manifesto for Agile Software Development*, addresses key questions: What are Agile Methodologies? What problem domains do Agile Methodologies address? What are the common principles behind Agile Methodologies? What are the similarities and differences between the various Agile Methodologies?

Attendee Background: The tutorial is targeted at software development managers, project managers, and team leaders. Basic project management knowledge will be helpful.

Presenter: *Jim Highsmith is director of Cutter Consortium's e-Project Management Practice, president of Information Architects, Inc., and author of* Adaptive Software Development: A Collaborative Approach to Managing Complex Systems *(Dorset House, 2000). He has 30 years experience as a consultant, software developer, manager, and writer. Jim has published dozens of articles in major industry publications and his ideas about project management in the Internet era were featured in recent issues of ComputerWorld and the Economic Times in India. In the last ten years, he has worked with both IT organizations and software companies in the US, Europe, Canada, South Africa, Australia, Japan, India, and New Zealand to help them adapt to the accelerated pace of development in increasingly complex, uncertain environments.*

13 How to Manage the Change from COBOL to OOP

Sunday, Morning
Marriott Hotel — Salon A

Markus Knasmüller, *BMD Systemhaus*

After solving the problems Y2K and Euro, the last big challenges for Cobol-programmers are over. Therefore most of them have to look for new fields of activity, but these are combined with new programming techniques like object-oriented programming. However, introducing object-oriented programming to old-style programmers is a rather hard task. This tutorial shows how this job was done at BMD Steyr, Austrians leading producer of accountancy software. It is a perfect support for everybody who wants to introduce or teach object-oriented programming. After presenting background information why one should change and how this change should be accompanied, a special course for former Cobol programmers is presented. Experiences, as well as tips and tricks, will round up the presentation.

Attendee Background: The participants should have basic knowledge of traditional programming languages like Cobol or PL/I and should have the wish to change to object-oriented programming.

Presenter: *Markus Knasmüller is head of the software department at BMD Systemhaus, Austrians leading producer of accountancy software. In this position he was responsible for the change of 50 programmers and 5 millions lines of code from COBOL to OOP. He is author of various research papers and books (for example:* From COBOL to OOP, dpunkt, 2001*) and has experience in teaching object-oriented programming at the university as well as in industry. Markus holds a Ph.D. in computer science and a degree in management information systems.*

14 Component and Service Architecture Modeling with UML

Sunday, Morning
Convention Ctr — Room 21

Desmond D'Souza, *Kinetium*

The word "architecture" often bestows instant importance to pretty powerpoint drawings and vague hand waving. And while e-business demands flexible configuration of components and web-services, those components and services will only plug together if they conform to shared "pluggable" architecture standards. Believing "architecture keeps designers from needless creativity," we outline a clear definition of architecture and architectural style based on UML packages, patterns, and refinement. Elements and rules of an architectural style are separated from the designs which use them, and component architectures use an abstract component-connector model.

Attendee Background: Attendees should be familiar with the UML.

Presenter: *Desmond D'Souza is founder and president of Kinetium. He is co-author of* Objects, Components, and Frameworks With UML: The Catalysis Approach *(Addison Wesley 1998), and a respected speaker internationally. He was previously senior vice president of component-based development at Platinum Technology and at Computer Associates. Kinetium provides client solutions that leverage shareable architectures for model-driven development and integration of systems, with a current focus on light-weight modeling architecture and methods. Desmond can be reached at dsouzad@acm.org.*

15 XML, XSD, and SOAP as a Better Component Model

Sunday, Morning
Marriott Hotel — Florida Salon VI

Don Box, *DevelopMentor*

The Simple Object Access Protocol (SOAP) is an XML-based protocol for exposing servers, services, components or objects over the web. SOAP codifies the use of existing technologies such as XML, XML Schema Definition (XSD) language, and HTTP to allow code to be accessed in an interoperable and Internet-friendly fashion. This tutorial covers the following topics: The XML Protocol Stack, the XML Schema Language, HTTP Myths vs. Reality, SOAP Encoding, SOAP Framing, SOAP and Extensibility, Architecture of a SOAP runtime, and Architecture of a SOAP application

Attendee Background: Attendees should be familiar with the basics of object-oriented programming and moderately comfortable with some RPC or messaging based technology such as CORBA, DCOM or RMI.

Presenter: *Don Box is a co-founder of DevelopMentor, a developer services company that provides education and support to the software industry at large. Don's research interests include component software integration, programming for concurrency, and XML-based serialization and metadata protocols. Don is a series editor at Addison Wesley and is the author of* Essential COM, *and a co-author of* Effective COM, *and* Essential XML, *all from Addison Wesley. Don is a contributing editor and columnist at Microsoft Systems Journal (now called MSDN Magazine) and an occasional contributor to XML.com. Don is also a co-author of the Simple Object Access Protocol specification and a member of the W3C Schemas Working Group. Don has a Master's Degree in Computer Science from the University of California at Irvine.*

16 An Introduction to Design Patterns

Sunday, Morning
Marriott Hotel — Florida Salon IV

John Vlissides, *IBM T.J. Watson Research*

Designing object-oriented software is hard, and designing reusable object-oriented software is even harder. Experience shows that many object-oriented systems exhibit recurring structures or “design patterns” of communicating and collaborating objects that promote extensibility, flexibility, and reusability. This tutorial describes a set of fundamental design patterns and, through a design scenario, demonstrates how to build reusable object-oriented software with them. The tutorial covers the roles design patterns play in the object-oriented development process: how they provide a common vocabulary, reduce system complexity, and how they act as reusable architectural elements that contribute to an overall system architecture.

Attendee Background: Attendees should understand basic object-oriented concepts, like polymorphism and type versus interface inheritance, and should have had some experience designing object-oriented systems. No prior knowledge of design patterns is required. Familiarity with Java is recommended.

Presenter: *John Vlissides is a member of the research staff at the IBM T.J. Watson Research Center in Hawthorne, NY. He has practiced object-oriented technology for over a decade as a designer, implementer, researcher, lecturer, and consultant. John is author of* Pattern Hatching, *co-author of* Design Patterns *and* Object-Oriented Application Frameworks, *and co-editor of* Pattern Languages of Program Design 2. *He has published many articles and technical papers on object-oriented themes in general and design patterns in particular. John is a columnist for Java Report and serves as Consulting Editor of Addison-Wesley’s Software Patterns Series. He has a Ph.D. in Electrical Engineering from Stanford University.*

17 Producing GUIs with Java

Sunday, Morning
Marriott Hotel — Meeting Room 12

Fintan Culwin,
South Bank University: London

The Java Foundation Classes supply a collection of user interface components. This tutorial attempts to introduce a representative selection of the most common and useful of them, showing how they can be combined to produce effective user interfaces. To accomplish this efficiently it is necessary to start with a representation of the required behavior of the interface and derive the detailed design from it.

Objectives:

- provide an introductory overview of the widgets supplied by the JFC;
- introduce the usability heuristics and style guides that can be employed in the detailed design of user interfaces;
- show how State Transition Diagrams (STDs) can describe the required behavior of an interface;
- introduce the Java event dispatch/ listener model;
- introduce and illustrate layout management policies;
- illustrate the use of STD, Class, Instance, Interface Layout and Object Interaction diagram notations;
- illustrate the realization of detailed three-layer designs in Java;
- introduce the resource management techniques, which improve the presentation of an interface.

Attendee Background: An intermediate level tutorial for attendees who have an initial familiarity with OO concepts and wish to develop further understanding in the context of GUI construction. Most of the exposition is at the source code level.

Presenter: *Fintan Culwin is a Reader in Software Engineering Education at South Bank University: London specializing in Software Engineering and HCI, particularly in the integration of usability considerations in the earliest stages of production processes. He has published five books, including two on Java, and is currently completing a sixth on the JFC. He has published extensively on Internet issues and has presented sessions on the Web and Java at a series of international conferences including: SIGCSE, BCS HCI, ITiCSE, CHI and OOPSLA.*

18 Designing Concurrent Object-Oriented Programs in Java

Sunday, Afternoon
Convention Ctr — Room 16

Doug Lea,
State University of New York (SUNY) at Oswego

David Holmes, *DSTC Pty Ltd.*

Concurrent programming has mostly been the domain of systems programmers rather than application developers, but Java's support of concurrency has enticed many to try their hand at concurrent applications. Effectively creating and managing concurrency within an application poses many design choices and trade-offs. This tutorial looks at more advanced issues in designing concurrent applications. It describes mechanisms for introducing concurrency into applications (threads, message-passing, asynchronous calls) and different models for application architectures, such as data-flow and event-driven designs. The tutorial also shows how concurrency controls can be abstracted into reusable support classes, and finally discusses how concurrent components and applications should be documented. Participants will learn how concurrent applications can be structured in different ways and how different mechanisms can be used to effect concurrent behavior. They will be exposed to a range of design patterns and techniques for introducing and managing concurrency within their applications and how to create reusable concurrency abstractions.

Attendee Background: This tutorial targets anyone involved, or planning to get involved, in the development of concurrent object-oriented applications. It is expected that the attendee is very familiar with OO concepts and the Java language, and has a good working knowledge of Java's concurrency mechanisms.

Presenters: *Doug Lea is a professor of Computer Science at the State University of New York at Oswego. He is author of the Java Series book,* Concurrent Programming in Java: Design Principles and Patterns, *co-author of the book,* Object-Oriented System Development, *and the author of several widely used software packages, as well as articles and reports on object-oriented software development.*

David Holmes is a Senior Research Scientist at the Cooperative Research Centre for Enterprise Distributed Systems Technology (DSTC Pty, Ltd.), in Brisbane, Australia. He completed his Ph.D. in the area of synchronization within object-oriented systems and has been involved in concurrent programming for a number of years. He is a co-author of the third edition of the Java Series book, The Java Programming Language.

19 Building Parsers with Java

Sunday, Afternoon
Marriott Hotel — Meeting Room 11
Steve Metsker, *Capital One*

By learning to write parsers you learn to bridge the gap between computers and the users of your language. You can nestle a new language into any niche, defining how your users interact with computers using text. This workshop introduces the basics of building a language from Sequence, Alternation, and Repetition objects. With these three objects, you can create any syntax-free language. In this session you will spend a large portion of class time writing parsers, using the tools this workshop introduces. You will learn when to create an XML-based language and when to use Java. You will also learn how to design a language and how to generate a working parser from this design.

Attendee Background: Attendees should be experienced Java developers.

Presenter: *Steve Metsker is a researcher and author who explores and writes about ways to expand the abilities of developers. Steve's articles have explained how to maintain relational integrity in object models, how to solve logic puzzles in Java, and how the conception of "object" differs between Plato and the OO languages. Steve's most recent publication is the book,* Building Parsers with Java.

20 Daily Builds Are for Wimps

Sunday, Afternoon
Convention Ctr — Room 15
Michael Two, *Thoughtworks*

Over the last couple of years we've been building a large J2EE application. One of the biggest lessons we've learned is to follow the Extreme Programming (XP) approach to Continuous Integration of our 250+ KLOC system. This session focuses on how we turned a project that needed days of fiddling around to get a build into a project that delivers a fully tested build every hour. We'll go through our automated testing process using JUnit and Excel based acceptance test driver. We will also talk about a set of open source tools we have developed to automate build processes using Ant. We will explore the code for the tools that connect to the source code control system, run code generators, compile, deploy, test and publish the build.

Attendee Background: Participants should be familiar with Java and basic XML syntax.

Presenter: *Michael is a developer and XP advocate at ThoughtWorks working on a very large J2EE application using XP. After studying physics in college he chose a career in software once he realized that staying up all night in an office is more fun than staying up all night in a lab. Michael wrote labor schedule optimization software in C++ before joining Thoughtworks in 1999.*

21 Designing with Patterns

Sunday, Afternoon
Marriott Hotel — Florida Salon IV

John Vlissides, *IBM T.J. Watson Research*

Design patterns are making the transition from curiosity to familiarity. Now that many people know *what* they are, they want to know *how* best to apply them. This tutorial shows how to leverage patterns in the software design process. It reveals the thinking behind pattern application—including when *not* to use a seemingly applicable pattern. It shows how the right patterns can improve a design and how the wrong patterns can degrade one. Students thus learn to apply design patterns to maximum benefit.

Attendee Background: Attendees should be well-grounded in object technology and should be familiar with the design patterns in *Design Patterns: Elements of Reusable Object-Oriented Software*, by Gamma, et al. Familiarity with Java is recommended.

Presenter: *John Vlissides is a member of the research staff at the IBM T.J. Watson Research Center in Hawthorne, NY. He has practiced object-oriented technology for over a decade as a designer, implementer, researcher, lecturer, and consultant. John is author of* Pattern Hatching, *co-author of* Design Patterns *and* Object-Oriented Application Frameworks, *and co-editor of* Pattern Languages of Program Design 2. *He has published many articles and technical papers on object-oriented themes in general and design patterns in particular. John is a columnist for Java Report and serves as Consulting Editor of Addison-Wesley's Software Patterns Series. He has a Ph.D. in Electrical Engineering from Stanford University.*

22 The .NET Framework: The Common Language Runtime and C#

Sunday, Afternoon
Marriott Hotel — Florida Salon VI

Don Box, *DevelopMentor*

The Common Language Runtime is a new implementation of many existing ideas in component technology. The CLR is a type-centric multi-paradigm component model and runtime that supports object-oriented programming, interface-based programming, and aspect-oriented programming. This tutorial covers the following topics: Managed Types Vs. Unmanaged Types, Managed Execution Vs. Unmanaged Execution, Programming Language vs. The Runtime, The CLR Type System, Loading and Linking, Runtime Type Management, Context and Remoting, and Web Services

Attendee Background: Attendees should be familiar with the basics of object-oriented programming and moderately comfortable with systems-programming issues such as thread and process management.

Presenter: *Don Box is a co-founder of DevelopMentor, a developer services company that provides education and support to the software industry at large. Don's research interests include component software integration, programming for concurrency, and XML-based serialization and metadata protocols. Don is a series editor at Addison Wesley and is the author of* Essential COM, *and a co-author of* Effective COM, *and* Essential XML, *all from Addison Wesley. Don is a contributing editor and columnist at Microsoft Systems Journal (now called MSDN Magazine) and an occasional contributor to XML.com. Don is also a co-author of the Simple Object Access Protocol specification and a member of the W3C Schemas Working Group. Don has a Master's Degree in Computer Science from the University of California at Irvine.*

23 Garbage Collection

Sunday, Afternoon
Marriott Hotel — Meeting Room 12

Richard Jones, *University of Kent*

Eric Jul, *University of Copenhagen*

This tutorial presents the issues facing modern high performance garbage collectors and examines the approaches taken by state of the art garbage collectors. Participants will gain a deeper insight into the operation of modern, high performance garbage collectors. The tutorial will enable participants to evaluate the benefits and costs of such garbage collection algorithms, to understand the implications for their code and to make informed choices between collectors.

Attendee Background: Participants will be experienced programmers familiar with basic garbage collection technology (for example having attended the introductory GC tutorial — although there would be some overlap). Basic knowledge of OO implementation would be useful but not essential.

Presenters: *Richard Jones is a Senior Lecturer and Deputy Director of the Computing Laboratory at the University of Kent. He is the prime author of the book on Garbage Collection. His interests include programming languages and their implementation, storage management and distributed systems. He is a member of the Steering Committee of the International Symposium on Memory Management and was Programme Chair for ISMM'98. He has presented several tutorials at OOPSLA and ECOOP.*

Eric Jul is an Associate Professor and Head of the distributed systems group at DIKU, the Dept. of Computer Science, University of Copenhagen. He is co-designer and principal implementer of the Emerald distributed object-oriented programming language. His interests include distributed, OO languages, operating systems support including distributed storage management and object-oriented design and analysis. He was Programme Chair for ECOOP'98. He has presented tutorials regularly at OOPSLA and ECOOP

24 Advanced Use Case Writing

Sunday, Afternoon
Marriott Hotel — Florida Salon V

Alistair Cockburn,
Humans and Technology

This tutorial is a chance for practiced use case writers to gather and ask the hard questions. What is the difference between Extends and Includes? What is the difference between a business use case and a system use case? When do we stop drawing pictures and start writing text? What expertise is required in the use case writing team? How do we control the mixed levels of writing across people? Where do I put the UI design, the data descriptions and all the other requirements? Can non-technical people write the use cases? Who reviews the use cases? How do we keep use cases writers from infringing on design? The tutorial is structured as part lecture, part workshop, and part open question-and-answer. The lecture introduces the new Stakeholders & Interests model of use cases, along with the notions of different design scopes and goal levels. The workshop portion gives the attendees a chance to try their hands at resolving certain kinds of frequently occurring problems, to sharpen their skills. The open question-and-answer section allows the attendees to ask questions currently plaguing them at work, and even trade answers and experiences amongst themselves.

Attendee Background: Attendees must have written some use cases and be familiar with basic use case concepts.

Presenter: *Alistair Cockburn is a highly regarded instructor and is known as one of the premier experts on use cases. His book,* Writing Effective Use Cases, *set the standard in the area and was nominated for Software Development's Jolt book award in 2001. Alistair has taught use case writing since 1994, and has also acted as consultant on project management, object-oriented design, and methodology to the Central Bank of Norway, the IBM Consulting Group, and the First Rand Bank of South Africa. Materials that support his workshops can be found at http://members.aol.com/acockburn, http://crystalmethodologies.org and http://usecases.org.*

25 Fractal Patterns and Frameworks in UML — Towards UML 2.0?

Sunday, Afternoon
Convention Ctr — Room 21
Desmond D'Souza, *Kinetium*

The UML can be used in a simple and consistent way to (a) use a "plug-in" framework approach from business and requirements patterns, through architecture and design patterns, to code, (b) treat "objects" and "use-cases" in a fractal manner, from business to code, with patterns of refinement, (c) specify and design components using "types" and "collaborations," and (d) define component architectures based on an extensible "kit" of architectural modeling elements. This tutorial shows how UML pattern models can be used in a fractal approach to modeling and design.

Attendee Background: Attendees should be familiar with the UML.

Presenter: *Desmond D'Souza is founder and president of Kinetium. He is co-author of the* CATALYSIS Method *(Addison Wesley, 1998), and a respected speaker internationally. He was previously senior vice president of component-based development at Platinum Technology and at Computer Associates. Kinetium provides client solutions that leverage shareable architectures for model-driven development and integration of systems, with a current focus on light-weight modeling architecture and methods. Desmond can be reached at dsouzad@acm.org.*

26 Aspect-Oriented Programming with AspectJ™

Monday, Full Day
Convention Ctr — Room 20

Gregor Kiczales,
Xerox PARC,
University of British Columbia

Erik Hilsdale, *Xerox PARC*

Aspect-oriented programming (AOP) is a technique for improving separation of concerns in software design and implementation. 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. We will use numerous examples to develop participants' understanding of aspect-oriented programming through AspectJ. We will also demonstrate AspectJ's integration with IDEs such as JBuilder 4.0 and Forte4J, and emacs. AspectJ is freely available at http://www.aspectj.org

Attendee Background: Attendees should have experience doing object-oriented design and implementation, and should be able to read Java code. No prior experience with aspect-oriented programming or AspectJ is required.

Presenters: *Gregor Kiczales is Professor of Computer Science and Xerox/Sierra Systems/NSERC Chair of Software Design at the University of British Columbia. He is also a Principal Scientist at the Xerox Palo Alto Research Center, where he leads the group that has developed aspect-oriented programming and AspectJ. The focus of his research is enabling programmers to write programs that, as much as possible, look like their design. Prior to developing aspect-oriented programming he worked on open implementation, metaobject protocols, and the CLOS object-oriented programming language. He is co-author of* The Art of the Metaobject Protocol, *a key work in computational reflection. He has given numerous invited talks, lectures, and tutorials at conferences, universities, and in industry.*

Erik Hilsdale is a member of the research staff at Xerox's Palo Alto Research Center. As a member of the AspectJ project team, he focuses on language design, pedagogy, and compiler implementation. He has written several conference and workshop publications in programming languages. He is an experienced and energetic instructor in programming languages with a long background with AspectJ.

27 Software Architecture: It's What's Missing From OO Methodologies

Monday, Full Day
Convention Ctr — Room 13

Jim Doble, *Tavve Software Company*
Gerard Meszaros, *Clearstream Consulting*
Ron Crocker, *Motorola, Inc*.

Explore the challenges associated with the development of large-scale, real-life, proprietary, object-oriented, distributed, embedded, and multi-tier software systems, and discuss the path of a new professional discipline: the software architect. The software architect needs to be able to organize software systems, and make strategic design decisions, to achieve business goals related to system availability, security, scalability, survivability, long-lived flexibility, large-scale granularity, data quality and maintenance, system metrics and reports, packaging and delivery mechanisms. These issues are not commonly explored via UML or other popular modeling approaches, but are nevertheless critical to the success of modern software development projects. The teaching style for this tutorial is case-driven, and hands-on in nature. Attendees will be divided into teams to work on architecture problems. Throughout the day, working sessions will be intermixed with instructor lectures, to achieve a participatory learning experience. The goal is that attendees will learn both from the instructors and from each other. As a result, the attendee is assumed to have experience building at least one real-life software system of substantial size.

Attendee Background: Attendees should either be currently working as software architects, trying to establish a software architecture practice within their company, or working on software systems where they believe an increased emphasis on architecture is needed. Attendees should have experience building at least one real-world software system of substantial size.

Presenters: *Jim Doble has worked as software developer, manager, and architect within the telecommunications industry for over 19 years. He started his career with Nortel Networks, primarily working on central office switching systems, spent two years with Allen Telecom developing cellular infrastructure products, recently worked for Motorola, Inc. on software architectures for cellular phones, and is currently employed as a principal engineer at Tavve Software Company, developing network management solutions. In addition to architecture, Jim's technical interests include patterns, prototyping, and tools development.*

Gerard Meszaros is an acknowledged expert in software architecture and patterns. He has led or participated in workshops on software architecture at OOPSLA since 1994. He has published patterns in the first three volumes of "Pattern Languages of Program Design." His clients include Nova Gas Transmission, Tandem Computers, TELUS Communications, Digital Technics, Intelligent Databases, TransCanada Pipelines, DMR, and IBM. He has been invited to speak or participate in panels at OOPSLA, PLOP, and other national and international conferences.

Ron Crocker is a Senior Member of Technical Staff in the Network and Advanced Technology department in Motorola, Inc. where he is responsible for cellular system architecture and design. He has over 15 years of experience with object-oriented technologies, starting as a C++ guinea pig.

28 Improving Your Use Cases

Monday, Full Day
Convention Ctr — Room 22

Bruce Anderson,
IBM Component Technology Services

Paul Fertig, *IBM Global Services*

This tutorial is a working session to help you produce effective use cases for functional requirements. We will go beyond simple examples to deal with many of the issues you will face (and we have faced!) in dealing with different kinds of systems, clients, and developers. We will look at some specific topics, such as how use cases relate to business process models, using generic use cases, distinguishing envisioning from designing, effort estimation from use cases, and system exceptions. We will also look at the process of running use case workshops, and at the relation between use cases and other requirements artefacts such as the business rules catalog and non-functional requirements. Attendees are encouraged to bring specific problems for discussion, supported by shareable documents if possible.

Attendee Background: You should have written some use cases and have experience of producing requirements documents. Knowledge of OO would be useful but is not essential.

Presenters: *Bruce Anderson, Senior Consultant in IBM Component Technology Services, has been using use cases in his consulting work for several years. He has worked with clients in the banking, insurance, petroleum, and telecom industries. Bruce served on the OOPSLA'98 use case panel, and taught tutorials on use cases at OOPSLA in 1999 and 2000, the latter with Paul.*

Paul Fertig, Senior IT Architect in IBM Business Innovation Services, has been responsible for requirements gathering and architecture in large services contracts for a number of years. He has worked with clients in the telecom, retail and investment banking industries. Paul co-authored a book on OO applications which has been a key influence on IBM's world-wide software development method.

29 Pattern-Oriented Software Architecture: Patterns for Concurrent and Networked Objects

Monday, Full Day
Convention Ctr — Room 15

Douglas Schmidt,
University of California, Irvine

Developing software for distributed systems that effectively utilizes concurrency over high-speed, low-speed, and mobile networks is a complex task. This tutorial describes how to apply patterns and frameworks to alleviate the complexity of developing concurrent and distributed communication software. These patterns and framework components have been used successfully by the speaker on production communication software projects at hundreds of commercial companies for telecommunication systems, network management for personal communication systems, Web-based content delivery systems, electronic medical imaging systems, real-time aerospace systems, distributed interactive simulations, and automated stock trading. The tutorial illustrates by example how to significantly simplify and enhance the development of communication software that effectively utilizes concurrency and distribution via the use of:

- OO design techniques — such as patterns, layered modularity, and data/control abstraction
- OO language features — such as abstract classes, inheritance, dynamic binding, and parameterized types
- Middleware — such as object-oriented frameworks for infrastructure middleware (such as ACE) and distribution middleware (such as CORBA ORBs)

The material presented in this tutorial is based on the book, *Pattern-Oriented Software Architecture: Patterns for Concurrent and Distributed Objects* (Wiley 2000), which is the second volume in the highly acclaimed Pattern-Oriented Software Architecture (POSA) series.

Attendee Background: The tutorial is intended for software developers who are familiar with general object-oriented design and programming techniques (such as patterns, modularity, and information hiding) fundamental OO programming language features (such as classes, inheritance, dynamic binding, and parameterized types), basic systems programming concepts (such as process/thread management, synchronization, and interprocess communication), and networking terminology (such as client/server architectures and TCP/IP).

Presenter: *Dr. Schmidt is an Associate Professor in the Electrical and Computer Engineering Department at the University of California, Irvine. He is currently also serving as a program manager the DARPA Information Technology Office (ITO) where he is leading the national research effort on distributed object computing middleware. His research focuses on design patterns, implementation, and experimental analysis of object-oriented techniques that facilitate the development of high-performance, real-time distributed object computing middleware on parallel processing platforms running over high-speed networks and embedded system interconnects. Dr. Schmidt is an internationally recognized and widely cited expert on distributed object computing patterns, middleware frameworks, and Real-time CORBA, and has published widely in top IEEE, ACM, IFIP, and USENIX technical journals, conferences, and books. His publications cover a range of experimental systems topics including high-performance communication software systems, parallel processing for high-speed networking protocols, real-time distributed object computing with CORBA, and object-oriented design patterns for concurrent and distributed systems.*

30 Object-Oriented Reengineering

Monday, Full Day
Marriott Hotel — Meeting Room 12

Serge Demeyer, *University of Antwerp*
Stéphane Ducasse, *University of Berne*

Surprising as it may seem, many of the early adopters of the object-oriented paradigm already face a number of problems typically encountered in large-scale legacy systems. Software engineers are now confronted with millions of lines of industrial source code, developed using object-oriented design methods and languages of the late 80s. These systems exhibit a range of problems, effectively preventing them from satisfying the evolving requirements imposed by their customers. This tutorial will share our knowledge concerning the reengineering of object-oriented legacy systems. We will draw upon our experiences with the FAMOOS project, to show you techniques and tools we have applied on real industrial OO systems to detect and repair problems. In particular, we will discuss issues like tool integration, design extraction, metrics, refactoring, and program visualisation.

Attendee Background: Participants should have practical programming experience in at least one OO language (Smalltalk, C++, Java, Eiffel, ...). Familiarity with UML is useful, though not required.

Presenters: *Serge Demeyer is a professor at the University of Antwerp (Belgium). He served as technical leader for the FAMOOS project and as such has been involved in the organization of several workshops (at ECOOP and ESEC) concerning object-oriented reengineering. He has given tutorials on Object-Oriented Reengineering at both OOPSLA and ECOOP and is currently writing a book reporting on his experience.*

Stéphane Ducasse is a post doctoral researcher at the Software Composition Group in Berne (Switzerland). He served as technical leader for the FAMOOS project and as such has been involved in the organization of several ECOOP workshops concerning object-oriented reengineering. He has given tutorials on Object-Oriented Reengineering at both OOPSLA and ECOOP and is currently writing a book reporting on his experience.

31 Patterns at Work

Monday, Morning
Marriott Hotel — Florida Salon VI

Frank Buschmann, *Siemens AG, Germany*

In this tutorial we present in detail a part of a concrete real-world system and how it is designed with patterns: the representation of physical storage in a warehouse management system as well as the client interface to this subsystem. Step by step we will re-play the process of the system's construction. We discuss the design problems that occur, present the patterns that could help in solving these problems, discuss design alternatives, and show how we actually applied the patterns we selected. By this we will see how the design of the system slowly grows and evolves towards the final architecture. We will also see and discuss how patterns are applied in practice and how they help building high-quality software with predictable properties. The tutorial concludes with a summary of our experiences from several projects in which we applied patterns: what worked, what could be improved, and what did we learn.

Attendee Background: Sound knowledge in object technology, basic knowledge of UML notation, basic knowledge of the pattern concept

Presenter: *Frank Buschmann is software 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.

32 Designing an Agile Methodology

Monday, Morning
Convention Ctr — Room 19
Alistair Cockburn,
Humans and Technology

The methodology of an organization is a social construction that includes the roles, skills, teaming, activities, techniques, deliverables, standards, habits and culture of the organization as it develops software. This tutorial starts with language and constructs needed to evaluate, compare, and construct methodologies. These include precision, accuracy, tolerance, relevance, and scale, along with the nine basic elements of a methodology. Several examples of effective, lightweight, and real methodologies are given, along with commentary on the social setting for each. The tutorial examines the conditions suited to shifting from a lighter to a heavier methodology and the penalty for doing so. The tutorial ends with the presentation of a small family of agile methodologies, optimized for productivity, making maximum use of human, face-to-face communication. Considerations about success and failure in affecting culture are visited again at the end. Learn to identify and diagnose the parts of your organization's methodology, and learn ways to make it more effective. Attendees should have significant software team experience, preferably but not necessarily OO, and must have used at least one methodology and thought about others.

Attendee Background: Experienced developers, team leaders, methodologists, and technology selectors trying to choose or design a methodology for their organization.

Presenter: *Alistair Cockburn, founder of Humans and Technology, was special advisor to the Central Bank of Norway for object technology and software project management, and the designer of the IBM Consulting Group's first OO development methodology. His books,* Surviving Object-Oriented Projects *and* Writing Effective Use Cases, *have garnered praise from practitioners for being pragmatic and readable. He is an expert on use cases, object-oriented design, project management, and software methodologies. He has been the technical design coach and process consultant on projects ranging in size from 3 to 90 people. Materials that support Alistair's workshops can be found at http://members.aol.com/acockburn and http://crystalmethodologies.org.*

33 Exposing and Consuming Web Services with .NET

Monday, Morning
Convention Ctr — Room 21

Casey Chesnut, *iigo*

This tutorial will demonstrate Web Services in the .NET Framework. A Web Service is application logic accessible through standard web protocols and data formats. They are an integral part of the .NET Framework. The tutorial will be divided into 2 parts. The first part will focus on how to expose a Web Service for clients to access, and the second part will cover how to consume the exposed Web Service from a variety of clients. Some time will be spent exploring applicable architectures and modeling techniques for Web Services. Throughout the presentation, non-functional requirements will also be considered (e.g. security, authentication, performance, etc.) The tutorial's objective is to give the audience an intermediate-level introduction to Web Services development in the .NET Framework, as well as design decisions that are pertinent to the programming model. The audience will be exposed to XML, SOAP, UDDI, WSDL, ASP.NET, and C#. This tutorial will be presentation based with code examples.

Attendee Background: The target audience will be Software Engineers, although Management will be interested to get a glimpse at Web Services and the different business models that are made possible. Basic understanding of Internet technologies will be helpful.

Presenter: *Casey Chesnut is Vice President of Technology for iigo, Inc. He specializes in cutting-edge technologies and has most recently been concentrating on Web Services in the .NET Framework. He holds two Masters degrees in software engineering.*

34 Efficient Architectures for Object-Oriented Component-Based Middleware

Monday, Morning
Marriott Hotel — Salon A

Michael Stal, *Siemens AG, Germany*

Due to the importance of distribution and object technologies, infrastructures for distributed object computing and component-based middleware have become commonplace. However, it is not sufficient to just read the specification of standards such as Java RMI, EJB, or CORBA, and then build applications using these standards. On the one hand, the transparency provided by these platforms helps developers to master the complexity of building distributed systems, but on the other hand, it is necessary to know the infrastructure's internal architectural design to leverage it efficiently. Unfortunately, the architectural principles behind infrastructures are not documented anywhere. Here, patterns come to our rescue. They do not only enable the solution of recurring problems in software development, but also help us to look inside existing software in order to understand it and leverage it efficiently. Thus, the goal of the tutorial is to show the basic principles behind distributed object computing and component-based middleware. Patterns from existing pattern books will be introduced step-by-step to reveal the overall architecture of these infrastructures. These patterns will not only help to understand middleware, but will also be applicable for the development of any distributed systems. In the first part of the tutorial we use patterns to explain the basic architecture of object-oriented middleware from a user perspective. In the second part we will dive into the internals of middleware frameworks.

Attendee Background: Attendees should be familiar with distributed systems. They should have basic experience with Java and C++. Knowledge with patterns is not required.

Presenter: *Michael Stal works as a Senior Principal Engineer for Siemens Corporate Technology where he is head of the Middleware & Application Integration Team. His main research areas include Object-Oriented Middleware, Patterns, Software Architecture, Web Technologies, and Component-based Software Development. Michael is Siemens representative at the OMG, and former member of the C++ standardisation working group X3J16. He is co-author of the books,* Pattern-Oriented Software Architecture - A System of Patterns *and* Pattern-Oriented Software Architecture - Vol. 2: Patterns for Concurrent and Networked Objects. *In addition, he serves as editor-in-chief of the German Java Spektrum magazine. Michael has published articles in many magazines and given talks at many conferences world-wide.*

35 Extreme Programming Live!

Monday, Morning
Marriott Hotel — Florida Salon IV

William Wake, *Capital One*
Steve Metsker, *Capital One*

Extreme Programming (XP) is an agile software development method that emphasizes ongoing user involvement, automated testing, and pay-as-you-go design. This tutorial introduces XP practices through hands-on exercises:

- Planning Game: User Stories, On-Site Customer
- Programming Game: Test-First Programming, Unit Testing, Pair Programming
- Refactoring Game: Code Smells, Once-and-Only-Once, Refactoring

The exercises are paper-based and use a fireworks factory as their domain. Student volunteers help play the part of the customer and the unit testing framework. As a participant, you will help create a live simulation of several key practices of Extreme Programming.

Attendee Background: Some familiarity with object-oriented concepts is helpful; no prior experience with XP is needed.

Presenters: *William Wake is interested in XP, patterns, human-computer interaction, and information retrieval. He is the author of* Extreme Programming Explored *and the inventor of the Test-First Stoplight and the XP Programmer's Cube.*

Steve Metsker is a researcher and author who explores and writes about ways to expand the abilities of developers. Steve's articles have explained how to maintain relational integrity in object models, how to solve logic puzzles in Java, and how the concept of "object" differs between Plato and the OO languages. Steve's most recent publication is the book, Building Parsers with Java.

36 Patterns and Architectures for J2EE Systems

Monday, Morning
Marriott Hotel — Florida Salon V

Kyle Brown, *IBM*

When people view the J2EE (Java 2 Platform, Enterprise Edition) specifications, too often all they see is a “bag of APIs” without a way to understand how the specifications work together in an application server. In this tutorial we will examine some template architectures for successful J2EE systems and show how a common set of design patterns can be applied to help designers navigate through the J2EE problem space. We will examine common client-side pitfalls and discuss the pros and cons of different design options, discuss how Java Servlets and JavaServerPages (JSPs) can work with and within technologies like Apache Struts, XML, and Extensible Style Sheets (XSL), and discuss how EJB (Enterprise JavaBean) systems can be structured to maximize component reuse while reducing wasted programmer effort.

Attendee Background: This tutorial is targeted to Java programmers and designers, with at least some exposure to J2EE technologies (a reading knowledge of the J2EE specification and the associated API specifications will be sufficient). Programmers who have had experience with one or more of the J2EE technologies will gain the most from this review of how all the technologies fit together and how problems are solved using the entire J2EE framework.

Presenter: *Kyle Brown is an Executive Java Architect with IBM’s WebSphere Services unit. He is an experienced presenter at OOPSLA and other industry conferences. He has over twelve years of experience with object-oriented systems, and has been specializing in Enterprise Java systems since 1997. He is a co-author of* The Design Patterns Smalltalk Companion *and* Enterprise Java Programming with IBM WebSphere*, both published by Addison Wesley Longman.*

37 Refactoring: Improving the Design of Existing Code

Monday, Morning
Convention Ctr — Room 16

Martin Fowler, *ThoughtWorks, Inc.*
Josh MacKenzie, *ThoughtWorks, Inc.*

Almost every expert in object-oriented development stresses the importance of iterative development. As you proceed with iterative development, you need to add function to the existing code base. If you are really lucky, that code base is structured just right to support the new function while still preserving its design integrity. Of course, most of the time we are not lucky, and the code does not quite fit what we want to do. You could just add the function on top of the code base. But soon this leads to applying patch upon patch, making your system more complex than it needs to be. This complexity leads to bugs, and cripples your productivity. Refactoring is all about how you can avoid these problems by modifying your code in a controlled manner. Done well, you can make far-reaching changes to an existing system quickly, and without introducing new bugs. You can even take a procedural body of code and refactor it into an effective object-oriented design. With refactoring as part of your development process you can keep your design clean, make it hard for bugs to breed and keep your productivity high. In this tutorial we'll show you an example of how a lump of poorly designed code can be put into good shape. In the process we'll see how refactoring works, demonstrate a handful of example refactorings, and discuss the key things you need to do to succeed. This tutorial is an introduction to refactoring. No prior refactoring experience is assumed and the content covers much the same ground as opening a couple of chapters of the refactoring book.

Attendee Background: developers and analysts

Presenters: *Martin Fowler is the Chief Scientist for ThoughtWorks Inc., an Internet professional services provider specializing in the delivery of highly strategic B2B e-Commerce solutions. For a decade he was an independent consultant pioneering the use of objects in developing business information systems. He's worked with technologies including Smalltalk, C++, object and relational databases, and EJB with domains including leasing, payroll, derivatives trading and healthcare. He is particularly known for his work in patterns, the UML, lightweight methodologies, and refactoring. He has written four books:* Analysis Patterns, Refactoring, *the award winning* UML Distilled, *and* Planning Extreme Programming.

Josh MacKenzie has been with ThoughtWorks for three years, serving as a developer, architect, and team lead. He has worked on projects in equipment leasing, insurance, and industrial supply and purchasing. These projects have utilized a wide variety of technologies, including J2EE, XML, Forte, and LDAP. Josh has also been instrumental in the exploration and adoption of agile methodologies on ThoughtWorks' projects. Prior to ThoughtWorks, Josh served as a Senior Engineer for Motorola, Inc. Energy Systems, where he designed and developed real-time testing and analysis software for electrochemical capacitors. He holds a B.A. in Physics and Mathematics, and an almost-M.S. in Chemical Engineering. Josh presented tutorials at JavaCon2000 on "Refactoring" and "Business Objects in J2EE."

38 Efficient Implementation of Object-Oriented Programming Languages

Monday, Afternoon
Marriott Hotel — Salon A

Craig Chambers,
University of Washington

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.

Attendee Background: 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: *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, heads the Vortex whole-program optimizing compiler project, and co-leads the DyC staged dynamic compilation project.*

39 Making the Software Process Transparent by Using Intelligent Agents

Monday, Afternoon
Marriott Hotel — Florida Salon V

Ivar Jacobson, *Rational*
Gunnar Övergaard, *Jaczone AB*

It has never been so hard to develop good software as today. Developers need more knowledge and skill than ever before. They need to be skilled in programming languages (e.g. Java, C++), system software platforms (.NET, J2EE), XML, middleware (WebSphere, Logicworks, etc.), the Unified Modeling Language, the Rational Unified Process, web architectures, etc. And they need to learn about these technologies faster than ever with almost no time for training and education. If they don't, their only rescue is to find shortcuts, use lightweight methodologies, and ignore well-proven best practices. And as usual, quality will suffer. There is another way. In this tutorial we will discuss how software agents can be used to reduce the gap between the individual developers' knowledge and what is needed. For instance, agents can minimize the process adoption thresholds so that the complexity of a process can become transparent to the developers and thus be perceived as lightweight. The individual developer will focus on the problem solving and creative part, letting the agents do the work that can be guided by formalized knowledge. We will discuss the process of formalizing knowledge as rules, how these rules will trigger in a given context, and how the agents can propose resolutions. Examples will be used to demonstrate the feasibility of agents in software development.

Attendee Background: System analysts, project leaders, software developers, people interested in methodologies, process development and software development tools Required experience: Some experience with software development and UML.

Presenters: *Dr. Ivar Jacobson serves as Vice President of Business Engineering for Rational Software Corp. Dr. Jacobson is the founder of Objectory AB in Sweden, which merged with Rational Software in 1995. He was one of the three original designers of the UML in 1997. He is the principal author of three influential and best-selling books,* Object-Oriented Software Engineering—A Use Case Driven Approach, The Object Advantage—Business Process Reengineering with Object Technology, Software Reuse: Architecture, Process, and Organization for Business Success, *and* The Unified Software Development Process.

Gunnar Övergaard serves as Vice President Content Development at Jaczone AB, and holds a Ph.D. in Computer Science. Gunnar has worked with process development, consulting, and education in the object-oriented field since the mid-1980s. Gunnar worked as VP of process development at Objectory in the critical development of the origin to the Rational Unified Process. He has participated actively in the development of UML since 1995.

40 Surviving Object-Oriented Projects

Monday, Afternoon
Marriott Hotel — Florida Salon VI

Alistair Cockburn,
Humans and Technology

After a decade of OO projects, what have we learned? Develop in increments to form a habit of delivering; make sure you have a decent executive sponsor, project manager and technical leader; get training; and don't forget the lessons of the last four decades. This tutorial walks through appropriate expectations, project setup, technology selection, working with the domain model, with increments and iterations, prototypitis and the death spiral, setting up teams, handling training, expanding, project management patterns and advice from hindsight. It includes over 60 specific strategies, truths, fixes, and recommendations. The tutorial is liberally sprinkled with case histories and insights from the participants themselves. It overlaps heavily with the book, *Surviving Object-Oriented Projects*. The tutorial is for anyone wanting to learn ways to get their OO project out of the mire, or avoid the mire in the first place.

Attendee Background: Neither being a OO novice nor an OO expert will interfere with the tutorial material. The only requirements is an interest in what makes a project work well.

Presenter: *Alistair Cockburn, founder of Humans and Technology, was special advisor to the Central Bank of Norway for object technology and software project management, and the designer of the IBM Consulting Group's first OO development methodology. His books,* Surviving Object-Oriented Projects *and* Writing Effective Use Cases, *have garnered praise from busy practitioners for being pragmatic and readable. He is an expert on use cases, object-oriented design, project management, and software methodologies. He has been the technical design coach and process consultant on projects ranging in size from 3 to 90 people. Materials that support Alistair's workshops can be found at http://members.aol.com/acockburn and http://crystalmethodologies.org.*

41 Refactoring to Patterns

Monday, Afternoon
Convention Ctr — Room 21

Joshua Kerievsky, *Industrial Logic*

While Software Patterns are undeniably powerful design aids, many programmers tend to overuse them, prematurely introduce them, or implement them in unnecessarily heavyweight ways. Refactoring to Patterns encourages a simpler, more disciplined approach to using Patterns, based on the philosophy of Extreme Programming. Using this approach, programmers wait for the right time to refactor a Pattern into a system and do so using the simplest possible Pattern implementations. In this tutorial, we will examine five Design Patterns and five cases where we might refactor these Patterns into Java code. During the process, we will investigate when it makes sense to refactor to a Pattern, and what are simple implementations of each Pattern we add.

Attendee Background: This is an intermediate-level tutorial. Attendees will be expected to understand Java and have basic exposure to Design Patterns.

Presenter: *Programming professionally since 1987, Joshua Kerievsky is the founder and chief programmer of Industrial Logic, Inc. (http://industriallogic.com), a company specializing in Patterns and XP. As an XP Coach, mentor, and leader of intensive workshops, Joshua helps organizations learn and use the software industry's very best practices. Joshua can be reached at Joshua@industriallogic.com.*

42 How to Really Fail at Software Architecture

Monday, Afternoon
Convention Ctr — Room 19

Luke Hohmann, *Independent Consultant*

Most books and lectures on software architecture focus on technical issues. This is clearly necessary, because software architecture must deal with technical concerns. A smaller subset focuses on other important issues such as “peopleware.” This is clearly necessary, for software systems are built by people to satisfy one or more needs. Unfortunately, few lectures focus on the business realities of software architecture. This tutorial addresses these business realities, for without addressing them your architecture will surely fail.

Attendee Background: Participants should have been a technical lead, first line manager, senior developer, or software architect for at least one project (including the one they’re working on right now, if this is their first).

Presenter: *Luke Hohmann is an independent consultant committed to coaching his clients to greater levels of performance. Mr. Hohmann is author of* Journey of the Software Professional: A Sociology of Software Development *from Prentice-Hall as well as numerous articles on software engineering management. A skilled instructor and speaker, Mr. Hohmann has been invited to many conferences. Mr. Hohmann can be contacted through e-mail at LukeHohmann@yahoo.com.*

43 The Art of Writing Use Cases

Tuesday, Full Day
Marriott Hotel — Meeting Room 12

Rebecca Wirfs-Brock,
Wirfs-Brock Associates

John Schwartz, *Wirfs-Brock Associates*

Use cases describe the behavior of a software system from an external usage perspective. There is an art to writing them clearly. Written carefully, use case models convey key usage specifications and can be tied to other requirements. Written poorly, use cases are confusing and ambiguous. This tutorial presents examples of good and bad use case descriptions, and practical techniques for writing three forms of descriptions: narratives, scenarios, and conversations. Narratives are high-level descriptions written from an external perspective. We show how to elaborate high-level descriptions, choosing either a scenario form, which emphasizes sequence, or a conversation, which highlights interactions between a user and the system. Tips for naming use cases, describing policies, errors, and exceptions, attaching other important information, describing meaningful pre- and post-conditions, and creating informative glossary entries are also presented. This tutorial will expose students to techniques for critically reading and revising use cases in various forms, techniques for asking probing questions and filling in use case details, and techniques for developing a use case model which interleaves both group activities with individual writing tasks and review.

Attendee Background: Attendees should be looking for practical ways to improve their writing. They should be familiar with writing and reading software requirements and usage descriptions. Attendees could benefit from an introduction to object concepts. However, an object background is not a prerequisite!

Presenters: *Rebecca Wirfs-Brock is president of Wirfs-Brock Associates, a firm specializing in the transfer of object analysis and design expertise to organizations and individuals through training, mentoring, and consulting. Rebecca has been involved with object technology since its infancy. She is the inventor of the set of development practices known as Responsibility-Driven Design. From development on the Tektronix implementation of Smalltalk in the early 1980s, through years of development and training experience, she is recognized as one of only a few knowledgeable and influential practitioners of object-oriented design. She spent 17 years as a Software Engineer at Tektronix, where she managed the first commercial Smalltalk effort and was the technical lead for the development of Color Smalltalk. Recently, she has authored use cases for a telecommunications framework and an online banking system and has mentored teams in use case writing, design, architecture and managing incremental, iterative object-technology projects. She practices what she teaches!*

John Schwartz is Vice President of Consulting Services at Wirfs-Brock Associates and a widely known and respected authority on object analysis and design. John has over 15 years of experience developing and managing object-oriented projects in telecommunications, medical, and CAD. He has served as Vice President and Director of Software Architecture of a 120-person telecom information technology group. While with ParcPlace Systems, he influenced the development of the Object Behavior Analysis method pioneered by Adele Goldberg and Kenny Rubin. John was chairman of the OMG's original Object Model Task Force, and developed the model that CORBA is based on. He has contributed to the definition and practical application of Object Behavior Analysis, and Responsibility-Driven Analysis and Design methodologies. He consults on design and methodology to major object-oriented projects. He has conducted over 100 tutorials and classes on object analysis, design, and programming.

44 Architecting Large Business Systems

Tuesday, Full Day
Convention Ctr — Room 25

Jens Coldewey, *Coldewey Consulting*
Alan O'Callaghan,
De Montfort University
Wolfgang Keller, *Generali Vienna Group*

Architecture is one of the key issues in successful software projects. In the recent years, many publications have discussed what architecture is about and how to describe it. However, there has been little information about how a good architecture is found and how to implement it into a project successfully. Based on the field research at De Montfort University and many years of hands-on experience, this tutorial teaches best-practices on how to develop an architectural model for business systems. It addresses both "green-field" projects and re-engineering efforts with a special emphasis on agile processes and on component architectures. The tutorial is designed interactively with more than half of the time dedicated for exercises and discussion.

Attendee Background: This tutorial aims at designers and project managers of object-oriented business systems who are interested in software architecture and who are open to interactive learning experiences. They should be familiar with object-oriented design patterns as published by Gamma, Helm, Johnson, and Vlissides in "Design Patterns — Elements of Reusable Object-Oriented Software"

Presenters: *Jens Coldewey (jens_coldewey@acm.org) is independent consultant in Munich, Germany, specialized in deploying agile processes and object-oriented techniques in large organizations. He consults architecture projects in several large projects. Jens Coldewey writes a column on Agile Processes in the German SIGS/101 magazine OBJEKTSpektrum.*

Alan O'Callaghan (aoc@dmu.ac.uk) is Senior Lecturer in computer science at De Montfort University, Leicester, England. He has consulted in the migration of legacy systems to object and component-based systems in a number of industrial sectors and authored the ADAPTOR pattern language. He writes a column on migration in the SIGS/101 journal Application Development Advisor.

Wolfgang Keller (wolfgang_keller@acm.org) is a principal architect for Generali Vienna Group. His responsibilities include the technical base for Generali's Phoenix line of insurance applications, product architecture, and project coordination for Generali's distributed development across parts of Europe.

45 The UML's Object Constraint Language (OCL) — Specifying Components

Tuesday, Afternoon
Marriott Hotel — Meeting Room 11

Jos Warmer, *Klasse Objecten*

Anneke Kleppe, *Klasse Objecten*

As the use of UML grows and UML is applied to more fields of software and systems engineering, the need for more precise specifications grows. This is crucial, for example, when generating code or test cases from specifications. In the component based world we need to be able to specify the behavior of components in a very precise way. This enables us to know whether components are plug-compatible and allows us to derive the behavior of assembled components and make sure that they have the desired behavior. For these purposes UML's Object Constraint Language is becoming more popular as a standardized and language independent specification mechanism. This tutorial shows the importance of constraints as an object-oriented specification technique and how they add value to the visual modeling techniques of UML. The OCL language itself and the connection with the visual UML diagrams is thoroughly explained. The final part of the tutorial will show how one can apply constraint modeling in UML to achieve the above described goals.

Attendee Background: The tutorial is targeted to people that have knowledge of and experience with analysis and design methods like UML. They should specifically have experience in developing object or class models.

Presenters: *Jos Warmer is senior consultant at Klasse Objecten. He is the chief architect of OCL and responsible within the UML core team for all matters concerning OCL.*

Anneke Kleppe is an independent OT consultant who founded her own company Klasse Objecten in 1995. She has developed her own training and mentoring program and has applied this with many clients. She has actively supported the UML core team on the subject of OCL.

Anneke and Jos have co-authored the book entitled, The Object Constraint Language: Precise Modeling with UML, *which has been published in the OT series by Addison-Wesley Longman, USA. They also wrote (Dutch) books on OMT and UML, published by Addison-Wesley.*

46 J2ME Design and Development Considerations

Tuesday, Afternoon
Convention Ctr — Room 16

David Hemphill,
Catapult Technologies, Inc.

Java 2 Micro Edition (J2ME) is Sun's Java 2 platform for consumer electronics and embedded devices. Writing software for limited devices offers unique challenges to the application developer. In many cases, developers are required to improvise in graphical user interface, persistent storage, I/O, and other areas in order to make the application work under resource constraints. The extremely varied nature of many of the targeted devices makes solving issues even more challenging.
In this presentation, many of the difficulties and potential solutions of developing mobile, wireless, and other limited device J2ME applications will be addressed.

Attendee Background: Participants should be familiar with Java programming and have some familiarity with J2ME.

Presenter: *David Hemphill is a Senior Software Architect with Catapult Technologies, Inc. He has over ten years of experience in developing and architecting software systems. David's technical expertise lies in Java, J2EE, J2ME, EJB, UML, XML, and relational databases. He is a graduate of the University of Wisconsin, Eau Claire.*

47 Embedded Systems in C++ — C++ Idioms, Patterns, and Architecture for Constrained Systems

Tuesday, Afternoon
Convention Ctr — Room 5

Detlef Vollmann,
Vollmann Engineering GmbH

Embedded software is different: it often involves hard real-time constraints and very limited memory. But the challenges for embedded systems are even higher: they must be very reliable (99.999% is not good enough) and need to tackle a variety of different memory types (standard RAM, EEPROM, Flash, buffered RAM, ...). Another frequent design problem is the different kinds of processing tasks: interrupt handling, hardware control, application processes, all interconnected by a selection of communication means. To design a system in such an environment, a rich programming language like C++ seems to be quite useful. But on the other hand, it is often more difficult to keep the tight control necessary, e.g., for hard real-time requirements, with such a high-level language. This tutorial will present and discuss various techniques for designing and implementing typical embedded systems problems in C++. Different language features are analyzed with respect to embedded requirements. Participants of this tutorial will acquire a thorough understanding of the potentials of C++ for the development of systems with tight limitations. They will learn effective techniques to deal with these limitations even for complex systems.

Attendee Background: Participants should have a good working knowledge of ISO C++. Experience in the design of embedded systems will be helpful but is not essential.

Presenter: *Detlef Vollmann has a background of 15 years in software engineering and more than 10 years with object technology. As an independent consultant he supports several Swiss companies with the design of object-oriented systems. Since 1991, he has authored and taught courses in C++, Object-Oriented Technologies, Software Architecture, and Distributed Computing for major Swiss companies.*

48 Adaptive Object-Model Architecture: How to Build Systems That Can Dynamically Adapt to New Business Requirements

Tuesday, Afternoon
Convention Ctr — Room 15

Federico Balaguer,
Software Architecture Group - Department of Computer Science, University of Illinois

Joseph Yoder,
Software Architecture Group - Department of Computer Science, University of Illinois

Architectures that can dynamically adapt to changing requirement are sometimes called “reflective” or “meta” architectures. We call a particular kind of reflective architecture an “Adaptive Object-Model (AOM)” architecture. This tutorial will explain AOMs and how to implement them. An Adaptive Object-Model is a system that represents classes, attributes, and relationships as metadata. It is a model based on instances rather than classes. Users change the metadata (object model) to reflect changes in the domain. These changes modify the system’s behavior. In other word, it stores its Object-Model in a database and interprets it. Consequently, the object model is active, when you change it, the system changes immediately. We have noticed that the architects of a system with Adaptive Object-Models often claim this is the best system they have ever created, and they brag about its flexibility, power, and eloquence. At the same time, many of the developers find them confusing and hard to work with. This is due in part because the developers do not understand the architecture. This tutorial will give a description of this architectural style and will make it easier for developers to understand and build systems that need to quickly adapt to changing business requirements.

Attendee Background: A good knowledge of object concepts is required. It would be useful if participants have a basic understanding of frameworks, though it is not necessary. A general understanding of the GOF patterns is required and Fowler’s Analysis Patterns is helpful.

Presenters: *Federico Balaguer has been developing object-oriented software for over ten years. He has worked on many projects including J.P. Morgan in Argentina. He is currently working on implementing Martin Fowler’s Analysis Patterns at Illinois Department of Public Health and is also working with Professor Ralph Johnson on finishing his Ph.D.*

Joseph W. Yoder has worked on the architecture, design, and implementation of various software projects dating back to 1985. Recently he has taught Object-Oriented concepts including Patterns and Smalltalk to Caterpillar and the Illinois Department of Public Health (IDPH) analysts and developers, and has mentored many developers on the development applications being deployed across the state of Illinois such as the Newborn Screening application, the Refugee System, and the Food Drug and Dairy application. He is also coordinating the efforts of this development as the primary architect of the reusable frameworks being developed. Joe is the author of over two dozen published patterns and has been working with patterns for a long time, writing his first pattern paper in 1995, and chairing the PLoP‘97, conference on software patterns.

49 No Stone Unturned: An Introduction to Test-First Programming

Tuesday, Afternoon
Convention Ctr — Room 3

Steve Freeman, *Big Blue Steel Tiger*

Writing effective unit tests is a skill that takes practice to do well but is at the core of Extreme Programming. During this tutorial we will demonstrate real examples that address situations many programmers find hard to test. We will also show how test-driven development improves the quality of the code produced. Finally, we will consider the practical boundaries of effective unit testing. This tutorial will improve attendees' understanding of how to write meaningful and effective unit tests, demonstrate live, test-driven development and pair-programming, and cover topics that many people find difficult when writing unit tests.

Attendee Background: The session is intended for working programmers who are interested in writing effective unit tests. They should be familiar with Java and, preferably, standard libraries such as JDBC and servlets. The tutorial has a bias towards web development, but the techniques it covers are applicable elsewhere.

Presenter: *Steve Freeman is a Principal Consultant at Big Blue Steel Tiger, where he develops e-commerce solutions and is also responsible for helping to move Big Blue Steel Tiger towards Extreme Programming. Prior to this, he ran the largest XP project in the UK at Lombard Risk Systems. He has degrees in Statistics and Music, and a Ph.D. in computer science from Cambridge University and has written software for research labs, shrink-wrap and bespoke systems.*

50 Designing Software Architecture for Quality: The ADD Method

Tuesday, Afternoon
Convention Ctr — Room 22

Len Bass,
Software Engineering Institute

Felix Bachmann, *Robert Bosch, GmbH*

It has long been recognized that quality attributes are, in large part, determined by the software architecture of a system. This recognition has been the basis for several software architecture analysis methods. The application of this recognition to the problem of designing the software architecture is a recent development, however. This tutorial will introduce the Attribute Driven Design (ADD) method. ADD is a method for designing the software architecture of a system or collection of systems based on an explicit articulation of the quality attribute goals for the system(s). The method is appropriate for any quality attributes but has been particularly elaborated for the attributes of performance, modifiability, security, reliability/availability and usability. The method has been used for designing the software architecture of products ranging from embedded to information systems.

Attendee Background: This half-day tutorial is designed for attendees who have practical knowledge of software architecture and experience in working with and designing large systems.

Presenter: *Len Bass is a senior software engineer at the Software Engineering Institute of Carnegie Mellon University (CMU). He has written or edited six books and numerous papers in a wide variety of areas of computer science including software engineering, human-computer interaction, databases, operating systems, and theory of computation. His most recent book,* Software Architecture in Practice *(co-authored with Clements and Kazman), received the Software Development Magazine's Productivity Award. He headed a group that developed a software architecture for flight training simulators that has been adopted as a standard by the U.S. Air Force. He also headed a group that developed a technique for evaluating software architectures for modifiability. He is currently working on techniques for the analysis of software architectures, on techniques for the development of software architectures for product lines of systems, and on the how to achieve usability through architectural means. He is the representative of the ACM to the International Federation of Information Processing technical committee on Software: Theory and Practice. Before joining CMU in 1986, he was professor and chair of the Computer Science Department at the University of Rhode Island. He received his Ph.D. in computer science in 1970 from Purdue University.*

Mr. Bachmann is currently Project Manager for the Product Line approach within Robert Bosch, GmbH. In cooperation with the Product Line Systems Program of the Software Engineering Institute (SEI), he makes this approach available to the Bosch business units. Prior to his current assignment, he worked as a member of the Robert Bosch research institute with the software development departments to address the issues of more functions and higher quality in the "call-control software," — the core of telecommunication products. This is where he developed the foundation for the next generation of telecommunications software. As a result of these efforts, Bosch developed the method OTES (Objects Through Essential Services) in which Mr. Bachmann played a decisive role. Mr. Bachmann also defined the corresponding software development process that describes in three levels how to develop high quality software in a timely fashion.

51 Creating Responsive, Scalable Systems

Tuesday, Afternoon
Convention Ctr — Room 13

Connie Smith,
Performance Engineering Services
Lloyd Williams,
Software Engineering Research

Performance, both responsiveness and scalability, is an important quality of today's software. Yet, many software systems cannot be used as they are initially implemented due to performance problems. These performance failures can translate into significant costs due to damaged customer relations, lost income, and time and budget overruns to correct the problem. Our experience is that performance problems are most often due to fundamental architectural or design problems rather than inefficient coding. Thus, performance problems are introduced early in the development process but are typically not discovered until late, when they are more difficult and costly to fix. This tutorial presents a systematic, quantitative approach to cost-effectively designing performance into object-oriented software systems. It also presents an overview of principles and patterns for designing performance into software as well as antipatterns for recognizing and fixing common problems. Objective: Participants will be acquainted with a cost-effective, quantitative approach to managing software performance. They will learn practical techniques for diagnosing performance problems early in the development process when these problems can be fixed quickly and easily. They will also learn techniques for designing performance into software.

Attendee Background: Attendees should be familiar with object-oriented development. No background or experience in software performance engineering is required.

Presenters: *Connie U. Smith, Ph.D., a principal consultant of the Performance Engineering Services Division of L&S Computer Technology, Inc., is known for her work in defining the field of Software Performance Engineering (SPE) and integrating SPE into the development of new software systems. Dr. Smith received the Computer Measurement Group's prestigious AA Michelson Award for technical excellence and professional contributions for her SPE work. She authored the original SPE book,* Performance Engineering of Software Systems, *and approximately 100 scientific papers. She is the creator of the SPE·ED performance engineering tool. She is a frequent speaker at conferences and has delivered numerous keynote addresses on SPE. In her work at L&S Computer Technology she specializes in the development and support of the performance engineering tool, SPE·ED, applying performance prediction techniques to software, teaching SPE seminars, and research and writing on SPE.*

Dr. Lloyd G. Williams is principal consultant at Software Engineering Research where he specializes in the development and evaluation of software architectures that meet quality of service objectives such as: performance, reliability, modifiability, and reusability. Dr. Williams was previously Associate Professor of Computer Science at the University of Colorado at Denver. He also served as Director of the Rocky Mountain Institute of Software Engineering, a non-profit organization founded to promote research and education in software engineering. His work has emphasized the transfer of leading-edge software engineering technology into widespread use. He has presented professional development seminars and served as a consultant on software development for more than 100 organizations in the USA, Japan, and Europe. He has authored numerous technical articles and is a contributor to the AIAA Progress Series book, Aerospace Software Engineering.

52 Leading Retrospectives on OO Projects: Looking Back to Move Forward

Tuesday, Afternoon
Convention Ctr — Room 14

Norm Kerth, *Elite Systems*

Linda Rising, *Independent Consultant*

In the object-oriented world, it's our fate to experiment with new ideas. It started with objects and continued with OMT, Fusion, Objectory, UML, Patterns, Extreme Programming and so on. Our community is good at accepting new concepts but are we good at evaluating the results? "You have paid the tuition, now are you going to learn the lessons?" Carefully reviewing an OO project at its end is one of the most obvious ways of improving your software development process and building OO mastery. Sadly, such a review rarely happens — why? Because we don't know how to do it; we don't know how to deal with the emotions; we don't know how to address failure; and we don't know how to convert the lessons into new practices. In this tutorial, we look at an entire philosophy and methodology to lead an effective review of a significant OO project. Expect to learn a number of tools, techniques and skill areas necessary to be an accomplished facilitator.

Attendee Background: Managers, Project Leaders, Software Process Group Specialists, Trained Facilitators, Methodologists, Technical Leaders

Presenters: *Norm Kerth is an experienced software engineer and researcher focusing on specification and design activities, quality assurance, continuous process improvement, project management and growing effective teams. He has led retrospectives for over 20 years and critics predict his book,* Project Retrospectives, *will become "the next classic in our field." Norm has been a full time consultant since 1984 and helps firms improve their software engineering discipline. He has particular interest in objects, pattern languages, and building high performance teams. Prior to starting his company, Elite Systems, he was a professor at the University of Portland. He has a decade of engineering experience with Tektronix and is a master teacher, with over 30 years of experience in front of students.*

Linda Rising has a Ph.D. from Arizona State University in the area of object-based design metrics. Her background includes university teaching experience as well as work in industry in the areas of telecommunications, avionics, and strategic weapons systems. She has been working with object technologies since 1983. She is the editor of A Patterns Handbook, The Pattern Almanac 2000, and Design Patterns in Communication Software.

53 Business Modeling with the UML

Tuesday, Afternoon
Marriott Hotel — Meeting Room 1

Granville Miller, *TogetherSoft*

Developing models of your business can serve many purposes. These models can help you as you continually engineer your business to keep up with the competitive pressures of today's business environment. Additionally, they can help as you develop the software systems that give you competitive advantage. UML modeling elements such as use cases and business object models are excellent ways of capturing existing business processes or the processes of the future. These models can then be used to communicate requirements to software engineers, the organization at large, or the people who will be a vital part of carrying out the objectives of the process. In this tutorial, we describe methods of capturing processes and modeling them. We present UML as a method of describing them. Finally, we introduce some additions to the UML which will aid organizations to deploy these systems and processes in their organizations.

Attendee Background: This tutorial is geared toward business analysts, requirements engineers, and software managers and developers who wish to place their software solutions in the context of a business process.

Presenter: *Granville Miller is currently a mentor at TogetherSoft. He is the co-author of* Advanced Use Case Modeling, Volume I, *and has two more books slated for the end of the year. He has been active in the OO community and has given tutorials at several conferences. He has been working heavily with use cases as a method of business modeling for several years.*

54 XP Meets UML: Development Processes for eTechnology

Tuesday, Afternoon
Convention Ctr — Room 23

Alan Cameron Wills,
Trireme International Ltd.

Extreme Programming has justifiably become wildly popular, especially among developers of eCommerce systems, for whom lightweight process and responsiveness to changing requirements are prerequisites. And yet practices like analysing and documenting requirements upfront used to seem like a good idea. So did careful definition of the interfaces between components — an essential for CBD and EAI. This tutorial draws on the presenter's experience in constructing a corporate process that combines the best of iterative development together with requirements and interface specification techniques from the Catalysis approach. We will look at how to ensure that requirements analysis raises questions to the customer, rather than just recording use-cases; how to relate the incrementally-delivered features to the use-cases in the requirements model; how to use the requirements and interface models to generate tests; and how to ensure that the customer-developer link is enhanced rather than weakened by the mediation of the requirements analyst. Participants will be able to improve the processes in their own companies. Part of the session will be in workshop mode, and contributions from people with experience in this area are welcome.

Attendee Background: Some UML; some incremental development.

Presenter: *Alan Cameron Wills is a consultant in methods and process working in a variety of fields on both sides of the Atlantic. He is joint author of the* Catalysis Approach to Component and Object Design.

55 Component-Based Design: A Complete Worked Example

Tuesday, Afternoon
Convention Ctr — Room 24

John Daniels,
Syntropy Limited, UK

Much has been written about component-based design but most books and articles deal in generalizations and possibilities, rather than providing explicit and clear guidance. This tutorial will follow a small case study from requirements capture to code-ready specifications, and will set out the full client and server architectures needed to make it work. The modeling and specification techniques used will follow the UML, with the target technologies being EJB or COM+. A web-based UI is assumed. To complete the example, the tutorial will show how to implement the case study using EJB.

Attendee Background: The tutorial is aimed at modelers, system designers and architects. A working knowledge of UML is required, and some understanding of EJB or COM+ would be an advantage.

Presenter: *John Daniels is a consultant at Syntropy Limited, providing help with system architectures and development processes to a number of large corporations. He was previously Application and Technical Architect for Bankers Trust in London, and before that Managing Director of pioneering consulting and training company Object Designers Limited. He has applied object technology in a range of industrial and commercial applications since 1985. He has given tutorials at many object technology conferences, has prepared and delivered many training courses, and has published extensively. He is co-author of* Designing Object Systems *(Prentice-Hall 1994) and* UML Components *(Addison-Wesley 2001).*

56 Developing Java Applications for Small Spaces

Tuesday, Afternoon
Convention Ctr — Room 6

Chris Carpenter,
RoleModel Software Inc.

Chris Collins,
RoleModel Software Inc.

Java has always promised the ability to run on any size platform, from mainframes to wristwatches. Now the promise of supporting "small" platforms is truly here. This tutorial will teach the current state of the art with respect to developing Java applications for the J2ME and IBM VisualAge Micro Edition application environments (JAE's). Developers accustomed to creating applications for the web, desktop, or workstation environments will receive instruction on how to approach developing their own small-environment Java applications from pioneers who have actually attempted (successfully and unsuccessfully) to build such applications using the different environments. There are significant differences in Sun's and IBM's approach to putting Java in small spaces. The benefits and limitations of each will be discussed. The tutorial will not only discuss how to get Java applications working on small devices but will also provide practical advice about when putting Java on a small device makes sense and when it does not. In addition, the tutorial will discuss integrating Java enabled devices into the much-ballyhooed Jini environment.

Attendee Background: Participants should be Java developers or technical leaders of Java projects.

Presenters: *Chris Carpenter is a senior level software engineer and architect working for RoleModel Software, Inc. Mr. Carpenter has been involved in object oriented development since 1991. He cut his object teeth building object-oriented distributed software frameworks written in Objective-C running on NeXTs. In the early 90's he participated in architecting and building distributed systems frameworks based on the then emerging CORBA 1.0 specification. His skill at looking beyond the obvious and finding solutions to his customer's problems has always been tied to the maxim "model the world the way you want it to be." Recently, he has been involved in the design and prototype of Java in small, remote devices and their integration into infrastructures that rely upon the remote devices for system solutions. Mr. Carpenter is an author of the Automated Meter Reading System patent along with patents pending involving Java and Jini in remote devices joined to larger enterprise frameworks.*

Chris Collins is a Senior Software Developer at RoleModel Software, Inc. While at RoleModel, Chris has created an acceptance test framework, developed an embedded Java application for a new Motorola, Inc. cell phone platform, and ported JUnit to run on Sun's J2ME platform. Before joining RoleModel in early 2000, he spent five years developing software for several organizations using many different languages for U.S. Department of Defense. Chris has a Masters in Computer Science and Software Engineering from the University of West Florida, currently teaches a Java programming course at North Carolina State University, has been an invited speaker on XP at Duke University, and presented a paper on process adaptation at XP2001.

57 Patterns for Making Your Business Objects Persistent in a Relational Database World

Wednesday, Afternoon
Convention Ctr — Room 3

Joseph Yoder, *The Refactory, Inc.*

For developing simple client-server applications, development environments such as VisualAge provide a visual language for generating the mappings of GUIs to database values and domain objects. For complex applications, tools such as TOPLink are very useful for simplifying the creation of persistent objects while hiding their implementation details. Quite often, application development requires tools for persistence that fall in between these two extremes. Just using the facilities provided by JDBC is not sufficient to work with objects. JDBC forces developers to work at the SQL level with rows and columns. Application developers do not want or need to write SQL statements to read or store their objects; they are busy solving the domain problem. This tutorial will describe how to make business objects persistent by mapping them to a relational database with minimal effort. It will also examine the patterns used to map domain-objects to a relational database.

Participants of this tutorial will learn a set of patterns and a language-independent object model that can be used for mapping business objects to a relational database. They will also learn how to develop a data access layer along with the design patterns used in the database tools provided by VisualAge and TOPLink.

Attendee Background: Basic knowledge of object concepts is required. A general understanding of relational databases and/or SQL is required. An understanding of patterns can be useful, but it is not required. The examples will be in Java so understanding the basics of Java is also desirable, but not necessary to understand the object-model.

Presenter: *Joseph W. Yoder has worked on the architecture, design, and implementation of various software projects dating back to 1985. These projects have incorporated many technologies and range from stand-alone to client-server applications, multi-tiered, databases, object-oriented, frameworks, human-computer interaction, collaborative environments, and domain-specific visual languages. Recently he has taught object-oriented concepts including Patterns and Smalltalk to Caterpillar and the Illinois Department of Public Health (IDPH) analysts and developers, and has mentored many developers on the applications being deployed across the state of Illinois, such as the Newborn Screening application, the Refugee System, and the Food Drug and Dairy application. He is also coordinating these development efforts as the primary architect of the reusable frameworks being developed and used for these applications. Joe is the author of over two dozen published patterns and has been working with patterns for a long time, writing his first pattern paper in 1995, and chairing the PLoP'97 conference on software patterns.*

58 Creativity in Software Development

Wednesday, Afternoon
Marriott Hotel — Meeting Room 11

Pete McBreen, *McBreen Consulting*

Now, more than ever, software development requires innovative thinking. Our challenge has shifted from writing the code to identifying and evaluating new ideas, processes and applications. Creating software is one of the most creative activities that humans undertake. The main limitation in software is the human imagination, and the limits on that are all self imposed. Through the application of creativity, it is possible to create truly great software. This tutorial explores ideas about creativity and how they relate to software development. Specific topics that will be covered include - brainstorming techniques for eliciting requirements - creating and evaluating alternate designs - creativity and software development processes - creativity and quality assurance - creativity for programmers. Objective: On completion participants will understand how to apply creative thinking strategies to software development.

Attendee Background: Developers, team leaders and managers who need to step up to the challenges of developing great software.

Presenter: *Pete McBreen is a course designer, teacher, and project lead in object technology. He is responsible for ensuring that project teams make effective use of object technology on projects including project startup, methodology and tool selection, mentoring, process improvement, system design and quality assurance. With over 16 years of industry experience, he has been successfully using and teaching object-oriented techniques since 1989.*

59 Architectures for Integrating Business Logic into J2EE

Wednesday, Afternoon
Convention Ctr — Room 22

Josh MacKenzie, *ThoughtWorks, Inc.*
Rebecca Parsons, *ThoughtWorks, Inc.*

J2EE has exploded onto the enterprise systems development world with a handy mix of powerful technologies. They make for an impressive list of features, but they are still young and it isn't so obvious how you actually put them together. ThoughtWorks likes being on the bleeding edge, so in the last couple of years we've put into production some sizeable enterprise applications. In doing this we've learned a lot about the different architectures you can use and their relative merits. In this talk we'll walk you through half a dozen architectural designs that we've seen in the field. We'll review the issues that you need to understand to build enterprise applications and use these issues to evaluate the candidate architectures. In passing we'll discuss a number of key patterns for server-side object design.

Attendee Background: Knowledge of Java and J2EE.

Presenters: *Josh MacKenzie has been with ThoughtWorks for three years, serving as a developer, architect and team lead. He has worked on projects in equipment leasing, insurance and industrial supply and purchasing. These projects have utilized a wide variety of technologies, including J2EE, XML, Forte, and LDAP. Josh has also been instrumental in the exploration and adoption of lightweight methodologies on ThoughtWorks' projects. Prior to ThoughtWorks, Josh served as a Senior Engineer for Motorola Energy Systems, where he designed and developed real-time testing and analysis software for electrochemical capacitors. He holds a B.A. in Physics and Mathematics, and an almost-M.S. in Chemical Engineering. Josh presented tutorials at JavaCon2000 on "Refactoring" and "Business Objects in J2EE." ThoughtWorks, Inc. is a leading custom e-business application and platform development firm.*

Rebecca Parsons is a Senior Architect for ThoughtWorks, Inc., a leading custom e-business application and platform development firm. While at ThoughtWorks, Rebecca has worked on a large scale leasing system for a financial services subsidiary of a Fortune 100 company. Prior to joining ThoughtWorks, Rebecca was on the faculty of the School of Computer Science at the University of Central Florida where she taught compilers, operating systems, programming languages and computational biology. Rebecca has worked at Los Alamos National Laboratory as well as in industrial positions. She has spoken at both academic and industrial conferences and has served on program committees and editorial review boards for various conferences and publications. Rebecca received a Ph.D. in Computer Science from Rice University in 1992.

60 Planning Agile Projects

Wednesday, Afternoon
Convention Ctr — Room 16

Cara Taber, *ThoughtWorks, Inc.*

Ron Crocker, *Motorola, Inc.*

Most software projects are very poorly planned. They often have a very impressive chart on the wall describing a plan; but that plan is so out of sync with reality that it is more dangerous than useful. The painful truth is that many projects these days are faced with changing requirements, where even half way through a release cycle you still aren't sure what needs to go in the product. In such situations many principles of project planning are undermined, and if this isn't recognized planning falls apart. Despite the uncertainties, agile projects must be planned and can be controlled. In this tutorial we'll look at a simple yet effective technique that can be used to do that. The core of the ideas are based on the planning approach of XP (Extreme Programming) as described in Planning Extreme Programming. However we'll take the techniques and extend the ideas to cover a broader range of agile processes to allow the planning approaches to fit in with processes such as Crystal and RUP. The talk will cover the purpose of planning and the basic principles of XP style planning: four variables, project velocity, yesterday's weather, and division into release and iteration plans. With release planning we'll look at how requirements are chunked up into features (stories), the relationship between features and use cases, how features are estimated, how features are allocated to iterations. In iteration planning we'll look at the break down of features into tasks, allocation of tasks to people, sizing of tasks, and how an iteration is tracked. We'll look at scaling the planning process while sticking to the underlying values, based on experiences running larger projects and global multi-site development.

Attendee Background: no specific background required

Presenters: *Cara Taber has been at ThoughtWorks, Inc., an Internet professional services provider specializing in the delivery of highly strategic B2B e-Commerce solutions, for three and a half years. During that time she has been a developer, designer, analyst and project manager. In her current role as Release Plan Manager, she works closely with the team project manager focusing on planning the iterations and orchestrating the monthly 60-person all team iteration kick-off meeting. She has also published an article with Martin Fowler entitled "An Iteration in the Life of an XP Project," which appeared in the November 2000 issue of the Cutter IT Journal.*

Ron Crocker is a Senior Member of Technical Staff in the Network and Advanced Technology department in Motorola, Inc. where he is responsible for cellular system architecture and design. He has over 15 years of experience with object-oriented technologies, starting as a C++ guinea pig.

61 Designing Small Memory Software: Development Patterns for Systems with Limited Memory

Wednesday, Afternoon
Convention Ctr — Room 25

James Noble, *Computer Science, Victoria University of Wellington, NZ*

Charles Weir, *Penrillian*

Typical OO development techniques assume systems with relatively large memories. Developers working with tight memory requirements need the flexibility and encapsulation that OO can provide, but cannot afford to produce large systems. This tutorial will describe how you can use OO techniques in a memory-constrained environment. Using design patterns and practical examples, this tutorial will teach the most important techniques that successful OO designers use for small memory software. After attending this tutorial, participants will be able to:

- prepare a memory budget;
- design a software architecture and component interfaces to minimize memory use;
- track memory consumption through the development process; and
- tailor user interfaces for small software.

The tutorial balances direct presentations (for overviews and to present each pattern) and case study exercises (to reflect on patterns and see how they can be applied).

Attendee Background: This tutorial targets anyone planning, or involved in, development of OO applications in limited memory. This tutorial is most useful to developers with a year's experience using an OO language and technical team leaders. Experience of memory-limited systems is helpful but not essential.

Presenters: *Dr. James Noble has recently returned home to lecture at the Victoria University of Wellington, New Zealand. While in Sydney, he established the Sydney Patterns Group, the first patterns group in the Southern Hemisphere, and he has extensive experience lecturing, teaching, and mentoring with software design, user interface design, and design patterns.*

Charles Weir is co-founder and managing director of Penrillian, a software house specialising in components for mobile communicators. Charles has more than fifteen years' experience as a software engineer and consultant in OO techniques. He was Symbian technical lead for the Ericsson MC218 communicator project, and software architect for the Psion Series 5 Web Browser. He is co-author of the book, Small Memory Software, *and has led many courses and workshops on OO design and implementation.*

62 Reflection in Java

Wednesday, Afternoon
Convention Ctr — Room 5

Ira Forman, *IBM*

Nate Forman, *Liaison Technology*

The use of reflection is an important technique for improving productivity. Reflection facilitates development of programs that are easily adapted to requirement changes. With reflection one can develop software engineering tools that examine or produce code. Reflection facilitates testing and problem determination by facilitating the automation of more tedious tasks. In general, reflection improves the flexibility, extensibility, and reusability of one's code. The Java language contains a highly effective reflection facility. The tutorial explains the concept of reflection, the Java metaobjects (including both introspective and intercessional interfaces), the proxy class, and dynamic compilation and class loading. The limits of Java reflection are addressed in the context of what reflection is capable of in general. In addition, the tutorial demonstrates the efficacy of the Java reflection facility for solving practical problems. Such problems include: program/application testing, generation of code, inspection of code, and use of dynamic class loading in a framework for application extension.

Attendee Background: An attendee must be a competent Java programmer.

Presenters: *Dr. Ira R. Forman works for IBM in Austin. As a member of IBM's Object Technology Products Group, which produced the SOMobjects Toolkit, he worked on the SOM Metaclass Framework. He started working in the area of object-oriented programming in 1984, when he worked at ITT Programming Technology Center. Forman received his Ph.D. in Computer Science from the University of Maryland, where he studied under Harlan Mills. Forman's specialties are object-oriented programming, distributed systems, and object composition. He is the coauthor of two books,* Interacting Processes: A Multiparty Approach to Coordinated Distributed Programming *and* Putting Metaclasses to Work: A New Dimension in Object-Oriented Programming.

Nate Forman works for Liaison Technology where he designs and programs application frameworks for their products. His specialties are patterns and object-oriented programming. Forman holds a M.S.E. in Software Engineering from the University of Texas at Austin and a B.S. in Computer Science from the College of Engineering at Cornell University.

63 Ruby for the Impatient

Wednesday, Afternoon
Marriott Hotel — Meeting Room 12

Dave Thomas,
The Pragmatic Programmers, LLC

Andy Hunt,
The Pragmatic Programmers, LLC

Smalltalk was ahead of its time: we're just entering the decade of the untyped, flexible language. And by all accounts, Ruby could well be the language of that decade. Small, but tremendously expressive, Ruby is finding favor among all kinds of developers. From web applications to numerical simulations at NASA, Ruby is gaining popularity and mindshare. As a developer, you owe it to yourself to have a look at Ruby. Even if you never write a line of Ruby code, the ideas in the language can greatly improve the way you think about design and the ways you implement your programs. And if you do start writing Ruby, you'll discover the tremendous productivity and readability gains that are possible. In this tutorial, we're offering a fast-track way to learn the language, its libraries, and its philosophy. Ruby is so compact and tidy, we're confident that in just three short hours we'll have you reading and writing Ruby like an old-timer.

Attendee Background: Attendees will be familiar with the concepts of object orientation and programming. Some familiarity with a scripting language such as Perl or Python may help, but is not a requirement. Attendees who program in Smalltalk will find much of Ruby comfortingly familiar.

Presenters: *Dave Thomas is prominent in the worldwide Ruby community. He co-authored the first English-language Ruby book, runs two Ruby web sites, manages a Ruby Wiki, and is a frequent contributor to the Ruby mailing lists. He has presented Ruby in Europe and the US, in lectures, and to local user groups. Dave is a partner in The Pragmatic Programmers, a software consultancy, and co-author of* The Pragmatic Programmer.

Andy Hunt is co-author of the best-selling book, The Pragmatic Programmer, *the new* Programming Ruby, *and various articles. Between writing, traveling, woodworking and playing the piano, Andy finds time for his consulting business specializing in agile software development. Andy has been writing software professionally since the early 1980s, and currently based in Raleigh, NC. He is President of the RTP chapter of the ICCA and a member of the ACM and IEEE.*

64 Realizing Extreme Programming as a Strategic Weapon for Innovation

Wednesday, Afternoon
Convention Ctr — Room 23

Ken Auer, *RoleModel Software, Inc.*

Roy Miller, *RoleModel Software, Inc.*

Extreme Programming seems to be more appropriate in some environments than in others. This tutorial explains why Extreme Programming is particularly suitable for those in a “new product” environment using object-oriented tools and techniques. It shows both business and technical players how they can use this process to effectively address these seemingly conflicting requirements:

- getting an idea to market fast while keeping quality high
- leveraging existing assets while quickly adapting to the changing demands of the market and investors
- making the best use of key people in expanding their market reach while not sacrificing their current market
- building a cohesive team in the midst of constant change
- keeping up with leading technology while still getting current work done,

Included in this tutorial will be a participatory “Extreme Hour” simulation showing how business and technology roles work together in XP to keep development and business moving together toward a common goal at the fastest pace possible.

Attendee Background: The target audience is anyone interested in exploring a new approach to leveraging object-oriented programming, systems, and languages in the development of new software products. The assumption is that attendees will have at least heard of XP and know something about it. However, this is not a prerequisite.

Presenters: *Ken Auer is President, Founder and Master Craftsman of RoleModel Software, Inc. He has been active in the development of object oriented software since 1985. In late 1998, RoleModel Software began building the first Extreme Programming Software Studio based on his vision. This is a place where apprentices, skilled journeyman, and software masters work together in an environment of continuous learning with extremely effective modes of collaboration to produce unusually adaptable and robust software for their clients. He is also the co-author of* XP Applied, *scheduled for publication by Addison-Wesley in October 2001.*

*Roy Miller is a Software Developer at RoleModel Software, Inc. Prior to joining RoleModel, Roy spent six years with Andersen Consulting (now Accenture), most recently as a Project Manager. Roy is a contributing author to IBM’s developerWorks Java Zone, has co-authored a book in the Addison Wesley XP Series (*XP Applied, *scheduled for publication in October 2001), and was a featured panelist at the “Business of XP” fishbowl at XP2001.*

65 Advanced Extreme Programming Testing Techniques

Wednesday, Afternoon
Convention Ctr — Room 24

Joseph Pelrine, *Daedalos Consulting*

How much testing is enough? Too little? Too much? What do developers need to test? The available Extreme Programming literature differentiates between unit testing and functional testing, and gives unit testing during development a (well-deserved and much-needed) high priority, but fails to address a number of other important aspects of developer testing: GUI testing, performance testing, and packaging/delivery testing, for example. This tutorial will illustrate new techniques such as implementing “skins” for JUnit and SUnit, defining test resources for managing items which remain active over a series of tests (e.g. database connections), and automating or integrating various other tests into JUnit and SUnit. The tutorial will be only partly lecture-based. You are encouraged to present problems (and possible solutions) encountered in your work, which the group will address and attempt to solve. We’d like to give out some diff files for SUnit and JUnit, some TestCase extensions, etc., and either do a few proposed tasks, or sit down and see if we can help each other solve some of our testing problems. In order to do this, we’d like to ask you (if possible) to bring along your laptop with floppy drive, your favorite flavor of Smalltalk or Java, SUnit or JUnit, a power cable or fresh batteries, and other related stuff that you think you might need. Also, bring along some enthusiasm and “looking-for-fun” attitude, and we’re sure to have a ball.

Attendee Background: Since the tutorial is (partly) hands-on, participants should have some experience in both Smalltalk or Java and Extreme Programming.

Presenter: *Joseph Pelrine is an expert Smalltalk programmer with over 12 years extensive OT experience and has worked with Kent Beck, the originator of XP, for a number of years. A former columnist for the Smalltalk Report and noted international speaker, he is currently a senior consultant with Daedalos Consulting in Switzerland. He is coauthor of the book,* Mastering ENVY/Developer, *recently published by Cambridge University Press.*

66 C++ Idioms

Wednesday, Afternoon
Convention Ctr — Room 13

Jan Christiaan van Winkel, *AT Computing*

This tutorial will address non trivial C++ programming constructs that experienced programmers use frequently. These constructs are too small to be called patterns, but they are part of the language skills of the proficient C++ programmer. Therefore we call them idioms. Several idioms will be discussed, such as:

Traits. Traits classes are used frequently in the C++ standard. Some libraries are built around traits classes, for example the Boost library (www.boost.org). Using traits, it is possible to get hold of information about a type at compile-time, that will then influence behavior at runtime.

Intermediate objects. These are often used as proxy objects to write something in C++ that the syntax disallows, such as a[1][2] where a is of class type. They exist only in the expression in which they are used.

Resource management through the “Resource Acquirement is Initialization” idiom. Not only allocated memory has to be returned. If a function needs to change a formatting flag in an ostream, that change has to be undone when the function ends.

After the tutorial the attendee will recognize several idioms, and know when to use them. As with any language, knowing idioms will improve your fluency.

Attendee Background: The attendees are expected to know C++.

Presenter: *JC van Winkel has a B.S. and an M.S. in computer science (the M.S. from the Vrije Universiteit Amsterdam). He works at AT Computing, a small courseware and consulting firm in Nijmegen, the Netherlands. There he teaches UNIX and UNIX-related subjects, including C and C++. Except for 1995, J.C. van Winkel has presented tutorials at all OOPSLAs since 1993. He is the Dutch representative in the ISO C++ standardization committee SC22/WG21.*

67 Patterns and Techniques for Developing Performance Effective Enterprise Java Beans

Wednesday, Afternoon
Marriott Hotel — Meeting Room 1

Matjaz B. Juric, Ph.D., *Assistant Professor*

In the tutorial we will present patterns and techniques for the design and implementation of performance effective Enterprise Java Beans (EJB) components. We will substantiate our discussion with real-world examples and performance measurements for different scenarios. We will look at the basic facts regarding EJBs and performance, explain the underlying concepts, discuss the remote method invocations, fine and coarse grained interfaces, take a look at the value objects and input validation on the client side, learn what is the throughput problem, present the advantages of the facade pattern, discuss the dependent objects, take a look at the instance management algorithms in terms of performance, show, how to manage persistence, transactions, concurrency and how to avoid deadlocks, learn how to lazy load enterprise beans and reuse resources, discuss the advantages of smart stubs and show how to accelerate marshaling, learn how to tune the performance when deploying EJBs, give practical guidelines for achieving scalability, etc. We will focus on the performance relevant changes in the EJB 2.0 specification, particularly on local interfaces, home methods, new container managed persistence schema, and relationships. Sound design alone is not sufficient for good performance. There are performance differences hidden in the application servers as well. Therefore we will present performance measurement results with different application servers.

Attendee Background: Participants should be familiar with OO concepts, distributed component models, Java language, Enterprise Java Beans (EJB) and possibly with Java 2 Enterprise Edition.

Presenter: *Matjaz B. Juric holds a Ph.D. in computer and information science. Currently he is an Assistant Professor at University of Maribor, Faculty of Electrical Engineering and Computer Science. His Ph.D. work has received an award from the Slovenian IEEE Section and he participated in the OOPSLA Doctoral Symposium in 1999. He received several awards for articles and an award for his B.Sc. work (from Slovenian Society for Informatics). His research area covers distributed systems and object technology, with special emphasis on Java, distributed object systems (EJB, RMI, RMI-IIOP, CORBA, COM+), component development, and performance. He has been involved in performance analysis and optimization by the development of RMI-IIOP, an integral part of Java 2 platform, in cooperation with IBM Java Technology Centre, Hursley, UK. Juric has published more than 140 publications, and twelve original scientific papers. He has published a chapter in the book,* More Java Gems, *(Cambridge University Press) and has written a chapter on performance in the upcoming Wrox,* Professional EJB Development *book.*

68 Pair Programming: Experience the Difference

Wednesday, Afternoon
Convention Ctr — Room 15

Laurie Williams,
North Carolina State University

Robert Kessler,
University of Utah

Pair programming is emerging as an important technique for developing higher quality code, faster. With pair programming, two software developers work on one computer, collaborating on the same design, algorithm, code, or test. This tutorial examines pair programming research results and anecdotal experiences of programmers who have transitioned to pair programming. It will discuss what works and what doesn't and will also explain techniques for fostering support in making a transition to pair programming, support from management, and support from peers. Hands-on activities will be used to demonstrate pair programming benefits. Participants will experience the difference between working alone and working in pairs. They will understand the research results that show pair programming works, learn how to pair program, what not to do when pairing, and how to transition to pair programming.

Attendee Background: This tutorial is targeted toward software developers and technical software development managers who are interested in transitioning to pair programming.

Presenters: *Dr. Laurie Williams is an assistant professor at North Carolina State University. In 2000, she completed her dissertation which demonstrated statistically that pair programmers were able to produce higher quality products in essentially half the time when compared to individual programmers. Prior to her recent academic career, Laurie worked at IBM for nine years. Laurie and Bob are collaborating on a book entitled,* Pair Programming Illuminated, *to be published by Addison-Wesley in 2002.*

Dr. Robert Kessler is a Professor of Computer Science and served as the last chairman of the University of Utah, Department of Computer Science (the department is now known as the "School of Computing"). He has founded several companies and served on the board of directors of others. Bob is an award-winning instructor having recently received the 2000 College of Engineering, Outstanding Teaching Award and the 2001 University of Utah, Distinguished Teaching Award.

69 Objects vs. The Web

Wednesday, Afternoon
Convention Ctr — Room 14

Alan Knight,
Cincom Systems of Canada

Naci Dai,
BEA Systems Inc.

Web development is the cool new paradigm. How do we stop ourselves from forgetting the lessons of previous paradigms and just hacking our way through? The web's core mechanisms lend themselves all too easily to cut-and-paste re-use, ad-hoc scripts, direct-to-database code, and fragmented business logic. In the name of time-to-market, too many of us abandon what we know and take the path of least resistance. There are capable and articulate people telling us this is exactly what we should be doing. We believe, that the lessons of objects do apply to the web. If we apply them wisely we can have good time-to-market for version 1.0, ship version 2.0 successfully, and improve the user experience along the way. But it's not obvious how to apply these lessons. What, if anything, is MVC for the web? What are the architectural layers? How do we support multiple channels within these layers, when presentation differences can easily creep into the domain logic? What about EJB? This tutorial surveys current web technologies with an emphasis on OO usage, provides best practices and examples from Java and Smalltalk, discusses myths and truths about components, and describes architecture and development practices that support good practices.

Attendee Background: Attendees should have a reasonable understanding of OO development. Experience with web development is helpful, although a basic familiarity with terms is adequate.

Presenters: *Alan Knight works on Smalltalk web tools for Cincom Systems. Prior to that he was chief architect of the TOPLink family of object-relational mapping products with The Object People and WebGain. He has spoken extensively at conferences including OOPSLA, Smalltalk Solutions and Java One, and is co-author of the book,* Mastering ENVY/Developer. *He can be reached at knight@acm.org.*

Naci Dai is an educator for BEA Systems Inc., prior to that he was the director of distributed computing with The Object People. He teaches object technology, design-patterns, and distributed computing. He leads and mentors web development projects for Fortune 500 companies. He has developed the distributed computing curriculum and services. He has a background in applied-engineering and computational physics. He has received his Ph.D. from Carleton University. He can be reached as nacidai@acm.org.

70 OPEN: A Flexible OO/CBD Process for Software-Intensive Systems Development

Wednesday, Afternoon
Marriott Hotel — Meeting Room 2

Brian Henderson-Sellers,
University of Technology, Sydney

The increased complexity associated with large-scale software-intensive systems development requires an increase in the sophistication of the methodology utilized. Following a general discussion on the value of processes, one specific OO/CBD example, OPEN (Object-oriented Process, Environment and Notation) is described in detail. Emphasis will be placed upon the need for flexibility of processes and how they can be constructed and configured to individual circumstances. Finally, some advice on how to transition to OO/CBD and deploy this process for the first time will be given. Objective: The tutorial objective is to present and understand the need for flexibility in process and how this can lead to an organizationally or project specific process instance using the OPEN Process Framework as an example. A secondary goal is to discuss how transition to an OO environment can be accomplished.

Attendee Background: Fully conversant with basic OO terminology and the need for a full lifecycle process. Experience with OO methodologies is advantageous. Those who would benefit most, would include project managers, systems developers, analysts and designers.

Presenter: *Brian Henderson-Sellers is Director of the Centre for Object Technology Applications and Research and Professor of Information Systems at University of Technology, Sydney (UTS). He is author of nine books on object technology and is well-known for his work in OO methodologies (MOSES, COMMA and OPEN) and in OO metrics. In 1999, he was voted number 3 in the Who's Who of Object Technology (Handbook of Object Technology, CRC Press, Appendix N). He is currently a member of the Review Panel for the OMG's Software Process Engineering Model (SPEM) standards initiative.*

NOTES: