Explore; Discover; Understand

ooPSLA

Montréal, Canada 2007

Welcome to ooPSLA 2007. I used to go to OOPSLA for the objects—back in the 1980s when there was lots to find/figure out about objects and how that approach—oop—related to programming in general. Nowadays objects are mainstream and I go for the programming. I love programs and programming. I laugh when people try to compare programming to something else, such as: "programming is like building a bridge" or "programming is like following a recipe to bake a soufflé." I laugh because programming is the more fundamental activity—people should be comparing other things to it: "writing a poem is like programming an algorithm" or "painting a mural is like patching an OS while it's running." I write programs for fun the way some people play sudoku or masyu, and so I love to hear and learn about programs and programming.

More than that, I come for the creative and dedicated people—their ideas, their stories, the work they do, the explorations and research they report—for the social events, for catching up, for meeting new people / all ages.

By tradition the general chair writes a very professional welcome. Last year's chair, Peri Tarr, did a great job, so I've reproduced and just slightly revised her welcome note right here:

> Welcome to ooPSLA [2007] the premier gathering of professionals from industry and academia, including practitioners, researchers, students, and educators—all sharing their experiences with object technology and its offshoots for the 21st year. ooPSLA offers an extraordinary array of venues and activities: presentations from invited speakers, research papers, essays, practitioner reports, expert panels, demonstrations, formal and informal educational symposia, workshops, and diverse tutorials from world-class lecturers. The popular Onward! track presents out-of-the-box thinking at the frontiers of computing. Posters discuss late-breaking results, culminating in the [Sixth] Annual SIGPLAN Student Research Competition. Lightning Talks allow anyone to speak to the community about anything on their minds. Like-minded people can gather informally and spontaneously at Birds-of-a-Feather sessions. And of course there are plenty of social opportunities for mingling and professional networking. That's just a sampling of what makes ooPSLA the conference of choice for software technologists, from recognized academics to undergraduate students, from industrial researchers to developers and managers, from the creators of technology to its users.

> *tomorrow*
> *we shall have to think up signs,*
> *sketch a landscape, fabricate a plan*
> *on the double page*
> *of day and paper.*
> *Tomorrow, we shall have to invent,*
> *once more,*
> *the reality of this world.*

—from "January First," Octavio Paz
translated by Elizabeth Bishop

Richard P. Gabriel, Conference Chair

Gold Supporters

IBM Research is the world's largest information technology research organization, with more than 3,000 scientists and engineers at eight labs in six countries. IBM has produced more research breakthroughs than any other company in the IT industry. For more information on IBM Research, visit www.research.ibm.com.

Silver Supporters

Founded in 1975, Microsoft is the worldwide leader in software, services, and Internet technologies for personal and business computing. The company offers a wide range of products and services designed to empower people through great software—any time, any place, and on any device. Microsoft Visual Studio Team System is the comprehensive tool for building the right team and fostering a culture of continuous improvement. For more information, please visit www.microsoft.com.

Bronze Supporters

Google is a global technology leader focused on improving the way people connect with information. Google's innovations in web search and advertising have made its website a top Internet destination and its brand one of the most recognized in the world. Google maintains the world's largest online index of websites and other content, and Google makes this information freely available to anyone with an Internet connection. Google's automated search technology helps people obtain nearly instant access to relevant information from it vast online index. For more information, visit www.google.com.

Friends

HP is a technology solutions provider to consumers, businesses, and institutions globally. The company's offerings span IT infrastructure, global services, business and home computing, and imaging & printing. Visit us at www.hpl.hp.com.

Cisco enables people to make powerful connections-whether in business, education, philanthropy, or creativity. Cisco hardware, software, and service offerings are used to create the Internet solutions that make networks possible-providing easy access to information anywhere, at any time. With more than 60,000 employees world-wide, Cisco's long tradition of innovation continues with industry-leading products and solutions in the company's core areas of routing and switching, as well as in advanced technologies such as: application networking; data center; digital media; IPICS mobility; security; storage networking; TelePresence; Unified Communications; and video.

Sun Microsystems Laboratories (Sun Labs) is Sun's applied research and advanced development arm. For more information about Sun Labs, please visit: www.research.sun.com.

Knowledge, information and quality—these are the three things that shape Springer Science+Business Media's business activities. Springer develops, manages, and disseminates knowledge—through books, journals ,and the Internet. Springer works with the world's best academics and authors in long-standing loyal partnerships based on mutual trust and always open to new input.

Room	8:30–12:00	13:30–17:00
510A–D	Wiki Symposium	
512E–G	Mini PLoP Bootcamp	
513E	Library-Centric Software Design	
514A	APL Tutorial	
515A	HPC-GECO/CompFrame	
516B	DesignFest®	
524A–C	International Symposium on Memory Management	
Tutorials	**T1: Software Evolution: Analysis and Visualization** Harald Gall, Michele Lanza 512C **T2: The Web: Distributed Objects Realized!** Stuart Charlton, Mark Baker 513B **T3: Revolutionizing Software Quality through Static Analysis Tools** Jonathan Aldrich 513C **T4: Planning and Executing Agile Projects in Non-Agile Environments** Peter Schuh 513D **T5: High Quality Software Architecture** Michael Stal 518A **T6: Principles of Aspect-Oriented Design in Java and AspectJ** Dean Wampler 518B **T7: Unit Test Patterns and Smells: Improving Test Code and Testability Through Refactoring** Gerard Meszaros 518C	**T8: From Use to User Interface: Collaboratively designing, prototyping, and testing user interface to help your users succeed** Jeff Patton 512D **T9: Pick Up Your Pen!** Steve Metsker 513B **T10: Object-Oriented Models of Requirements and High-level Software Design** Hermann Kaindl 513F **T11: Best Practices for Model-Driven Development** Markus Voelter 514B **T12: Software Architecture Refactoring** Michael Stal 518A **T13: LINQ From the Source** Erik Meijer, Ted Neward 516C **T29: Using FindBugs in Anger** David Hovemeyer, William Pugh 518C
Workshops	**W1: Process in oo Pedagogy—The 6th 'Killer Examples' workshop** Carl Alphonce, Jürgen Börstler, Michael Caspersen, Adrienne Decker, Michael Kölling 512B **W2: Managing Complexity** Klaus Marquardt, Lise Hvatum, Jens Coldewey 512H **W3: 1st International Workshop on In Process Software Engineering Measurement and Analysis** Philip Johnson, Alberto Sillitti 514C **W4: 1st International Workshop on Pattern Languages: Addressing Challenges** Mohamed E. Fayad, Chia-chu Chiang, Huascar A. Sanchez, Pablo Chacin, A. Kannammal 515B **W5: 5th International Workshop on SOA & Web Services Best Practices** Olaf Zimmermann, Anders Aas Bjerkestrand, Dr. Amir Zeid, Lars Arne Skaar 516D **W6: 7th OOPSLA Workshop on Domain-Specific Modeling** Juha-Pekka Tolvanen, Jeffrey G. Gray, Matti Rossi, Jonathan Sprinkle 519A **W7: Eclipse Technology Exchange 2007** Li-Te Cheng, Alessandro Orso, Martin Robillard 519B	

Room	8:30–12:00	13:30–17:00
510A–D	Wiki Symposium	
512D	Doctoral Symposium	
514AB	APL 2007	
515A	HPC-GECO/CompFrame	
516A–C	Educators' Symposium	
516DE	Dynamic Languages Symposium	
517C, 520B–E	Mini PLoP	
524A–C	International Symposium on Memory Management	
Tutorials	**T14: Security Patterns and Secure Software Architecture** Munawar Hafiz 512F **T15: Java API Design** Boris Bokowski 513A **T16: Test-Driven Development—Hands-on!** Niclas Nilsson, Jimmy Nilsson 513B **T18: Software Architecture Documentation in the Real World** Markus Voelter, Michael Kircher 518A **T19: Introduction to Concurrent Programming in Java** David Holmes, Joe Bowbeer 518B **T20: The Scala Experience—programming with functional objects.** Marttin Odersky, Ted Neward, Gilles Dubochet 518C	**T22: Building Embedded and Stand-alone Domain Specific Languages: Principles and Practise** Eric Van Wyk 512F **T23: Using Java ME to Program Your Mobile Phone** Jonathan Knudsen 513A **T24: The Art of Telling Your Design Story** Rebecca Wirfs-Brock 513B **T25: Integrating Architecture-Centric Methods into Object-Oriented Analysis and Design** Raghvinder Sangwan 513C **T26: Why Users Say, "Start with the Screen!": Effective Test-Driven Development, Presentation Layer-First** Bobby Norton, Chris Stevenson 513D **T27: SOA in Reality** Nicolai Josuttis 518A **T28: Java Concurrency Utilities in Practice** David Holmes, Joe Bowbeer 518B **T40: Software Security: Building Security In** Gary McGraw 518C
Workshops	**W6: The 7th OOPSLA Workshop on Domain-Specific Modeling** Juha-Pekka Tolvanen, Jeffrey G. Gray, Matti Rossi, Jonathan Sprinkle 519A **W8: Integration of Open Source Components into Large Software Systems** Michael Weiss, Tony Bailetti, Peter Carbone 512A **W9: Versions, Releases, and Distribution** Klaus Marquardt, Lise Hvatum 512B **W10: The Popularity Cycle of Graphical Tools, UML, and Libraries of Associations.** Jiri Soukup, Martin Soukup 512C **W11: The 1st Workshop on Programming Languages and Integrated Development Environments** Sean McDirmid, Robert Fuhrer, Julian Dolby, Eugene Vigdorchik 512E **W12: No Silver Bullet—a Retrospective on the Essence and Accidents of Software Engineering** Steven Fraser, Dennis Mancl, Bill Opdyke 512G **W13: Semantic-Based Systems Development** Sergio De Cesare, Grant Holland, Carsten Holtmann, Mark Lycett 512H **W14: The 1st International Workshop on Unified Data Mining Engine: Addessing Challenges** Mohamed E. Fayad, Tarek Helmy, Somenath Das, Eduardo M. Segura 514C	
Evening	**Welcome Reception 17:00–19:00** 710	

Room	8:30–10:00	10:30–12:00
517A	**Once Upon a Time, Like Never Before: The Challenge of Telling the Next Story** Peter Turchi	**Creating Passionate Users** (tentative) Kathy Sierra
517B		**Research: Growing Java** The JastAdd Extensible Java Compiler Jeannie: Granting Java Native Interface Developers Their Wishes ILEA: Inter-Language Analysis across Java and C
516AB		**Essay** Confessions of a Used Programming Language Salesman Erik Meijer
517C		
516CD		**Panel** Celebrating 40 Years of Language Evolution: Simula 67 to the Present and Beyond Steven Fraser, James Gosling, Anders Hejlsberg, Ole Lehrman Madsen, Bertrand Meyer, Guy L. Steele Jr.
516E	DesignFest®	
510A–D	Wiki Symposium	
514AB	APL	
Tutorials	**T30: Incremental Releases Users & Stakeholders Will Love** Jeff Patton 515C	

Room	10:00–12:00	15:00–17:00
512EF	Demos	Demos
512GH	Demos	Demos

13:30–15:00	15:30–17:00	Room
Second Life: The World's Biggest Programming Environment Jim Purbrick & Mark Lentczner	**Onward!** Living in the Comfort Zone Living it up with a Live Programming Language	**517A**
Research: Runtime Techniques / GC Statistically Rigorous Java Performance Evaluation MicroPause: Proactively Invoking Garbage Collection for Improved Performance Probabilistic Calling Context	**Research: Inheritance and Visibility (15:30–17:30)** Variant Path Types for Scalable Extensibility Dependent Classes Component NextGen: A Sound and Expressive Component Framework for Java User-Changeable Visibility: Resolving Unanticipated Name Clashes in Traits	**517B**
Practitioner Reports Effective Preparation for Design Review—Using UML Arrow Checklist Leveraged on the Gurus Knowledge Understanding the Value of Program Analysis Tools		**516AB**
	Invited Talk: Brian Marick "Actor-Network Theory: Nothing to Do with TCP/IP or Distributed Objects"	**517C**
Panel Domain Specific Languages—Another Silver Bullet? Henry Balen, James Lapalme, Marc Frappier, Kevin P. Tyson		**516CD**
DesignFest®		**516E**
Wiki Symposium		**510A–D**
APL		**514AB**
T31: Green Bar for C++—Unit Testing and Refactoring C++ Peter Sommerlad 512C **T32: Writing Adaptable Software: Mechanisms for Implementing Variabilities in Code and Models** Markus Voelter 514C **T33: Software Best-Practices: Agile Deconstructed** Steven Fraser, David Lorge Parnas 515B **T34: The WS-* Jungle** Michael Stal, Joerg Bartholdt 515C		**Tutorials**

17:15–18:30	20:00–21:30	Room
Demo: Collaborating with Jazz: A Concurrent Demonstration at OOPSLA 2007 and CASCON Erich Gamma, John Wiegand, and Team, IBM Rational (light refreshments)		**516AB**
	50 in 50 Guy L. Steele Jr. Richard P. Gabriel	**517A**

Room	8:30–10:00	10:30–12:00
517A	**Collaboration and Telecollaboration in Design** Frederick P. Brooks, Jr.	**Onward!** From 0 to 1,169,600 in 3 minutes PowerPoint and Complexity The Elephant in the Room X3D Web Software Visualization in Action!
517B	**Research: Language Design** Transactions with Isolation and Cooperation StreamFlex—High-throughput Stream Programming in Java Can Programming be Liberated from the Two-Level Style?—Multi-Level Programming with DeepJava	**Research: Software Design** The Causes of Bloat, The Limits of Health Notation and Representation in Collaborative Object-Oriented Design WebRB: Evaluating a Visual Domain-Specific Language For Building Relational Web-Applications
516AB		**Essay** The Transactional Memory / Garbage Collection Analogy Dan Grossman
516CD		**Panel (11:00–12:30)** "No Silver Bullet" Reloaded —A Retrospective on "Essence and Accidents of Software Engineering" Steven Fraser, Frederick P. Brooks, Jr., Martin Fowler, Ricardo Lopez, Aki Namioka, Linda Northrop, David Lorge Parnas, David Thomas
510D		
Tutorials	**T35: Scripting Your (and Your Company's) Second Life** Cristina Lopes, William Cook 510A **T36: Agile in Face of Global Software Development** Jutta Eckstein 514A **T37: Use-Case Patterns and Blueprints** Gunnar Overgaard, Karin Palmkvist 514B **T38: Embedded Objects with C++: Idioms, Patterns and Architecture for Constrained Systems** Detlef Vollmann 515A **T39: Pattern-Oriented Software Architecture: A Pattern Language for Distributed Computing** Douglas Schmidt 515B	
Workshops	**W15: Agility Unlimited?** Klaus Marquardt, Jens Coldewey, Johannes Link 514C	

13:30–15:00	15:30–17:00	Room
Elephant 2000: A Programming Language Based on Speech Acts John McCarthy	**Precise Software Documentation: Making Object-Orientation Work Better** David Lorge Parnas	**517A**
	Research: Type and Typestate (15:30–17:30) Modular Typestate Checking for Aliased Objects Type Qualifier Inference for Java Establishing Object Invariants with Delayed Types Modular Verification of Higher-Order Methods with Mandatory Calls Specified by Model Programs	**517B**

	Practitioner Reports
	So We Thought We Knew Money
	Anticorruption: A Domain-Driven Design
	   Approach to More Robust Integration
	Agile Enterprise Software Development Using
	   Domain-Driven Design and Test First
		516AB

	Panel The Role of Objects in a Services-obsessed World John Tibbetts, Ward Cunningham, Carl Lentz,Jeroen van Tyn	**516CD**
Lightning Talks		**510D**
T41: Behaviour Driven Development using JBehave Elizabeth Keogh, Dan North 514A		**Tutorials**
T42: Fault Tolerant Software with Patterns Robert Hanmer 514B		
T43: Testing: It's Part of Your Job—Make it Effective! Neil Harrison 515A		
T44: Skills for the Agile Designer Rebecca Wirfs-Brock 515B		
W15: Agility Unlimited? Klaus Marquardt, Jens Coldewey, Johannes Link 514C		**Workshops**

12:00–13:30	19:00–???	
BoF: Collaborative Development Research using Jazz Harold Ossher		**516AB**
	Special Event	

Room	8:30–10:00	10:30–12:00
517A	**Context, Perspective, and Programs** Gregor Kiczales	**Onward!** No Ifs, Ands, or Buts: Uncovering the Simplicity of Conditionals Epi-Aspects: Aspect-Oriented Conscientious Software
517B	**Research: Isolation and Repair** Using Early Phase Termination To Eliminate Load Imbalances At Barrier Synchronization Points STARC: Static Analysis for Efficient Repair of Complex Data Tracking Bad Apples: Reporting the Origin of Null and Undefined Value Errors	**Research: Ownership** Inferring Aliasing and Encapsulation Properties for Java Multiple Ownership Ownership Transfer in Universe Types
517C		**Invited Talk: Mark Bernstein** Intimate Information for Everyone's Everyday Tasks: What hyperfiction and new media theory teach us about programming (and teenage diaries)
516CD		**Panel** The Future of SOA: What worked, what didn't and where is it going from here Mamdouh Ibrahim, Kerrie Holley, Nicolai M. Josuttis, Brenda Michelson, Dave Thomas, John deVadoss
510D		
Tutorials	**T45: Domain-Driven Design: Putting the Model to Work** Eric Evans 510B **T46: Real-time Programming on the Java Platform** David Holmes, Tony Printezis 514B **T47: Totally Awesome Computing: Python as a General-purpose Object-oriented Programming Language** Chuck Allison 514C **T48: Introduction to Software Product Line Development Methods** Charles Krueger 515B	

Room	10:00–12:00	15:00–17:00
512EF	Demos (10:30–12:00)	Demos (15:00–16:30)
512GH	Demos	Demos

13:30–15:00	15:30–17:00	Room
Meta-objects for the world around us Pattie Maes	Awards	**517A**
Research: Language Specification Lost in translation: Formalizing proposed extensions to C# The Java Module System: core design and semantic definition AWESOME: A Co-Weaving System for Multiple Aspect-Oriented Extension	**Research: Runtime Techniques (15:30–17:30)** Scalable Omniscient Debugging Using HPM-Sampling to Drive Dynamic Compilation MOP: An Efficient and Generic Runtime Verification Framework Making Trace Monitors Feasible	**517B**
Practitioner Reports A Practical, High-Volume Software Product Line Making Frameworks Work: A Project Retrospective	**Invited Talk: Brian Foote** Big Balls of Muds: An introduction to Post-Modular Programming	**517C**
		516CD
Lightning Talks		**510D**
T21: Personas, Profiles, Actors, & Roles: Modeling users to target successful product design Jeff Patton 510C **T49: Creating Plugins and Applications for the Eclipse Platform** Alex Romanov, Amir Kirsh 510A **T50: Domain-Driven Design: Strategic Design for Larger Systems** Eric Evans 510B **T51: Introduction to Ruby** Glenn Vanderburg 514A **T52: Building Service-Oriented Architectures with Web Services** Olaf Zimmermann 514B **T53: UML in Action** Mohamed E. Fayad 514C		**Tutorials**

	17:00–18:00	
	Ice Cream Social	**517D**

Keynotes and invited talks provide perspective and insights intended to frame your experiences at ooPSLA. The best talks should challenge your thinking and reveal your (hidden) assumptions.

Keynote: Once Upon a Time, Like Never Before: The Challenge of Telling the Next Story	Tuesday Oct 23, 08:30–10:00	517A

Readers turn to narrative for certain familiar pleasures; and yet, reading the opening sentences, they hope to find themselves in unknown territory. They want to be lost in a book, transported through a shared act of imagination. If what they read seems too strange, though, if they start to feel truly lost, they're likely to feel anxious, frustrated, even angry. The challenge for the writer, then—the challenge for every discoverer and creator—is to communicate with the past, while guiding the reader (or follower, or user) someplace new.

Using examples from writing and cartography, this talk will explore the challenges of discovery, the challenges of presenting those discoveries, and how the presentation itself is often the key to discovery (think Impressionism). It will also consider the tension between intention and inspiration, or good luck. Columbus was headed for India, James Cook mapped the Pacific only because he couldn't find Terra Australis, and both Mark Twain (soon after publishing *The Adventures of Huckleberry Finn*) and F. Scott Fitzgerald (in the weeks following the publication of *The Great Gatsby*) expressed despair over their failure to write the book they thought they meant to write. The thing they had discovered—the thing they had created—transcended their own conception of a "good book." Before we can lead anyone anywhere, we need to look clearly at where we are, and to prepare ourselves to see like never before.

http://www.peterturchi.com/

Peter Turchi, Warren Wilson College

Peter Turchi is the author of four books: a novel, "The Girls Next Door"; a collection of stories, "Magician"; a book of non-fiction, with Barry Clifford, "The Pirate Prince"; and "Maps of the Imagination: The Writer as Cartographer." He has also co-edited, with Charles Baxter, a collection of essays by Warren Wilson MFA fiction faculty, "Bringing the Devil to His Knees: The Craft of Fiction and the Writing Life" and, with Andrea Barrett, a fiction anthology by Warren Wilson MFA faculty, "The Story Behind the Story." His short fiction has appeared in Ploughshares, Story, Alaska Quarterly, and The Colorado Review, among other magazines; it has been anthologized, nominated for the Pushcart prize, and cited by the editors of Best American Short Stories. His awards include a National Endowment for the Arts Fellowship Grant, North Carolina's Sir Walter Raleigh Award, and an Illinois Arts Council Literary Award. He has taught at the University of Arizona, Northwestern University, Columbia College, and the Breadloaf Writers conference. He has directed and taught in Warren Wilson's MFA Program for Writers since 1993. He holds an M.F.A from the University of Arizona and a B.A. from Washington College.

Keynote: **Second Life: The World's Biggest Programming Environment**　　**Tuesday** **Oct 23, 08:30–10:00**　　**517A**

Second Life is large, on-line virtual world where avatars dance, fly, buy virtual clothing, play games, have meetings…and program. About 256k residents of Second Life write code that runs 24/7 in over 2M simulated objects in a continuous 3D landscape twice the size of Montréal. This giant, collaborative development environment is run on a large grid of over 12k CPUs in a grid of "simulators" that run the land of Second Life. The simulators have an integral virtual machine for the scripting language people use. Despite the inherit difficulties, the system demonstrably does enough right to enable development of a huge amount of content in Second Life. As the virtual world has grown, we have been evolving its infrastructure for programming in several ways. Integration of the Mono virtual machine presented a huge set of challenges but offers major advantages as Second Life grows. We have also had to architect and extend in light of the fact that Second Life is a continuously running system on which over a million people rely. Finally, apart from the language and run-time environment, Second Life also presents a social environment in which to program collaboratively. Within Linden Lab, we have pioneered the use of Second Life as integral part of our development methodology even when working on the underlying code of Second Life itself. These experiences point toward a re-imagining of programming as a globally immersive collaborative experience.

Jim Purbrick & Mark Lentczner, Linden Lab

Dr Jim Purbrick has both academic and industry experience in designing and building virtual worlds. At Nottingham University he worked on the MASSIVE-3 virtual environment system and Prix-Ars-Electronica–winning mixed-reality games with IGDA award winners, Blast Theory. In industry Jim designed online games at Codemasters, developed networking and load balancing technology for Warhammer Online, and is currently working on scripting and networking technology for Second Life while setting up Linden Lab Brighton.

Mark Lentczner directs a software engineering studio at Linden Lab. His studio is primarily focused on the architectural extension of Second Life and the software infrastructure to support its expansion to Internet scale. He appears in Second Life as "Zero Linden." Mr. Lentczner has worked in Silicon Valley for over 20 years, leading engineering teams on projects including virtual machines, software tools, cell phone browsers, and audio processing. He held leadership positions at Apple Computer, OpCode Systems, and Go Corporation before running his own consulting firm for a decade. He is a graduate of Harvard with a degree in Applied Math and Music.

| Invited Speakers: 50 in 50	**Tuesday** **Oct 23, 20:00–21:30**	**517A**

Languages—what's to learn from them? Relics of the past; we know how to design them / to use them. Types / messages / invocation / loops / numbers / methods / big ol' libraries / lots of = signs. Heh, but what is programming, and what roles do programming languages play in that process? We have learned a lot over the last five decades: organizing principles, established conventions, theory, fashions, and fads. "Those who cannot remember the past are condemned to repeat it." In this talk we survey what we think are the most important lessons of the past that future programmers—and future programming language designers—ought not forget. We illustrate each lesson by discussing specific programming languages of the past, and endeavor to shine what light we can on the future.

Guy L. Steele Jr, Sun Microsystems Laboratories
Richard P. Gabriel, IBM Research

Sun Fellow Guy Steele is a researcher for Sun Microsystems Laboratories, working on the Programming Language Research project. His research interests include Algorithms, Compilation, Distributed Systems, High Performance Computing, Java, Lisp, Scheme, Object Oriented Programming, Operating Systems, Programming Languages, Software, and Supercomputer design.

Richard P. Gabriel is a Distinguished Engineer at IBM Research, looking into the architecture, design, and implementation of extraordinarily large, self-sustaining systems. He is the award- winning author of four books and a poetry chapbook. He lives in California.
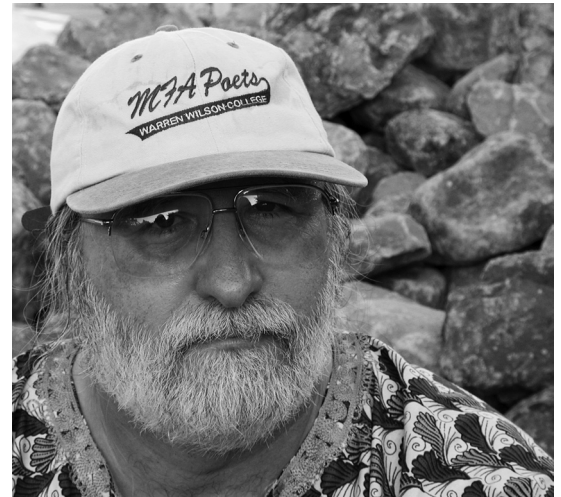
Invited Speaker: Collaboration and Telecollaboration in Design	**Wednesday** Oct 24, 08:30–10:00	**517A**

A new characteristic of design in the 20th century is the dominant use of teams to do design. We design with teams both because we are in a hurry and because our creations require more skills than one mind can master. Yet we want our designs to have excellence, and that requires conceptual integrity. Achieving conceptual integrity in team design is then a formidable challenge. Telecollaboration is now, in the 21st Century, not only possible but even fashionable. The mantra of "telecollaboration" assumes implicitly that collaboration is a good thing per se. The more one collaborates, the better. This is far from self-evident; it probably is not true. Nevertheless, there are parts of the design process where collaboration not only shares out the work, but also produces a better design. Here telecollaboration can be most fruitful. Analysis of these aspects of design inevitably generates opinions on how design should be done and taught.

Frederick P. Brooks, Jr. University of North Carolina at Chapel Hill

Frederick P. Brooks , Jr. received an A.B. summa cum laude in physics from Duke and a Ph.D. in computer science from Harvard. He joined IBM, working in Poughkeepsie and Yorktown, NY, 1956–1965. He was an architect of the Stretch and Harvest computers and later the project manager for the development of IBM's System/360 family of computers. For this work he received a National Medal of Technology jointly with Bob O. Evans and Erich Bloch Brooks. With Dura Sweeney in 1957, Brooks patented an interrupt system for the IBM Stretch computer that introduced most features of today's interrupt systems. He coined the term computer architecture. His System/360 team first achieved strict compatibility, upward and downward, in a computer family. His early concern for word processing led to his selection of the 8-bit byte and the lowercase alphabet for the System/360, engineering of many new 8-bit input/output devices, and introduction of a character-string data type in the PL/I programming language. In 1964 he founded the Computer Science Department at the University of North Carolina at Chapel Hill and chaired it for 20 years. Currently, he is Kenan Professor of Computer Science. His principal research is in real-time, three-dimensional, computer graphics—"virtual reality." Brooks distilled the successes and failures of the development of Operating System/360 in The Mythical Man-Month: Essays in Software Engineering, *(1975, 20th Anniversary Edition, 1995). He further examined software engineering in a 1986 paper,* No Silver Bullet. *He is a member of the National Academy of Engineering, the National Academy of Science, and the American Academy of Arts and Sciences. He has received the ACM A.M. Turing Award, the IEEE John von Neumann Medal, the IEEE Computer Society's McDowell and Computer Pioneer Awards, the ACM Allen Newell and Distinguished Service Awards, the AFIPS Harry Goode Award, and an honorary Doctor of Technical Science from the Swiss Federal Institute of Technology*

Keynote: **Elephant 2000: A Programming Language Based on Speech Acts**   **Wednesday** Oct 24, 13:30–15:00   **517A**

Elephant 2000 is a proposed programming language good for writing and verifying programs that interact with people (e.g., transaction processing) or interact with programs belonging to other organizations (e.g., electronic data interchange). Communication inputs and outputs are in an I/O language whose sentences are meaningful speech acts identified in the language as questions, answers, offers, acceptances, declinations, requests, permissions, and promises. The correctness of programs is partly defined in terms of proper performance of the speech acts. Answers should be truthful and responsive, and promises should be kept. Sentences of logic expressing these forms of correctness can be generated automatically from the form of the program. Elephant source programs may not need data structures, because they can refer directly to the past. Thus a program can say that an airline passenger has a reservation if he has made one and hasn't cancelled it. Elephant programs themselves can be represented as sentences of logic. Their extensional properties follow from this representation without an intervening theory of programming or anything like Hoare axioms. Elephant programs that interact non-trivially with the outside world can have both input-output specifications, relating the programs inputs and outputs, and accomplishment specifications concerning what the program accomplishes in the world. These concepts are respectively generalizations of the philosophers' illocutionary and perlocutionary speech acts. Programs that engage in commercial transactions assume obligations on behalf of their owners in exchange for obligations assumed by other entities. It may be part of the specifications of an Elephant 2000 programs that these obligations are exchanged as intended, and this too can be expressed by a logical sentence. Human speech acts involve intelligence. Elephant 2000 is on the borderline of AI, but the talk emphasizes the Elephant usages that do not require AI.

John McCarthy, Stanford University

John McCarthy received the Turing Award in 1971 for his major contributions to the field of Artificial Intelligence. He was responsible for coining the term "Artificial Intelligence" in 1955. McCarthy championed mathematical logic for Artificial Intelligence. In 1958, he proposed the Advice Taker, which inspired later work on question-answering and logic programming. He invented the Lisp programming language and published its design in Communications of the ACM in 1960. He helped to motivate the creation of Project MAC at MIT; he left MIT for Stanford University in 1962, where he helped set up the Stanford AI Laboratory. In 1961, he was the first to publicly suggest (in a speech given to celebrate MIT's centennial) that computer time-sharing technology might lead to a future in which computing power and even specific applications could be sold through the utility business model (like water or electricity). This idea of a computer or information utility was very popular in the late 1960s, but faded by the mid-1970s as it became clear that the hardware, software and telecommunications technologies of the time were simply not ready. However, since 2000, the idea has resurfaced. He is currently Professor Emeritus at Stanford University.
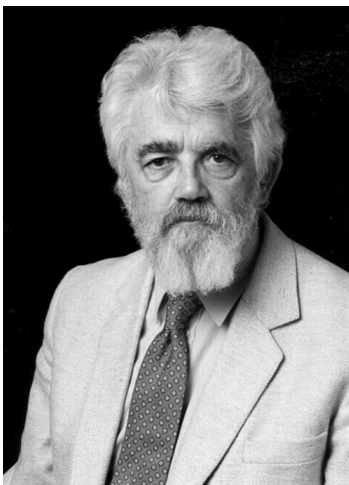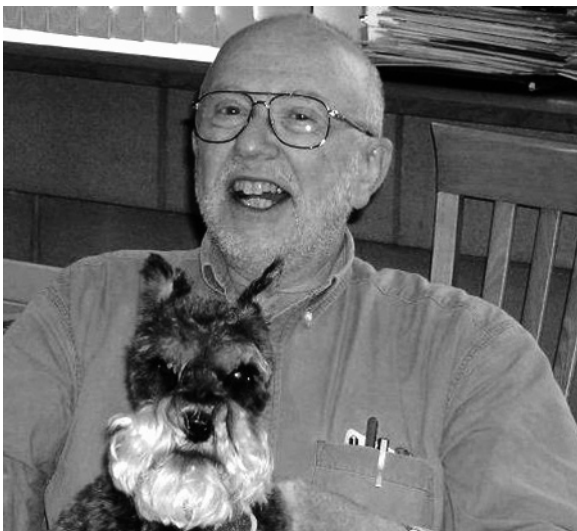
Invited Speaker: Precise Software Documentation: Making Object-Orientation Work Better

Wednesday
Oct 24, 15:30–17:00

517A

Computer Scientists have been talking about the use of of object-orientation (under a variety of rubrics) to achieve "separation of concerns" for more than 40 years. In all that time, it has been taken for granted that it was the structure of the program text itself that mattered. Whenever it was felt that additional information was needed it was assumed that this would be closely associated with the program text either as simple comments, as in-line assertions, or "woven" in with the program text using elaborate tools.

This talk takes a different position. It argues that for true separation of concerns we need an integrated set of separate documents, some of which are to be read by people who will never read the code, some that describe the structure of the code, and some that describe the behaviour of individual components. It describes a collection of mathematical ideas and notations that make it possible to produce documentation that is both precise and readable. It then describes the use of these documents in testing and inspection. Finally, it discusses the way that other programming paradigms, particularly functional programming, can be used to make these documents more useful.

David Lorge Parnas, University of Limerick

David Lorge Parnas is Professor of Software Engineering, SFI Fellow, Director of the Software Quality Research Laboratory at the University of Limerick, Professor Emeritus at McMaster University, and Adjunct Professor at Carleton University. Parnas received his B.S., M.S., and Ph.D. in Electrical Engineering—Systems and Communications Sciences—from Carnegie Mellon University, and honorary doctorates from the ETH in Zurich and the Catholic University of Louvain. He is a Fellow of the Royal Society of Canada and of the Association for Computing Machinery (ACM) and a Member of the Royal Irish Academy. He is licensed as a Professional Engineer in Ontario. Parnas won an ACM 'Best Paper' award in 1979, two 'Most Influential Paper' awards from the International Conference on Software Engineering, the 1998 ACM SIGSOFT 'Outstanding Research Award', the 'Practical Visionary Award' given in honour of the late Dr. Harlan Mills, and the 'Component and Object Technology' award presented at TOOLS99. He was the first winner of the Norbert Wiener Prize from Computing Professionals for Social Responsibility and received the FifF prize from the Forum Informatiker für Frieden und Verantwortung in Germany. He is the author of more than 240 papers and reports. Many of his papers have been repeatedly republished and are considered classics. A collection of his papers can be found in Hoffman, D.M., Weiss, D.M. (eds.), "Software Fundamentals: Collected Papers by David L. Parnas."

Keynote: Context, Perspective, and Programs	Thursday Oct 25, 8:30–10:00	517A

Context plays a large role in our perspective on the world around us—people see things differently depending on background, role, task at hand, and many other variables. How do different contexts affect developer perspectives on software? What different ways do developers want to see a program? What different ways do they want to work with a program? How does a program mean different things to different people? How does context influence perspective? How do different contexts and perspectives interact? Can these interactions be reified, controlled, and parameterized? A broad range of work has explored these questions, but many issues remain open. We lack a general understanding of the concepts and mechanisms that can support the changes in perspective we need. We lack the ability to handle context and perspective systematically, easily and reliably throughout software development. Work is needed in a number of areas, from conceptual foundations to theory, languages, tools, and methods. A truly satisfying handle on these issues may even require a material expansion of the foundations of computation—or at the very least the foundations of programming languages.

Gregor Kiczales, University of British Columbia

Gregor Kiczales is Professor of Computer Science at the University of British Columbia. His work is directed at enabling programmers to write programs that, as much as possible, look like their design. He has pursued this goal in a number of projects. He led the Xerox PARC teams that developed aspect-oriented programming and AspectJ. He worked on principles of open implementation, that build on reflection to allow principled access to a module's implementation strategy. He is a co-author, with Danny Bobrow and Jim des Rivières of "The Art of the Metaobject Protocol". He was one of the designers of the Common Lisp Object System (CLOS), and developed the standard PCL implementation of CLOS. He was the implementer of the first Boxer system, a computational medium for non-computer experts.

Invited Speaker: Meta-objects for the world around us	Thursday Oct 25, 13:30–15:00	517A

This talk is informed by the concepts of computational reflection including reflective architectures, causal connections, and the definitions of these concepts in an object-oriented setting, and by an effort to radically rethink the human-machine interactive experience. The latter effort endeavors to design interfaces that are more immersive, more intelligent, and more interactive, and by doing so to change the human-machine relationship and to create systems that are more responsive to people's needs and actions and that become true "accessories" for expanding our minds.

Pattie Maes, MIT

Pattie Maes is an associate professor in MIT's Program in Media Arts and Sciences and interim head of the Program in Media Arts and Sciences. She founded and directs the Media Lab's Ambient Intelligence research group. Previously, she founded and ran the Software Agents group. Prior to joining the Media Lab, Maes was a visiting professor and a research scientist at the MIT Artificial Intelligence Lab. She holds bachelor's and PhD degrees in computer science from the Vrije Universiteit Brussel in Belgium. Her areas of expertise are human-computer interaction, artificial life, artificial intelligence, collective intelligence, and intelligence augmentation. Maes is the editor of three books, and is an editorial board member and reviewer for numerous professional journals and conferences. She has received several awards: Newsweek magazine named her one of the "100 Americans to watch for" in the year 2000; TIME Digital selected her as a member of the Cyber-Elite, the top 50 technological pioneers of the high-tech world; the World Economic Forum honored her with the title "Global Leader for Tomorrow"; Ars Electronica awarded her the 1995 World Wide Web category prize; and in 2000 she was recognized with the "Lifetime Achievement Award" by the Massachusetts Interactive Media Council.

Events

Welcome / Poster Reception **(light refreshments)**	**Monday** **Oct 22, 17:30–19:30**	**710**

The Welcome Reception is open to all conference attendees. This is an excellent forum to share information on the day's activities. The reception is an informal and highly interactive environment that gives ooPSLA attendees the opportunity to engage with one another in discussions about relevant, ongoing work and critical issues in key areas.

The Posters program begins with a special session at the Welcome Reception. All posters will be on display and the authors will be present to meet with attendees and discuss their work. The reception gives conference attendees the chance to learn about work in many areas and about preliminary research results. Come meet your old friends and make some new ones.

Newcomer Orientation	**Monday** **Oct 22, 19:30–20:30**	**516AB**

There's so much to do and see at ooPSLA that planning a schedule can be daunting, particularly for newcomers. So whether this is your first ooPSLA or you're just feeling overwhelmed, be sure to join us Monday evening for the Newcomer Orientation right after the Welcome Reception. We'll answer your questions and give you tips on how to navigate the conference.

Collaborating with Jazz **(light refreshments)**	**Tuesday** **Oct 23, 17:15–18:30**	**516AB**

A Concurrent Demonstration at OOPSLA 2007, Montréal and CASCON 2007, Toronto
Erich Gamma, John Wiegand, and Team, IBM Rational

The Eclipse Platform was developed using agile practices, but the team never had the luxury of developing in a single location or even a single time zone. The team adapted and evolved its own practices and those adopted from the agile community in order to support their work to ship quality software on time. The team has naturally been exploring how tools can help with these practices. These explorations are now called the Jazz Project. You'll have the opportunity to see Jazz in action as the team presents the current state of Jazz technology and a joint demonstration concurrently and collaboratively from OOPSLA 2007 in Montreal and CASCON 2007 in Toronto. Join us afterward for light refreshments and interaction with the team.

Special Event	Wednesday Oct 24, 19:00–???	

The special event: let's talk about it. The ooPSLA special event typically is special—like, the San Diego Zoo, OMSI, science museums, aquariums, etc etc. This year the special event is mundane. We will glorify it. We will draw from the outside and merge with the commonplace while creating a sense of transparency and interface. We will leave from the Hyatt by bus, but walking is easy enough. The place is ordinary, but we'll fill it with us / with some this and that, some music maybe (some "music" maybe), some curiosities, some films. It will stay open late. Most of us will walk back. In the end, those oos will be shriveled even more.

Ice Cream Social + Invitation to ooPSLA 2008	Thursday Oct 25, 17:00–18:00	516AB

The Ice Cream Social serves as the kick-off event of next year's ooPSLA where you will be able to pick up your ooPSLA 2008 poster and just plain yap.

Awards

Most Influential 1997 OOPSLA Paper	Tuesday Oct 23, 13:30	517A

Every year, SIGPLAN gives the ooPSLA Most Influential Paper award to the author(s) of the paper presented at the ooPSLA held 10 years prior to the award year that is judged to have had the most influence over the past decade. The award includes a prize of $1,000 to be split among the authors of the winning paper.

Student Research Competition ACM Distinguished X Award	Thursday Oct 25, 15:30–17:00	517A

The winner of the ACM SIGPLAN Student Research Competition will be announced, as will the ooPSLA recipients of the ACM Distinguished Scientist / Engineer / Member and Fellow awards.

This year we have so much space, we decided to permit people to organize their of BoFs and just pick a room. We'll provide some bulletin-board space or you can use the oopsla wiki (http://www.oopsla.org/wiki2007/).

Collaborative Development Research using Jazz	**Wednesday** **Oct 24, 12:00–13:30**	**516AB**

Harold Ossher, IBM Research

Jazz (http://jazz.net) is a joint project between IBM Rational and IBM Research to build a scalable, extensible team collaboration platform for seamlessly integrating tasks across the software lifecycle. As such, it is a suitable platform for research both on tools and environments for team development and on team interactions and development artifacts (e.g., repository mining). This Birds of a Feather session is intended to bring together members of the Jazz team, researchers who are already doing Jazz-related research, as well as researchers who may be interested in doing so. There will be short demonstrations / presentations on three Jazz-related projects done at the University of British Columbia, the University of California, Irvine, and the University of Victoria, followed by general discussion and Q&A.

Chair: Brian Sletten, independent consultant

As one of few conferences to bring together active and enthusiastic participation from the academic, commercial, and government domains, ooPSLA represents a key opportunity to expose the innovations and advancements suggested by your work. As a key view into the status of a dynamic field, the ooPSLA demos track highlights both individual and collective advancements from one year to the next. This year's demos will compel progress in the software landscape and just might influence the industry's influencers.

CruiseControl.rb—continuous integration the Rails way	Tuesday 10:00	512E

Alexey Verkhovsky, ThoughtWorks

CruiseControl.rb is the third reincarnation of ThoughtWorks open-source continuous integration franchise. It's basic purpose in life is to alert members of a software project when one of them checks something into source control that breaks the build. CC.rb brings to the world of continuous integration the same values that Ruby On Rails brought to the world of web frameworks. Simplicity, convention over configuration and aesthetic beauty. The demonstration first goes through installation and getting started. This is the boring part, so we will be done with it in the first 5 minutes. Then we will look at doing something more interesting, such as:

- building projects in any language on any platform
- code coverage analysis
- custom notification systems with email, instant messaging, RSS feeds and lava lamps
- managing build dependencies
- configuring integration and deployment builds
- distributed parallel builds

Compile-time Typechecking for Custom Java Type Qualifiers	Tuesday 10:30	512E

Matthew M. Papi, MIT CSAIL   **Michael D. Ernst**, MIT CSAIL

Type qualifiers help to detect and prevent errors by helping programmers to organize data, and by allowing tools to perform more sophisticated analyses. Building on existing Java tools and APIs, we have created a system that enables programmers to add custom type qualifiers to the Java language. The system extends the javac Java compiler to allow programmers to write type qualifiers in their programs and to create compiler plugins that define custom type qualifiers and enforce the semantics of these qualifiers at compile time. In the demonstration, we introduce a plugin to Sun's Java compiler that uses our system to typecheck the ''nonnull'' qualifier during program compilation. Programmers can write ''nonnull'' to specify that a variable cannot take on the value null; then, by using the ''nonnull'' plugin, they can check for ''nonnull'' errors at compile time and rid their programs of null pointer exceptions. The demonstration will discuss the benefits and use of the ''nonnull'' plugin as well as how to create new type qualifiers with our system, using the ''nonnull'' plugin as an example. We have created typecheckers for other qualifiers (including ``interned'' and ''readonly''), and we will briefly demonstrate these typecheckers alongside our ''nonnull'' plugin.

Demonstrations

Green—A Scalable Class Diagramming Tool for Eclipse

Tuesday 11:00   **512E**

Carl Alphonce, University at Buffalo, SUNY
Adrienne Decker, University at Buffalo, SUNY
Brian Mcskimming, Universtiy at Buffalo SUNY

Gene Wang, Universtiy at Buffalo SUNY
Joshua Gardner, Universtiy at Buffalo SUNY
Zachary Marzec, Universtiy at Buffalo SUNY

Green is a live round tripping UML class diagram editor for Eclipse, originally designed with the intention of focusing CS1/CS2 students on modeling and design. Green's ease of use and personalizable features have allowed it to grow into a robust and scalable tool providing end users with an easy to use application satisfying their individual class diagramming needs. Green's live round tripping capability allows users to generate (Java) code from UML class diagrams and generate diagrams from (Java) code and have them both update each other as any changes are made. Green's editor features a simple point-and-click interface. Right-clicking in the editor brings up a context menu which includes refactoring, quick-fix functionality and incremental exploration. Incremental exploration allows an end user to select any type displayed in the diagram and have Green display all types which have relationships originating with the selected type. This allows the user to explore the design of a piece of code quickly class by class, based only upon the relationships contained within. All relationships in Green are plugins to Green itself. Each relationship can be added or removed with little effort in order to tailor the user's experience as they desire. For instance, if you are only concerned with Generalization (Inheritance) relations, all other relationships can be removed completely from Green and its relationship recognizer without impacting any other functionality. This demo will demonstrate the main features of Green, including forward and reverse engineering, incremental exploration, and the installation and removal of relationship semantics.

Leveraging Integrated Model-Driven Development and Software Product Line Development Technologies

Tuesday 11:30   **512E**

Charles Krueger, BigLever Software

In recent years, software product line (SPL) development has emerged as a highly effective strategic commercial practice. However, model-driven development (MDD) has remained an under-served part of the SPL development lifecycle—until now. The recent integration of leading-edge MDD and SPL technologies, co-developed by Telelogic and BigLever Software, enables development organizations to leverage the synergistic benefits of both practices. This demonstration provides insight into how integrated MDD and SPL technologies can be used to effectively manage product line diversity in MDD processes. With the simple, elegant approach featured in this demo, development organizations can achieve a new level of optimization in development productivity, portfolio scalability and time-to-market. Attendees will see pragmatic techniques for utilizing SPL consolidation, first-class model variation points, and automated production capabilities, rather than creating cloned copies of MDD models for each product or building "one-size-fits-all" models for all products. The methods described in this demo are broadly applicable and do not depend on any particular application domain or programming language.

AmbientTalk/2: Object-oriented Event-driven Programming

Tuesday 10:00   **512G**

Tom Van Cutsem
Stijn Mostinckx
Elisa Gonzalez Boix
Stijn Timbermont

Jorge Vallejos
Jessie Dedecker
Wolfgang De Meuter

The recent progress of wireless networks and mobile hardware technologies has lead to the emergence of a new generation of so-called "collaborative" applications. These applications are deployed on mobile devices equipped with wireless infrastructure which collaborate spontaneously with other devices in the environment forming mobile ad hoc networks. Distributed programming in such networks is substantially complicated by the fact that the connection between devices is often very intermittent and because devices lack any kind of centralized coordination facility. Any application designed for mobile ad hoc networks has to deal with these new hardware phenomena. In order to aid the developer in writing software for mobile ad hoc networks, new programming paradigms and languages are required which ease distributed programming by taking these phenomena into account from the ground up.

Debugging and Testing Behavioral UML Models

Tuesday 10:30　　　　512G

Dolev Dotan, IBM Haifa Research Lab
Andrei Kirshin, IBM Haifa Research Lab

As software complexity increases, the process of software development is shifting from being code-centric to model-centric. For this purpose, UML augments the object-oriented paradigm with powerful and flexible behavioral modeling capabilities. It allows the developer to describe the system's behavior in a higher level of abstraction by using state machines, activities, and interactions. To facilitate such model driven development, we present a plug-in for IBM Rational's modeling tools, which enables the execution, debugging and testing of UML models. The presentation will show how to use our tools to discover defects early in the development cycle, thus preventing costly rework at later stages. We will highlight the innovative features of our tools, such as model-level debug control, interactive dynamic debugging, and the extensibility that allows developing support for UML profiles.

Supporting Systems QoS Design and Evolution through Model Transformations

Tuesday 11:00　　　　512G

Amogh Kavimandan, Vanderbilt University　　　　**Aniruddha Gokhale**, Vanderbilt University

We describe Quality of service pICKER (QUICKER), a model-driven QoS mapping toolchain for supporting the design and evolution of application QoS. QUICKER automates the mapping of QoS requirements onto platform-specific QoS configuration options by (1) choosing appropriate subset of QoS options for given QoS policies and (2) assigning values to each of these selected QoS options. QUICKER also provides support for validating the generated QoS configurations and resolving any dependencies between them.

Green Applications: Software Applications that Optimize Energy Usage

Tuesday 11:30　　　　512G

Hoi Chan, IBM T.J.Watson Research Center　　　　**Jeff Kephart**, IBM T.J.Watson Research Center

Energy consumption is a major cost of operating IT equipment in organizations and data centers. Hardware and software manufacturers are beginning to include power saving options in a range of products from processors to the operating systems (OS). Previous work on power saving has focused mainly on OS control of the operating modes of laptop computers and mobile devices to maximize battery life. To take full advantage of these newly available energy saving features, we have developed a power management agent with a set of power control and monitoring interfaces which allows power saving features to be implemented on individual applications. We call these power saving capable applications Green Applications. In this demonstration, we will discuss and show a web based Green Application which implements these power saving features. This application runs on an IBM HS20 Blade Server with Linux OS and equipped with a processor (Intel Xeon 3 GHz) capable of power management. When the application runs, it continuously adjusts the power level (range from standby to maximum) of the server processor in accordance with the state and performance requirements of the application (expressed as policies) via the power management agent. The power management agent can manage multiple Green Applications and can set the power level of the processor according to the aggregate power requirement of the applications. Using an advanced monitoring tool, the IBM Tivoli Monitoring System, we will show graphically the dynamic interactions of the state and performance of the application, including processor temperature, CPU utilization, power cap, and power consumed in a single integrated view, and presents the amount of energy used in comparison with running the same application with no power saving features included.

Demonstrations

BigLever Software Gears and 3-Tiered SPL Methodology

Tuesday 15:00 512E

Charles Krueger, BigLever Software

Software product line (SPL) technologies and tools allow development organizations to engineer their product line portfolio as though it is a single system. This demonstration provides insight into how innovative SPL approaches shift the development focus from a multitude of products to a single software production line capable of automatically producing all of the products in a product line portfolio. More specifically, this demo will explore the SPL technology that enables companies to employ the pragmatic 3-Tiered SPL Methodology—a new generation of product line development methods that are yielding order-of-magnitude improvements in time-to-market, engineering cost, product quality, and portfolio scalability. The demo will spotlight best methods from recent industry case studies—including Software Product Line Hall of Fame inductees LSI Logic and Salion—and provide pragmatic insights into how the 3-Tiered Methodology allows companies of all types and sizes to realize a new level of benefits, in terms of both technical and business impact. The methods described are broadly applicable and do not depend on any particular application domain or programming language.

Theory-infected: or How I Learned to Stop Worrying and Love Universal Quantification

Tuesday 15:30 512E

David Saff, MIT CSAIL

Frequently executing tests on hand-picked example scenarios validates an initial software implementation and protects against future regressions, but cannot verify behavior on other examples the developer hasn't considered, even if she can easily verbally state the general principle, or "theory" involved (i.e.: "If text is not null, parse(text) must not throw a NullPointerException"). How can a developer get feedback when a theory fails? Formal verification tools require a rarer set of skills and tools from testing, and separate the theories from the equally useful tests. Tillman and Schulte propose a better approach, Parameterized Unit Tests. These express theories in code similar to unit tests, specifying behavior over all possible values of the input parameters. Theories and tests should be written and executed at the same time, with the developer choosing which is best to express her intentions. Theories are used in two different feedback loops. Theory evaluation uses a developer-supplied set of inputs for quick feedback. Theory exploration can use random testing, symbolic execution, and heuristics to look for new inputs on which the theory fails. I will demonstrate the new native support for theory evaluation in JUnit, using them to test-drive a simple class. I will then go on to demonstrate Theory Explorer, new a free tool that works on top of Agitar's JUnit Factory service, which finds new values that the developer has not considered. Theories can more quickly create a robust test suite, and they push for an algebraic design based on immutable types.

Using Sybase Workspace to Build Service Oriented Architecture (SOA) Application Quickly

Tuesday 16:00 512E

Mads Torgersen, Microsoft Corporation   **Meyer Tanuan**, Sybase Canada

In the past ten years, there is significant growth in the number of new technologies for Java developers. More recently, Web Services and Service Oriented Architectures (SOA) are becoming popular. Projects are now more complex and are becoming more challenging to complete on time. Service Oriented Architecture (SOA) is a set of frameworks, patterns, design & development principles enabling the implementation of loosely coupled software applications based on services as the architectural elements. A typical SOA application consists of services that encapsulate existing information sources from databases, messaging, Java classes, Service Oriented Architecture Protocol (SOAP) and Enterprise Java Beans (EJB). This demonstration will focus on how Sybase Workspace helps build SOA applications quickly. Sybase WorkSpace is a service-oriented unified design and development environment that includes the power of enterprise modeling with comprehensive tooling capabilities. This demonstration shows how enterprise modeling, database development, Web application development, services-oriented development and orchestration, and mobile development all come together to build SOA applications quickly. Sybase WorkSpace leverages the open-source Eclipse tooling framework and combines it with visual drag and drop development, debugging tools, sample code and cheat sheets, to drastically reduce manual coding and to cut the typical development tool learning curve.

Extracting a Domain Specific Language From An Example

Tuesday 16:30   **512E**

Ville Oikarinen, Sysart Oy

This demonstration shows a lightweight and fast method for creating a tested and working domain specific language. The method is demonstrated using the **ngrease** metalanguage, but it can also be used with any traditional template language. It is recommended to read the introduction to the **ngrease** language (http://ngrease.sourceforge.net/introduction.html). The creation of a new language is started by writing a representative example of the final product with a test that tests the transformation from a stub source to the result. The test is made to pass by writing a "constant" transformer that unconditionally outputs the result. The method is closely related to the Parameterize Method refactoring (http://www.refactoring.com/catalog/parameterizeMethod.html): At each step some part of the transformer template is converted from a constant subtree to a reference to data read from the source tree, thus driving additions to the new language. Optionally, each refactoring step can be driven by a new test that demonstrates the lack of parameterization of some part of the final product

Javari: Enforcing and Inferring Immutability in Java

Tuesday 13:00   **512G**

Michael D. Ernst, MIT CSAIL   **Jaime Quinonez**, MIT CSAIL
Telmo Correa, MIT CSAIL

Accidental mutation causes difficult-to-detect errors, since the mutation itself is not different from other mutations that happen through the program, and a mutation error is not immediately detected. We present a type-system based approach to the accidental mutation problem. This demonstration will show how the tools can be used to detect and prevent accidental mutability errors. It will demonstrate the Javarifier, inferring annotations to the code, and the Javari type checker, enforcing them. Javari is an extension of the Java language that permits the specification and compile-time verification of immutability constraints. Programmers can state the mutability and assignability of references using a small set of type annotations. The extension is compact enough to be implemented through Java annotations, keeping the code backwards compatible. Two related tools will be presented: the Javarifier, an implementation of a type inference algorithm for Javari, and the Javari type checker, a plugin for the javac compiler that enforces the annotations related to immutability. The Javarifier can be used to automatically annotate existing code, enabling Javari to be more easily adopted and preventing accidental mutability errors in future code that uses the annotated code. The Javari checker is a javac plugin that enforces the Javari annotations. The Javari language toolset can be downloaded at http:/pag.csail.mit.edu/javari

Querying in C# - How Language Integrated Query (LINQ) works

Tuesday 15:00   **512G**

Mads Torgersen, Microsoft Corporation

Language Integrated Query (LINQ) is part of the upcoming version 3.5 of the .NET Framework. As a combination of APIs and enhancements to the .NET programming languages, LINQ provides a uniform approach to querying of data across any data source. LINQ pulls the querying experience into the programming language space, providing full static typing and tool support. LINQ is built to be pluggable, allowing data source providers to insert their own query engines. Using the new C# 3.0 this demonstration peels apart the layers of LINQ to show how a smooth user experience on the surface emerges from new language features, naming conventions and metaprogramming facilities.

Demonstrations

TuningFork: Visualization, Analysis, and Debugging of Complex Real-time Systems

Tuesday 15:30 **512G**

David Bacon, IBM Research
Perry Cheng, IBM Research

David Grove, IBM Research

Debugging the timing behavior of real-time systems is notoriously difficult, and with a new generation of complex real-time systems whose size is measured in tens of millions of lines of code, the difficulty is increasing enormously. We have developed TuningFork, a tool especially designed for visualization, analysis, and debugging of large-scale real-time systems. The system is capable of recording high-frequency events at sub-microsecond resolution with almost no perturbation to the application. The visualization tool is capable of viewing system activity online in real-time, and users can simultaneously explore the data interactively. Data can be gathered from multiple levels (OS, JVM, application) and synthesized into visualizations that enable understanding the interactions between the layers. Interactive exploration of hypothesis is naturally supported by direct manipulation and drag and drop operations to quickly build up complex visualizations.

Finding Bugs in Eclipse

Tuesday 16:00 **512G**

William Pugh, Uniiversity of Maryland

This will be a live demonstration of FindBugs, a static analysis bug finding tool, on the current development version of Eclipse 3.4. FindBugs reports issues such as null pointer dereferences, comparing incompatible types with equals, invalid method calls, infinite recursive loops, bad integer operations, and more. FindBugs reports more than 550 such issues in Eclipse 3.3. During this demonstration, we'll give a quick overview of the FindBugs GUI and walk through 10-20 bug warnings, categorize each warning as to whether or not fixing the issue is important, and enter comments about the bug. We'll be able to browse warnings by date of introduction, so we can see if the issues introduced in the past month are more or less serious than the issues that have been in the code base since Eclipse 3.3, 3.2 or earlier. Vocal audience participation is encouraged, and participants with laptops can follow along and enter their own categorization and comments either during the demonstration or afterwards. Audience members with commit privileges to the Eclipse project will get free FindBugs T-shirts. We'll also briefly demonstrate how to set up FindBugs as part of a production development environment.

Democratizing The Cloud

Tuesday 16:30 **512G**

Erik Meijer, Microsoft Corporation

Programming distributed data-intensive web and mobile applications is gratuitously hard. As the world is moving more and more towards the software as services model we have to come up with practical solutions to build distributed systems that are approachable for normal programmers. Just like Visual Basic democratized programming Windows by removing much of the boilerplate such as message pumps and window handles that contributed more to the problem than to the solution, we propose a toolkit of language extensions, APIs, and tools that do the same for web programming. As a result, ordinary programmers can concentrate on the essential aspects of building data intensive distributed applications such as partitioning and flowing code and data across tiers, deployment, security, etc. without getting bogged down in low level details.

Visual Basic 9   Thursday 10:00   512G

Erik Meijer, Microsoft Corporation

When people hear "Visual Basic" they remember QuickBasic from the DOS days. As a result, their kneejerk reaction is often that Visual Basic is not really a serious language, and they do not really pay it the attention it actually deserves. In reality Visual Basic is a full-fledged modern object-oriented language with many unique features such as static typing where possible but dynamic typing where necessary, declarative event handling, deep XML integration with optional layered XSD types, highly expressive query comprehension syntax, type inference, etc. etc. This makes Visual Basic actually more interesting to both researchers and practitioners alike than the popular static languages such as Java or C# and dynamic languages such as Ruby or JavaScript.

Security Testing with Selenium   Thursday 10:30   512G

Vidar Kongsli, Bekk Consulting AS

Selenium is a tool for creating and running automated web tests that fits well into agile projects, where it can be used for creating acceptance tests corresponding to the application's user stories. This demonstration will show how Selenium additionally can be leveraged to create security tests for web applications. We model security threats as misuse stories, similar to user stories, except that we focus on illegal or non-normative use of the application. We then go on to create security tests in Selenium which will manifest the misuse story by exploiting vulnerabilities in the application. Contrary to a normal acceptance test, the security test will pass when the misuse story cannot be carried out. This approach can be seen as a contribution to strengthening the security focus in agile projects by trying to apply familiar agile concepts, methods, and tools to the security aspects of the application. We have found that several of the most common security vulnerabilities in web applications can be addressed with this approach, such as cross site scripting (XSS), broken authentication and session management, information leakage, and improper error handling. This demonstration will show examples of such vulnerabilities and corresponding tests, in addition to discuss the cases where there are shortcomings.

Ready for Distribution? Turning Modular into Distributed Applications with the R-OSGi Deployment Tool   Thursday 11:00   512G

Jan Rellermeyer, ETH Zurich   **Timothy Roscoe**, ETH Zurich
Gustavo Alonso, ETH Zurich

In this demonstration we show drag-and-drop distribution of centralized, modular Java applications. Our system is based on OSGi, an industry standard for building Java applications out of modular units loosely connected through services. Since OSGi is a centralized system, we have elaborated a solution to seamlessly distribute OSGi applications along the boundaries of services and thereby turning arbitrary OSGi applications into distributed applications. In this demonstration, we present an Eclipse based tool that takes the source code of an OSGi application as input, produces a graph of its modules and module dependencies, and allows the user to deploy the application across a distributed system by dragging-and-dropping its constituent modules on different machines. By defining constraints on the distribution, the tool can also support advanced features like load-balancing or redundancy of modules.

Demonstrations

PTIDEJ and DECOR: Identification of Design Patterns and Design Defects

Thursday 10:30   **512E**

Naouel Moha, Ptidej Team—GEODES Group, DIRO, University of Montréal
Yann-gaël Guéhéneuc, Ptidej Team—GEODES Group, DIRO, University of Montréal

The PTIDEJ project started in 2001 to study code generation from and identification of patterns. Since then, it has evolved into a complete reverse-engineering tool suite that includes several identification algorithms. It is a flexible tool suite that attempts to ease as much as possible the development of new identification and analysis algorithms. Recently, the module DECOR has been added to PTIDEJ and allows the detection of design defects, which are recurring design problems. In this demonstration, we particularly focus on the creation and use of identification algorithms for design patterns and defects.

DEMOCLES: A Tool for Executable Modeling of Platform-Independent Systems

Thursday 11:00   **512E**

Christian Glodt, University of Luxembourg   **Elke Pulvermueller**, University of Luxembourg
Pierre Kelsen, University of Luxembourg

The main goal of model-driven architecture is the generation of the full implementation of a system based on a precise description of a platform-independent model and a platform model. Such a description must accurately specify the static structure as well as the dynamic behavior of the system. We present a tool—called DEMOCLES—that realizes a hybrid approach to platform-independent modeling. It describes the static structure using a modified UML class diagram that separates query operations from modifier operations. The former are defined in the class diagram via OCL constraints, while the latter are defined using a MOF-based metamodel that contains modifier operations and properties as first-class entities and augments them with associations and OCL expressions. The tool is an Eclipse-plugin that offers overlay views of the structure and behavior with visual editing capabilities and permits execution of a platform-independent system.

DigitalAssets Discoverer: Automatic Identification of Reusable Software Components

Thursday 11:30   **512E**

Eduardo M. Gonçalves, DigitalAssets Innovation Labs   **Kleber R. Bacili**, DigitalAssets
Marcilio Oliveira, Unicamp University

DigitalAssets Discoverer is a tool that implements a group of indicators for automatic identification of software components that can be reused in the development of new applications and Web Services. This tool brings into light the J2EE applications portfolio developed in-house, increasing productivity and anticipating the ROI for companies. The process of components analysis and harvesting uses an interactive user graphical interface that enables the tuning of selected indicators, visualization of the results and publishing of the identified components publication into a reusable software development assets repository.

Model-driven Development with Predictable Quality

Thursday 15:00   **512E**

James Ivers, Carnegie Mellon Software Engineering Institute
Gabriel A. Moreno, Carnegie Mellon Software Engineering Institute

Many software systems have stringent quality attribute requirements. For example, industrial robots must perform tasks with strict deadlines, medical devices must comply with safety requirements, and most software must minimize security vulnerabilities. Although analysis theories and techniques addressing these requirements have existed for many years, they are not widely used because of the resources and expertise required to create, maintain, and evaluate analysis models. Consequently, developers usually rely on

testing to verify the satisfaction of these requirements, incurring expensive overruns when they are not met. The PACC Starter Kit (PSK) is an eclipse-based development environment that combines a model-driven development approach with reasoning frameworks (RFs) that package the expertise needed to apply quality attribute analyses. A reasoning framework is used to achieve some level of confidence (statistical or proof-based) in predictions of runtime behavior based on specifications of component behavior. RFs ensure that designs satisfy analytic assumptions and then automatically generate analysis models appropriate for reasoning about specific runtime behaviors. The PSK demonstration focuses on two RFs. The performance RF predicts latency in soft and hard real-time systems. The model checking RF analyzes runtime behavior for safety properties, such as detecting buffer overflows, and can generate proofs that binary code satisfies such properties. The demonstration also touches on the included code generator and runtime environment. The PSK generates complete implementations for assemblies of components; this automation ensures consistency between executable code and analysis models. The runtime environment consists of the Pin component technology and a simple real-time operating system extension.

Mining Implementation Recipes of Framework-Provided Concepts in Dynamic Framework API Interaction Traces	Thursday 15:30	512E

Abbas Heydarnoori, University of Waterloo   **Krzysztof Czarnecki**, University of Waterloo

Application developers often apply the Monkey See/Monkey Do rule for framework-based application development, i.e., they use existing applications as a guide to understand how to implement a desired framework-provided concept (e.g., a context menu in an Eclipse view). However, the code that implements the concept of interest might be scattered across and tangled with code implementing other concepts. To address this issue, we introduce a novel framework comprehension technique called FUDA (Framework API Understanding through Dynamic Analysis). The main idea of this technique is to extract the implementation recipes of a given framework-provided concept from dynamic traces with the help of a dynamic slicing approach integrated with clustering and data mining techniques. In this demonstration, we present the prototype implementation of FUDA as two Eclipse plug-ins, and use them to generate the implementation recipes for a number of concepts in Eclipse views and GEF editors by using only a few example applications.

Improving Quality Together	Tuesday 13:00 Thursday 16:00	512E

David Jones, Intelliware Development Inc.   **Gordon Cameron**, Intelliware Development Inc.
Brett Dubroy, Intelliware Development Inc.

One recent change in software development is developers starting to take responsibility for the quality of their work by writing and executing automated tests. As with any new activity, there is a wide range of ways to perform this task. DevCreek collects, aggregates, and displays testing activity for individuals, teams, and the developer community as a whole. The goal is to provide individuals with global norms with which they can compare their testing. We currently have twenty person-years worth of data, with the goal of collecting thousands more from a variety of languages, tools, and domains. This demonstration will show the data collection, real-time feedback, and aggregate reporting facilities within DevCreek. We will exercise our Eclipse plug-in during a short, live development session. DevCreek's real-time data collection and visual display will provide feedback on both areas of activity and development idiom. This lets developers react when straying from their desired practices, and supports a self improvement feedback loop. The team becomes more aware of how it is functioning as a whole, with the increased transparency allowing other stakeholders to become more involved. Activity collected from the team is aggregated on the server and can be accessed from a web browser by all members of the team. The stream of test run executions can support generation of rich reports, such as a compact visualization of testing activity that can be used to recognize alternative testing patterns.

Demonstrations

Improve Software Quality with SemmleCode—an Eclipse Plugin for Semantic Code Search

Thursday 15:00   **512G**

Mathieu Verbaere, Semmle Ltd.   **Oege De Moor** Semmle Ltd.
Elnar Hajiyev Semmle Ltd.

Navigate code, find bugs, compute metrics, check style rules, and enforce coding conventions in Eclipse with SemmleCode. SemmleCode is a new free Eclipse plugin that allows you to phrase these tasks as queries over the codebase—it thus takes the search facilities in Eclipse to a whole new level. A large library of queries for common operations is provided, including metrics and J2EE style rules. Query results can be displayed as a tree view, a table view, in the problem view, as charts or graphs, all with links to the source code.

Lagrein: Recovering and Visualizing the Software Development Process

Thursday 15:30   **512G**

Andrejs Jermakovics, Free University of Bolzano-Bozen   **Marco Scotto**, Free University of Bolzano-Bozen
Giancarlo Succi, Free University of Bolzano-Bozen

Estimating the real effort spent to implement the requirements of a software system, without superimposing any overhead on the development team, represents a paramount opportunity to keep a software project under control. Lagrein is a software system that tries to address this problem by supporting managers and developers in exploring how a software system has been developed. It supports the visualization of multiple metrics (polymetric views), it links individual requirements to the portions of the source code expected to implement them, it couples the source code with the effort spent in producing it. With a certain level of approximation, Lagrein makes it possible to estimate the effort required to implement each single requirement.

The JastAdd Extensible Java Compiler

Thursday 16:00   **512G**

Torbjörn Ekman, Oxford University

The JastAdd Extensible Java Compiler, JastAddJ, is a high quality Java compiler that is easy to extend with new analyses as well as new language constructs. It is built using the metacompiler tool JastAdd which provides advanced support for constructing modular and extensible compilers. In this demonstration we introduce JastAddJ by example through a pair-programming session. The running example is to extend a Java 1.4 compiler with the enhanced for-statement as introduced in Java 5. It is a simple language construct with well-known syntax and semantics but it still highlights many interesting parts of the system. Name binding is affected since it introduces a new declaration, and additional type checking is required to ensure that the expression that the for statement operates on is actually iterable. Flow sensitive analyses such as ensuring definite assignment and computing unreachable statements are also affected. Too put things into perspective we describe how the same techniques scale up to more complex language extensions. They have been used successfully to implement all language features of Java 5 as modular extensions to Java 1.4. Another example is a pluggable type system for non-null type checking and type inferencing. The system as well as the extensions are open-source and available at http://jastadd.cs.lth.se This demonstration targets an audience having taken an introductory course in compiler construction, i.e., basic knowledge of parsing, abstract syntax trees, and symbol tables.

Automatic Support for Model-Driven Specialization of Object-Oriented Frameworks using ALFAMA

Thursday 16:30   **512G**

André L. Santos, University of Lisbon

Object-oriented frameworks are recognized as powerful assets for software reuse. However, the full potential of framework usage is somehow constrained due to difficulties concerning learning and productivity. Specially in the case of mature frameworks, their usage can be significantly facilitated by adopting a Domain-Specific Modeling (DSM) infrastructure, comprising a modeling language and a code generator for producing framework specializations. Current approaches for developing this kind of model-driven support always assume that a language and a code generator have to be manually developed to fit the framework implementation. We argue that these approaches have certain limitations concerning consistency and evolution issues. This demonstration is about a tool-supported approach that we refer to as ALFAMA. Our approach dissents from the state of the practice, in a sense that a DSM infrastructure is automatically derived from the framework implementation, at the cost of having an additional specialization layer based on aspect-oriented programming. We demonstrate the tool usage using the Eclipse Rich Client Platform (RCP) framework.

Sunday, Tuesday — DesignFest®

Chairs: Don Roberts, University of Evansville
John Brant, The Refactory

DesignFest® is new this year! We are going to celebrate and explore the design process. Where is the line between building and designing? Where is the line between art and engineering? DesignFest® will explore this question by having teams of attendees attack the same problem in different ways including: traditional methodologies, test-driven design, prototyping, and scrap-heap programming. At the end of the session, the teams will participate in a round-table discussion of the various artifacts produced and the relative merits of the various design approaches.

DesignFest®	**Sunday 8:30–17:00**	**516B**
DesignFest®	**Tuesday 8:30–17:00**	**516E**

Monday — Doctoral Symposium

Chair: Elisa Baniassad, The Chinese University of Hong Kong

The goal of the Doctoral Symposium is to provide useful guidance for the completion of the dissertation research and initiation of a research career. The Symposium provides an interactive forum for doctoral students in one of two phases in their doctoral progress, apprentices and proposers:

Apprentices are students who are just beginning their research, are not ready to actually make a research proposal, but are interested in learning about structuring research and getting some research ideas

Proposers are students who have progressed far enough in their research to have an Idea Paper and a structured proposal, but will not be defending their dissertation in the next 18 months.

Doctoral Symposium	**Monday 8:30–17:00**	**512D**

Student Volunteers

Chair: Michael Richmond, IBM Research

The Student Volunteer program is an opportunity for students from around the world to associate with the top people in programming languages, object-oriented technology, research, and software development. Students volunteer a few hours of their time performing tasks that help the conference run smoothly. These tasks include assisting with registration, providing information about the conference to attendees, and monitoring tutorials.

Chairs: **Pascal Costanza**, Vrije Universiteit
        Robert Hirschfeld, University of Potsdam

The Dynamic Languages Symposium (DLS) at OOPSLA 2007 in Montréal, Canada, is a forum for discussion of dynamic languages, their implementation and application. While mature dynamic languages including Smalltalk, Lisp, Scheme, Self, and Prolog continue to grow and inspire new converts, a new generation of dynamic scripting languages such as Python, Ruby, PHP, Tcl, and JavaScript are successful in a wide range of applications. DLS provides a place for researchers and practitioners to come together and share their knowledge, experience, and ideas for future research and development.

Program	**Monday 8:50–17:15**	**516DE**

8:50–9:00	Opening Remarks
9:00–10:00	**Invited Talk 1** Tradeoffs in Retrofitting Security: An Experience Report *Mark S. Miller*
10:00–10:30	Break
10:30–12:00	**Research Papers 1 (Multi-paradigm Programming)** Report on the Probabilistic Language Scheme *Alexey Radul* OMeta: an Object-Oriented Language for Pattern Matching *Alessandro Warth and Ian Piumarta* Arrays of Objects *Morten Kromberg*
12:00–13:30	Break
13:30–15:00	**Research Papers 2 (Integrating Static Features into Dynamic Languages)** Relationally-Parametric Polymorphic Contracts *Arjun Guha, Jacob Matthews, Robert Bruce Findler, and Shriram Krishnamurthi* Dynamic Ownership in a Dynamic Language *Donald Gordon and James Noble* RPython: Reconciling Dynamically and Statically Typed OO Languages *Davide Ancona, Massimo Ancona, Antonio Cuni, and Nicholas Matsakis*
15:00–15:30	Break
15:30–17:00	**Research Papers 3 (Software Adaptation)** An Adaptive Package Management System for Scheme *Manuel Serrano and Erick Gallesio* Highly Dynamic Behaviour Adaptability through Prototypes with Subjective Multimethods *Sebastián González, Kim Mens* Mirages: Behavioral Intercession in a Mirror-based Architecture *Stijn Mostinckx, Tom Van Cutsem, Stijn Timbermont, and Eric Tanter*
16:15–17:15	**Invited Talk 2** Bringing Dynamic Languages to .NET with the DLR *Jim Hugunin*

Tradeoffs in Retrofitting Security: An Experience Report

In 1973, John Reynold's and James Morris' Gedanken Language retrofit object-capability security into an Algol-like language. Today, there are active projects retrofitting Java, Javascript, Python, Mozart/Oz, OCaml, Perl, and Pict. These represent a variety of approaches, with different tradeoffs regarding legacy compatibility, safety, and expressivity. In this talk I propose a taxonomy of these approaches, and discuss some of the lessons learned to date.

Mark S. Miller, Google Inc.

Mark S. Miller is a research scientist at Google, open source coordinator for the E secure distributed programming language, co-creator of the agoric paradigm of market-based computing, and an architect of the Xanadu hypertext publishing system.

Bringing Dynamic Languages to .NET with the DLR

From the beginning, Microsoft's .NET framework was designed to support a broad range of different programming languages on a Common Language Runtime (CLR). The CLR provides shared services to these languages ranging from a world-class GC and JIT to a sandboxed security model to tools integration for debugging and profiling. Sharing these features has two huge benefits for languages on the CLR. First, it's easier to implement a language because lots of difficult engineering work is already done for you. Second, and more importantly, these languages can seamlessly work together and share libraries and frameworks so that each language can build on the work of the others. The CLR has good support for dynamic languages today. IronPython-1.0 demonstrates this. The new Dynamic Language Runtime (DLR) adds a small set of key features to the CLR to make it dramatically better. It adds to the platform a set of services designed explicitly for the needs of dynamic languages. These include a shared dynamic type system, standard hosting model and support to make it easy to generate fast dynamic code. With these additional features it becomes dramatically easier to build high-quality dynamic language implementations on .NET. More importantly, these features enable all of the dynamic languages which use the DLR to freely share code with other dynamic languages as well as with the existing powerful static languages on the platform such as VB.NET and C#.

Jim Hugunin, Microsoft Corp.

Jim Hugunin is an architect on the Common Language Runtime (CLR) team at Microsoft where he drives work to further improve the support for dynamic languages within the .NET platform - starting with the initial successes of IronPython. Prior to joining Microsoft, Jim worked at Xerox PARC as one of the principal designers of the AspectJ language and tools. Jim is also the creator of Jython, one of the first and still one of the most popular scripting languages for the Java platform.

Chairs: **Joseph Bergin**, Pace University
           Steve Metsker, Dominion Digital

The Educators' Symposium—the 16th at ooPSLA—is the premier forum for educators in academia and industry with an interest in object-oriented and other related technologies. This one-day symposium is a venue for consultants and academics to share experiences and to explore new ideas that can help us understand and teach better. It achieves these goals through featured talks, paper presentations, interactive sessions, demonstrations, posters, and group discussions.

Teaching Strategies for Reinforcing Structural Recursion with Lists	Monday 9:00	516A–C

Michael H Goldwasser, Saint Louis University   **David Letscher**, Saint Louis University

Recursion is an important concept in computer science and one which possesses beauty and simplicity, yet many educators describe challenges in teaching the topic. Kim Bruce champions the early use of structural recursion in an object-oriented introductory programming course as a more intuitive concept than traditional (functional) recursion. He uses many graphical examples for motivation (e.g., nested boxes, a ringed bullseye, fractals), providing concreteness to the recursive concept. Internally, most of those examples are disguised forms of a basic recursive list pattern. Recursive lists are important in and of themselves and a mainstay within the functional programming paradigm. However further challenges exist in providing a tangible presentation for pure lists when disassociated from a graphical structure. We describe an active-learning exercise in which students play the roles of distinct objects that together comprise the structure of a single, recursive list. This activity establishes intuition that we later use when developing a complete implementation of a recursive list class. Our approach demonstrates a rich set of recursive patterns involving several distinct forms of a base case and varied use of parameters and return values.

Keynote: Kathy Sierra	Monday 11:00	516A–C

Introducing Computer Science with Project Hoshimi	Monday 16:00	516A–C

Ramiro A, Berrelleza, Tecnologico de Monterrey   **Maria Elena Echeagaray Chávez**, Tecnologico de Monterrey
Javier Sánchez González, Tecnologico de Monterrey

Introductory programming course have two very specific difficulties for novice students. First is the lack use of real world examples in the sessions. It is very difficult to find areas of application where all the students are familiar enough and that offers challenging and engaging examples. Second is the lack of palpable examples of the job done. Introductory courses in other fields generate products that the students can show to others, and feel proud about it. In CS1, for example, explaining cycles by printing a series of numbers on the screen doesn't yield the same sense of ac-complishment as drawing a basic perspective in an archi-tecture class. We propose using Project Hoshimi [1], a Microsoft Platform, as a base for introducing computer programming to CS1 students. Through the paper we dis-cuss the main advantages and disadvantages in our experi-ence of using Project Hoshimi, comparing its use against other more traditional approaches, as well as against other graphic programming methods such as Alice or videogame based learning.

Continuing Professional Development by Practitioner Integrated Learning	Monday 16:30	516A–C

Anna Borjesson, Ericsson   **Miroslaw Staron**, IT University of Goteborg
Lars Pareto, IT University of Goteborg   **Ulrika Lundh Snis**, University West

To prevent skilled professionals from being phased out or forced into professions for which they are not talented, organized forms of lifelong learning are needed. Continuing professional devel-opment is an approach supporting lifelong learning. This approach is however criticized for being expensive and not providing the necessary knowledge. In response to this, we have executed a study in order to understand how universities can effectively support continuous professional development. By involving industry professionals as participants in university courses using problem based learning, we have designed what we call *Practitioner Integrated Learning* (PIL). This learning approach has shown positive effects in terms of level of learning, realism, knowledge diffusion, study load and costs. We present a 15-months action research project integrating industry managers and university students in a continuing professional development effort. Based on this study, we argue that PIL is a learning approach that effectively supports continuing professional development.

Chair: Guy L. Steele Jr., Sun Microsystems Laboratories

An ooPSLA essay is a rigorously peer-reviewed reflection upon technology, its relation to human endeavors, or its philosophical, sociological, psychological, historical, or anthropological underpinnings. An essay can be an exploration of technology, its impacts, or the circumstances of its creation; it can present a personal view of what is, explore a terrain, or lead the reader in an act of discovery; it can be a philosophical digression or a deep analysis. At its best, an essay is a clear and compelling piece of writing that enacts or reveals the process of understanding or exploring a topic important to the ooPSLA community. It shows a keen mind coming to grips with a tough or intriguing problem and leaves the reader with a feeling that the journey was worthwhile.

Confessions of a Used Programming Language Salesman	Tuesday 10:30–12:00	516A

Session Chair: William Cook

Programmers in the real world wrestle every day to overcome the impedance mismatch between relational data, objects, and XML. For the past ten years we have been working on solving this problem by applying principles from functional programming, in particular monads and comprehensions. By viewing data as monads and formulating queries as comprehensions, it becomes possible to unify the three data models and their corresponding programming languages instead of considering each as a separate special case. To bring these theoretical ideas within the reach of mainstream programmers, we have worked tirelessly on transferring functional programming technology from pure Haskell, via Comega to the upcoming versions of C# 3.0 and Visual Basic 9 and the LINQ framework. Functional programming has finally reached the masses, except that it is called Visual Basic instead of Lisp, ML, or Haskell!

Erik Meijer , Microsoft Corporation

The Transactional Memory / Garbage Collection Analogy	Wedneday 10:30–12:00	516A

Session Chair: Guy L. Steele Jr.

This essay presents remarkable similarities between transactional memory and garbage collection. The connections are fascinating in their own right, and they let us better understand one technology by thinking about the corresponding issues for the other.

Dan Grossman , University of Washington

Invited Talks

Chair: Robert Biddle, Carleton University

This year in addition to the keynote and invited speakers, we have 3 invited talks. We've done this because there are good speakers with ideas and views worth hearing who are not superstars on the keynote circuit.

Actor-Network Theory: Nothing to Do with TCP/IP or Distributed Objects	**Tuesday 15:30–17:00**	**517C**

Brian Marick , Exampler Consulting

Intimate Information for Everyone's Everyday Tasks: What hyperfiction and new media theory teach us about programming (and teenage diaries)	**Thursday 10:30–12:00**	**517C**

Mark Bernstein , Eastgate Systems

Big Balls of Muds: An introduction to Post-Modular Programming	**Thursday 15:30–17:00**	**517C**

Brian Foote , Industrial Logic, Inc.

Lightning Talks   Wednesday/Thursday

	Wednesday 13:30–17:00 **Thursday 13:30–17:00**	**510D**

Chairs: Cédric Beust, Google, Inc
         David Leibs, AMD

A lightning talk is a five-minute presentation on any topic of interest to the OOPSLA community. Conference attendees sign up in advance and at the conference to reserve their five minutes at the mike. Software developers working on exciting projects…researchers with big ideas not yet ready for a paper…any conference attendee who wants to shift the community in a new and promising direction—you'll hear all of these in lightning talks, and more!

Chair: Cristina Videira Lopes, University of California, Irvine

Objects have grown up, software is everywhere, and we are now facing a consequence of this success: the perception that we know what programming is all about and that building software systems is, therefore, just a simple matter of programming…with better or worse languages, tools, and processes. But we know better. Programming technology may have matured, programming languages, tools, and processes may have proliferated, but fundamental issues pertaining to computer **P**rogramming, **S**ystems, **L**anguages, and **A**pplications are still as untamed, as new, and as exciting as they ever were. Just think:

What is the nature of the gap between the systems we want and the systems we program?
How can we cope with the uncertainty of the real world in the systems we build?
How can we find/maintain the overlapping conceptual models underneath programs?
How can we weave, unweave, and reweave the conceptual threads that compose programs?

In Onward!, we ask these questions, and explore for answers.

Living in the Comfort Zone	Tuesday 10:30–12:00	517A

A comfort zone is a tested region of a system's input space within which it has been observed to behave acceptably. To keep systems operating within their comfort zones, we advocate the interposition of rectifiers between systems and their input sources. Rectifiers are designed to transform inputs to ensure that they are within the comfort zone before they are presented to the system. Rectifiers enforce a highly constrained input format and, if necessary, discard information to force inputs to conform to this format. Potential benefits of this approach include the elimination of errors and vulnerabilities, the excision of undesirable excess functionality from large, complex systems, and a simplification of the computing environment. We have developed a rectifier for email messages and used this rectifier to force messages into a specific constrained form. Our results show that this rectifier can successfully produce messages that keep the Pine email client strictly within code previously confirmed (during a small testing and training session) to function acceptably. Our results also show that the rectifier completely eliminates a security vulnerability in the Pine email client. And finally, the rectifier is able to accomplish these goals while still preserving an acceptable amount of information from the original messages.

Martin Rinard , MIT

Living it up with a Live Programming Language	Tuesday 10:30–12:00	517A

A dynamic language improves programmer productivity through flexible typing, a focus on high-level programming, and by streamlining the edit-compile-debug cycle. Live languages go beyond dynamic languages with more programmer-centric features. A live language supports live programming that provides programmers with responsive and continuous feedback about how their edits affect program execution. A live language is also based on declarative programming constructs such as rules or data-flow connections so that programmers can write less code. A live language should also provide programmers with responsive semantic feedback to enable time-saving services such as code completion. This paper describes the design a textual live language known as SuperGlue. SuperGlue is based on reactive values known as signals that are supported with declarative data-flow connections and dynamic inheritance. Through signals and dynamic inheritance, SuperGlue supports live programming, declarative programming, and responsive semantic feedback. We demonstrate live programming in SuperGlue with a working prototype.

Sean McDirmid, École Polytechnique Fédérale de Lausanne (EPFL)

No Ifs, Ands, or Buts: Uncovering the Simplicity of Conditionals

Thursday 10:30–12:00 **517A**

Schematic tables are a new representation for conditionals. Roughly a cross between decision tables and data flow graphs, they represent computation and decision-making orthogonally. They unify the full range of conditional constructs, from if statements through pattern matching to polymorphic predicate dispatch. Program logic is maintained in a declarative canonical form that enforces completeness and disjointness among choices. Schematic tables can be used either as a code specification/generation tool, or as a self-contained diagrammatic programming language. They give program logic the clarity of truth tables, and support high-level direct manipulation of that logic, avoiding much of the mental computation demanded by conventional conditionals.

Jonathan Edwards, MIT

Epi-Aspects: Aspect-Oriented Conscientious Software

Thursday 10:30–12:00 **517A**

Conscientious software is a recently proposed paradigm for developing reliable, self-sustaining software systems. Conscientious software systems consist of an allopoietic part, which encapsulates application functionality, and an autopoietic part that is responsible for keeping the system alive by monitoring the application and adapting it to environmental changes. Whilst the application can be implemented in a general purpose programming language, the autopoietic part requires a special autopoietic programming language, which does not contain any elements that allow programmers to introduce critical bugs that might crash or stall the conscientious software system. Practical application of the conscientious software paradigm requires solutions to two open problems: The design of suitable lithe programming languages and the proposal of concrete architectures for combining the autopoietic and allopoietic parts. In this paper, we propose a concrete, aspect-oriented architecture for realizing conscientious software. In particular, we introduce epi-aspects, a construct for upgrading new and existing applications into conscientious software. Apart from presenting the architectural design of epi-aspects, we provide an autopoietic simulator and a concrete framework for developing epi-aspects in Java. This framework and the simulator are used to conduct a case study in which we develop and test a conscientious Java application.

Sebastian Fleissner , The Chinese University of Hong Kong
Elisa Baniassad , The Chinese University of Hong Kong

From 0 to 1,169,600 in 3 minutes: Nine months of testing by the DevCreek team

Wednesday 10:30–12:00   **517A**

One recent change in software development is developers starting to take responsibility for the quality of their work by writing and executing automated tests. As with any new activity, there is a wide range of ways to perform this task. DevCreek helps illuminate the testing process and present a visual representation of the underlying rhythms.

This film presents nine months of testing and development activity on the DevCreek tool itself in an animated form. We attempt to reveal, warts and all, the effort expended running tests, the exercised test methods, and the added test methods.

Each frame represents an additional hour of testing activity. The results of all 1.1 million test method executions are incorporated. Each principal's sequence of test run results scroll by, where each pixel-wide tick corresponds to the result of a single test run. Test methods are arranged by module and highlighted based on recent execution result. Context is provided by indicating the features added for each release and the list of new test methods.

David Jones, Intelliware Development Inc.

PowerPoint and Complexity

Wednesday 10:30–12:00   **517A**

Everyone talks about software bloat, feature-creep and the ever increasing complexity of software. Each new version of a software package adds in new features. Very rarely, features are removed. But what really happens as software evolves?

This animated film illustrates the evolution of PowerPoint over seven versions from 1987-2001. With each version the user has faced ever-increaing application complexity. Knowing how software evolves is of increasing importance as we move to building ultra-large scale software and developing software in software-ecologies.

This film uses abstract graphical representations of the application features and feature-relationships changes over time together with time-lapse graphs to give an understanding of how this application has evolved.

Michael Richmond, IBM Almaden Research Center

The Elephant in the Room: Who Will Take Care of the Code?

Wednesday 10:30–12:00   **517A**

Something extraordinarily strange is going on in the computer industry. Despite the widespread availability of jobs, the challenging and exciting nature of the work, and the impressive earning potential of workers, the industry is experiencing a massive decline in the number of new recruits. Universities and companies alike are starting to wonder: where did all the warm bodies go? Numbers from the US are showing declines upwards of 20%, and across North America it is not uncommon to find decreases in students applying to computer science post-secondary programmes that double that.

It appears that a dearth of highly-trained professionals is looming, which is bound to negatively impact future advances in computer technology and ultimately threaten major economies. Given the current crisis, many groups are addressing the issue, with varying approaches and perspectives. The film "The Elephant in the Room: Who Will Take Care of the Code?", exposes the symptoms of this problem and examines some of the current solutions. In particular, this film profiles efforts at the University of Victoria in the CScIDE (Computer Science Initiatives for Diversity and Equity) research group and their experiements introducing programming concepts to children in grades two through seven.

Rebeca Dunn-Krahn, University of Victoria   **Yvonne Coady**, University of Victoria

X3D Web Software Visualization in Action!

Wednesday 10:30–12:00   **517A**

3D web software visualization has always been expensive, special purpose, and hard to program. Most of the technologies used require large amounts of scripting, are not reliable on all platforms, are binary formats, or no longer maintained. We can make 3D software visualization of object-oriented programs cheap, portable, and easy by using X3D, which is a new open standard for web 3D graphics. In this film we show our X3D web software visualizations in action.

Craig Anslow, Victoria University of Wellington   **Stuart Marshall**, Victoria University of Wellington**Robert Biddle**, Carleton University
James Noble, Victoria University of Wellington

Panels

Chair: Joseph Yoder, The Refactory

Every year ooPSLA attracts software developers and researchers from all over the world. They gather at ooPSLA to learn the craft of software development, to understand best practices in standard technologies and methodologies, and to envision new ideas that will revolutionize how we develop software in the future. Those who attend may be fledgling programmers or industry revolutionaries. Through ooPSLA Panels, everyone has the opportunity to see the world through the perspective of leading software experts in industry and academia.

Come listen in as they debate technology trends, best practices, and the evolution of code. Panels at ooPSLA offer attendees a unique opportunity to participate in discussions. The panels promise engaging topics important to today's software engineers. The panels tackle current topics like Domain Specific Languages; programming language evolution; progression of code design; the impact of services on objects; and Service Oriented Architecture as a whole. Come participate in lively discussions centered on pertinent issues and debates. This year's panels boast a host of exciting and influential members of the programming community.

Celebrating 40 Years of Language Evolution: Simula 67 to the Present and Beyond	Tuesday 10:30–12:00	516C

Simula 67 (SIMple Universal LAnguage 67) is considered by many as one of the earliest - if not the first - object-oriented language. Simula 67 was developed by Ole-Johan Dahl and Kristen Nygaard in Oslo, Norway and has greatly influenced object-oriented language development over the past 40 years. This panel brings together leading programming language innovators to discuss and debate past, present, and future language evolutions.

Steven Fraser, Cisco Systems
James Gosling, Sun Microsystems
Anders Hejlsberg, Microsoft

Ole Lehrman Madsen, Aarhus University
Bertrand Meyer, ETH Zurich
Guy L. Steele Jr, Sun Microsystems Laboratories

Domain Specific Languages—Another Silver Bullet?	Tuesday 13:30–15:00	516C

Every few years the computing industry sees the emergence of another "silver bullet". Our "trade" press follows the siren call, vendors become optimistic (at least the marketing departments), managers and developers think that all will be solved, and the customer is skeptical as usual. Is our industry fated to go through hype cycles, and what is the underlying reality? This panel will explore this by looking at what may be a current "silver bullet"—Domain Specific Languages (DSLs). DSLs have become a focus of current interest with the increasing popularity of model driven development and dynamically typed languages. By allowing software engineers to focus on a specific domain it can be argued that DSLs bridge the gap between the business and implementation. Thus allowing us to produce better systems. Are DSLs really a "silver bullet", and what do we mean when we say that? The panel will take the form of a non-traditional debate (modeled on a public radio talk show "Left, Right, and Centre"). We have three practitioners with differing experiences using DSLs taking three differing positions: 'for', 'against', and somewhere in the 'centre'. Unlike the radio show there will also be audience participation. Keywords: Silver Bullet, Domain Specific Language, Model Driven Development.

Henry Balen, CGI
James Lapalme, CGI

Marc Frappier, University of Sherbrooke
Kevin P. Tyson, Bear, Stearns & Co. Inc

"No Silver Bullet" Reloaded—A Retrospective on "Essence and Accidents of Software Engineering"

Wednesday 11:00–12:30   **516C**

Twenty years after the paper *No Silver Bullet: Essence and Accidents of Software Engineering* by Frederick P. Brooks first appeared in IEEE Computer in April 1987 (following its 1986 publication in Information Processing, ISBN 0444-7077-3) does the premise hold that the complexity of software is not accidental? How have the "hopes for silver" which included: high-level language advances, object-oriented programming, artificial intelligence, expert systems, great designers, etc. - evolved? Panelists will discuss what has changed and/or stayed the same in the past twenty years - and the paper's influence on the community.

Steven D. Fraser, Cisco Systems
Frederick P. Brooks, Jr., UNC, Chapel Hill
Martin Fowler, ThoughtWorks
Ricardo Lopez, Qualcomm

Aki Namioka, Cisco Systems
Linda Northrop, SEI (CMU)
David Lorge Parnas, University of Limerick
David Thomas, Bedarra Labs

The Role of Objects in a Services-obsessed World

Wednesday 15:30–17:00   **516C**

Every half-decade or so, the computing world is infected by a meme that energizes IT and stimulates architectural thinking but also distorts discussion and clouds judgment. A few generations ago it was objects; now it's services. As every developer has noticed, SOA is everywhere—even, perhaps, in some places it shouldn't be. What has this focus on services done to the role of objects? Some of the more extreme SOA proponents maintain that service-reuse replaces object-reuse across the board. More mainstream architects view these two models as complementary reuse strategies at different levels of scale. There are even object diehards who think that services can't come close to the flexibility and durability of objects. This panel will represent the full range of opinions. Specifically, panelists and attendees will be challenged to explore such topics as: Where is the scalability boundary between object responsibilities and service responsibilities? Do we have to conceptualize, design, or implement differently at different levels of scale? Where and how do binary components such as RMI, EJB, or CORBA fit in alongside more loosely-coupled interfaces such as web services? Panelists will also be invited to comment on whether some of the new concepts defined into SOA—orchestration and discovery, for example—can be profitably fed back into the object world.

John Tibbetts, Kinexix
Ward Cunningham, AboutUs.org

Carl Lentz, DTE Energy
Jeroen van Tyn, DBI Consulting

The Future of SOA: What worked, what didn't and where is it going from here

Thursday 10:30–12:00   **516C**

For the past few years Service Oriented Architecture (SOA) has risen from being a concept promoted by few to become one of the most important architecture styles enterprises are trying to adopt. Though many of the basis upon which SOA is founded are not new, its link to achieving business results makes it very appealing. However, as with any new and promising concept, a great deal of hype is associated with SOA, which could spill trouble for both its promoters and for those who committed to its adoption. As of today there is no shortage of claims from both believers and doubters of its benefits. There are many challenges facing its early adopters as well as missing capabilities needed to support all business needs. The next few years are critical to the success of SOA. To silence its doubters and live up to its high expectations, SOA must evolve and extend to incorporate features from other emerging paradigms, techniques, and architecture. This panel will address the challenges facing enterprises in their attempt to adopt SOA and in realizing its full benefits and shed some light on the future of SOA that would ensure its success. The panelists will be asked to address the following questions 1) Is SOA a revolution which requires a paradigm shift (similar to OO for example)? Or is it an evolution? And what are the challenges? 2) What aspects of current SOAs do work well? What evidence do we have? Do we have specific quantified cases showing the business benefits for an enterprise as a result of adopting SOA? 3) What didn't work? (For example, large overhead, performance, methodology, etc.) 4) What does the future hold for SOA? And what are the priorities? (For example, should SOA adopt a light weight approach similar to Web 2.0? or focus on building Composite Services? How would SOA address "user needs" Should future SOA enhancements focus on incorporating Event Driven Architecture? Or on providing Hardware as a Service (HaaS)?

Mamdouh Ibrahim, IBM
Kerrie Holley, IBM
Nicolai M. Josuttis, Independent Architect

Brenda Michelson, Elemental Links, and SOA Consortium
Dave Thomas, Beddara Research Labs
John deVadoss, Microsoft

Chair: Nadyne Mielke, Microsoft Corporation

ooPSLA Posters provide an excellent forum for authors and conference attendees to discuss work on the horizon of research and practice. Posters showcase speculative, late-breaking results or introduce interesting, innovative work. The Poster session, held during the Monday Welcome Reception, is highly interactive. It allows authors and interested participants to connect to each other and to engage in discussions about the work presented.

Monday 17:00–19:00	**710**
Tuesday–Thursday 8:30–17:00	**517D**

Selecting Object-Oriented Source Code Metrics to Improve Predictive Models Using a Parallel Genetic Algorithm

Source code metrics are used as input features to predictive models of quality. There are many measures of coupling, cohesion, inheritance, and complexity. We present a prototype that uses a genetic algorithm for feature selection to improve a classifier's ability to identify cognitively complex components in a biomedical Java application.

Rodrigo A Vivanco, University of Manitoba   **Dean Jin**, University of Manitoba

Process in OO Pedagogy—The Sixth "Killer Examples" Workshop

This year's theme of the "killer examples" workshop is "process in OO pedagogy"; how to teach to "think in objects" and convey a clear and effective process for OO problem solving. The poster will summarize the workshop results and offer the ooPSLA community a forum for the exchange of experiences.

Carl Alphonce, University at Buffalo, SUNY   **Jürgen Börstler**, Umeå University
Michael Caspersen, University of Aarhus   **Adrienne Decker**, University at Buffalo, SUNY
Michael Kölling, University of Kent

OOMatch: Pattern Matching as Dispatch in Java

We present an extension to Java, dubbed OOMatch. It allows method parameters to be specified as patterns, which are matched against the arguments to the method call. When matches occur, the method applies; if multiple methods apply, the method with the more specific pattern overrides the others.

Adam J Richard, University of Waterloo   **Ondrej Lhotak**, University of Waterloo

The JastAdd Extensible Java Compiler

JastAddJ is a Java compiler that is easy to extend with new analyses and language constructs. The main systems contains a Java 1.4 compiler with modular Java5 extensions. It compares favourably in quality and size compared to other Java compilers, and runs within a factor of three compared to javac.

Torbjörn Ekman, Oxford University   **Görel Hedin**, Lund University

Open Framework for Conformance Testing via Scenarios

Scenarios are vital for the specification of software systems. We are developing an open framework for the specification, execution, and conformance evaluation of scenarios. The scenarios define a contract which is bound to an implementation under test. The scenarios are executed by our framework to ensure conformance against the contract.

Dave Arnold, Carleton University   **Jean-Pierre Corriveau**, Carleton University
Vojislav Radonjic, Carleton University

Mining Modal Scenarios from Execution Traces

We present a novel dynamic analysis framework and tool to mine inter-object modal scenario-based specifications in the form of UML2 variant of Live Sequence Charts (LSC) from execution traces of object-oriented reactive systems. A case study on *Jeti*, a popular instant messaging application, highlights the benefits of our approach.

David Lo, National University of Singapore   **Shahar Maoz**, The Weizmann Institute of Science
Siau-Cheng Khoo, National University of Singapore

Ruby Refactoring Plug-In for Eclipse

We present our refactoring plug-ins for the Eclipse Ruby Development Tools IDE. Refactoring is a very important technique for every software engineer to ensure the healthiness of code and a cornerstone of agile software development. We have implemented sixteen automated refactorings and three code generators, for example Rename Variable.

Thomas Corbat, IFS HSR Rapperswil   **Mirko Stocker**, IFS HSR Rapperswil
Lukas Felber, IFS HSR Rapperswil   **Peter Sommerlad**, IFS, HSR Rapperswil

Refactoring Support for the C++ Development Tools

We present our refactoring plug-ins for Eclipse's C++ Development Tooling (CDT). With CDT exist a good and free IDE for C/C++ developers. Unfortunately there is only a rename refactoring available. But now with our plug-in we provide several refactorings to improve the support for the developers.

Emanuel Graf, IFS, HSR Rapperswil   **Peter Sommerlad**, IFS, HSR Rapperswil
Guido Zgraggen, IFS, HSR Rapperswil

CUTE: C++ Unit Testing Easier

This article describes the design and use of the CUTE C++ testing framework and its integration into the Eclipse C++ Development Tooling. Unit testing supports code quality and is a corner stone of agile software development. CUTE and its Eclipse plug-in are an easy to use C++ testing framework.

Peter Sommerlad, IFS, HSR Rapperswil   **Emanuel Graf**, IFS, HSR Rapperswil

A Framework for Distributed Collaborative Conceptual Model Development

One of the major concerns in the processes which involve human analysts is the existence of uncertainty/inconsistency. In this paper, we propose a model based on belief theory that attempts to capture the degree of analysts' uncertainty towards their expressed specifications and uses these information to create an integrated unique model.

Ebrahim Bagheri, University of New Brunswick   **Ali A. Ghorbani**, University of New Brunswick

Towards a Pattern Language for Adaptive Object Model

The adaptive object-model architectural style makes heavy use of patterns and all the attempts to document it have been done using patterns. Nevertheless, these patterns have been written using different templates and styles. This work presents a more comprehensive and homogeneous pattern language for describing this kind of architectural style.

Leon Welicki, ONO (Cableuropa S.A.)   **Rebecca Wirfs-Brock**, Wirfs-Brock Associates
Joseph W. Yoder, The Refactory Inc   **Ralph Johnson**, University of Illinois

Combining Structural Subtyping and External Dispatch

This poster presents the design of our language, Unity. Unity has a novel subtyping system that combines both by-name and structural subtyping. With this combination, Unity provides the flexiblity of structural subtyping while still allowing static typechecking of external methods.

Donna Malayeri, Carnegie Mellon University   **Jonathan Aldrich**, Carnegie Mellon University

Green—A Flexible UML Class Diagramming Tool for Eclipse

Green is a UML class diagramming tool for the Eclipse IDE. The primary focus of Green is on its live round-tripping capabilities as well as its extensible development style. A concrete and technical overview of these areas as well as a brief look into Green's future will be shown.

Gene Wang, Universtiy at Buffalo SUNY   **Joshua Gardner**, Universtiy at Buffalo SUNY
Brian McSkimming, Universtiy at Buffalo SUNY   **Adrienne Decker**, University at Buffalo, SUNY
Zachary Marzec, Universtiy at Buffalo SUNY   **Carl Alphonce**, University at Buffalo, SUNY

Adding Rigorous Statistics to the Java Benchmarker's Toolbox

This poster advocates using a statistically rigorous data analysis methodology for reporting Java performance. The key idea is to model non-determinism (caused by JIT, GC, ...) by computing confidence intervals. We show that prevalent data analysis techniques lead to misleading or incorrect conclusions. We present a toolkit to address this.

Andy Georges, Ghent University   **Dries Buytaert**, Ghent University
Lieven Eeckhout, Ghent University

Checking Framework Plugins

Software frameworks contain many internal constraints which plugins cannot break. These constraints are relative to the context of the plugin, and they can involve multiple framework objects. This work creates a lightweight mechanism specifying semantic constraints on the framework and checking plugins against this specification.

Ciera Jaspan, Carnegie Mellon University   **Jonathan Aldrich**, Carnegie Mellon University

Swashup: Situational Web Applications Mashups

Distributed programming has shifted from private networks to the Internet using heterogeneous Web APIs. This enables the creation of composed services exposing user interfaces, i.e., mashups. However, this programmable Web lacks unified models that can facilitate mashup creation and deployments. This poster demonstrates a platform to facilitate Web 2.0 mashups.

E. Michael Maximilien, IBM Almaden Research Center   **Ajith Ranabahu**, Wright State University
Stefan Tai, IBM TJ Watson Research Center

Status report on JSR-305: Annotations for Software Defect Detection

Java Specification Request 305 defines a set of annotations that can understood by multiple static analysis tools. Rather than push the bleeding edge of static analysis, this JSR is represents an attempt to satisfy different static analysis tool vendors and the engineering issues required to make these annotations widely useful.

David Hovemeyer, York College of Pennsylvania   **William Pugh**, University of Maryland

Synthesizing Reactive Systems from LSC Requirements using the Play-Engine

In this work, we consider the development of object-based reactive systems using Live Sequence Charts. We discuss our extension to the smart play-out feature of the Play-Engine, which allows for the automatic generation of systems that are guaranteed to satisfy LSC specifications.

Hillel Kugler, New York University   **Cory Plock**, New York University
Amir Pnueli, New York University

CASE-FX: Feature Modeling Support in an OO CASE Tool

Generative Programming advocates developing a family of systems rather than a set of single systems. Feature modeling can assist in supporting the development of such software product lines through software reuse. To our knowledge, CASE-FX is the first implementation of state-level feature modeling support within a CASE tool.

Alain Forget, Carleton University   **Sonia Chiasson**, Carleton University
Dave Arnold, Carleton University

Using FindBugs On Production Software

This poster will present our experiences using FindBugs in production software development environments, including both open source efforts and Google's internal Java code base. We summarize the defects found, describe the issue of real but trivial defects, and discuss the integration of FindBugs into Google's Mondrian code review system.

Nathaniel Ayewah, University of Maryland   **John Penix**, Google
William Pugh, University of Maryland   **YuQian Zhou**, Google
David Morgenthaler, Google

Toward a Framework for Requirement Change Management in HealthCare Software Applications

Requirements volatility is an issue in software development life cycle which often originated from our incomplete knowledge about the domain of interest. In this paper, we propose an agent-based approach to manage evolving requirements in software applications using an integrated ontology-driven framework.

Arash Shaban-nejad, Concordia University   **Volker Haarslev**, Concordia University

Compile-time Type-checking for Custom Type Qualifiers in Java

We have created a backward-compatible type qualifier system for Java, including a framework for enforcing qualifier semantics at compile-time. Our poster will illustrate the creation of an example qualifier with our system and the benefits of a "nonnull" qualifier, which helps detect, eliminate, and prevent null pointer errors.

Matthew M. Papi, MIT CSAIL   **Michael D. Ernst**, MIT CSAIL

From Developer's Head to Developer Tests: Characterization, Theories, and Preventing One More Bug

Unit testing is an effective way to prevent developer bugs, but how can we build on unit testing frameworks to prevent more bugs with less effort? First, theories are developer-written statements of correct behavior over a large set of inputs. Second, characterization tools summarize observations over a large number of directed executions.

David Saff, MIT CSAIL

How Can We Liberate Ourselves From Pointers?

Pointers or references can be identified as the root cause of many fundamental problems in programming languages, resulting in underspecified object dependencies and missing hierarchical encapsulation. We therefore propose to abandon references from the language and to use expressive program relations instead.

Luc Bläser, Computer Systems Institute, ETH Zurich

Randoop: Feedback-directed Random Testing for Java

We describe Randoop, a tool that generates unit tests for Java using feedback-directed random test generation. After describing the input, output, and generation algorithm, we discuss the ways that a user can interact with Randoop to specify properties to check, methods to test, and methods to use to create assertions.

Carlos Pacheco, MIT   **Michael D. Ernst**, MIT CSAIL

Tuesday

Room	8:30–10:00	10:30–12:00	13:30–15:00	15:30–17:00
517A	Once Upon a Time, Like Never Before: The Challenge of Telling the Next Story — Peter Turchi	Creating Passionate Users (tentative) — Kathy Sierra	Second Life: The World's Biggest Programming Environment — Jim Purbrick & Mark Lentczner	Onward! Living in the Comfort Zone / Living it up with a Live Programming Language
517B		Research: Growing Java — The JastAdd Extensible Java Compiler / Jeannie: Granting Java Native Interface Developers Their Wishes / ILEA: Inter-Language Analysis across Java and C	Research: Runtime Techniques / GC — Statistically Rigorous Java Performance Evaluation / MicroPause: Proactively Invoking Garbage Collection for Improved Performance / Probabilistic Calling Context	Research: Inheritance and Visibility (15:30–17:30) — Variant Path Types for Scalable Extensibility / Dependent Classes / Component NextGen: A Sound and Expressive Component Framework for Java / User-Changeable Visibility: Resolving Unanticipated Name Clashes in Traits
516AB		Essay — Confessions of a Used Programming Language Salesman / Erik Meijer	Practitioner Reports — Effective Preparation for Design Review—Using UML Arrow Checklist Leveraged on the Gurus Knowledge / Understanding the Value of Program Analysis Tools	
517C				Invited Talk: Brian Marick — "Actor-Network Theory: Nothing to Do with TCP/IP or Distributed Objects"
516C		Panel — Celebrating 40 Years of Language Evolution: Simula 67 to the Present and Beyond / Steven Fraser, James Gosling, Anders Hejlsberg, Ole Lehrman Madsen, Bertrand Meyer, Guy L. Steele Jr.	Panel — Domain Specific Languages—Another Silver Bullet? / Henry Balen, James Lapalme, Marc Frappier, Kevin P. Tyson	
516E		DesignFest®		
510A–D		Wiki Symposium		DesignFest®
514AB		APL		Wiki Symposium
Tutorials	T30: Incremental Releases Users & Stakeholders Will Love — Jeff Patton 515C			APL

T31: Green Bar for C++—Unit Testing and Refactoring C++ — Peter Sommerlad 512C
T32: Writing Adaptable Software: Mechanisms for Implementing Variabilities in Code and Models — Markus Voelter 514C
T33: Software Best-Practices: Agile Deconstructed — Steven Fraser, David Lorge Parnas 515B

Room	10:00–12:00	15:00–17:00
512EF	Demos	Demos
512GH	Demos	Demos

	17:15–18:30	20:00–21:30
516AB	Demo: Collaborating with Jazz: A Concurrent Demonstration at OOPSLA 2007 and CASCON — Erich Gamma, John Wiegand, and Team, IBM Rational (light refreshments)	
517A		50 in 50 — Guy L. Steele Jr. / Richard P. Gabriel

Schedule (continued)

Room	13:30–15:00	15:30–17:00
517A	Elephant 2000: A Programming Language Based on Speech Acts — John McCarthy	Precise Software Documentation: Making Object-Orientation Work Better — David Lorge Parnas
517B		Research: Type and Typestate (15:30–17:30) — Modular Typestate Checking for Aliased Objects / Type Qualifier Inference for Java / Establishing Object Invariants with Delayed Types / Modular Verification of Higher-Order Methods with Mandatory Calls Specified by Model Programs
516AB		Practitioner Reports — So We Thought We Knew Money / Anticorruption: A Domain-Driven Design Approach to More Robust Integration / Agile Enterprise Software Development Using Domain-Driven Design and Test First / Panel

Room	8:30–10:00	10:30–12:00	13:30–15:00
517A	Collaboration and Telecollaboration in Design — Frederick P. Brooks, Jr.	Onward! — From 0 to 1,169,600 in 3 minutes / PowerPoint and Complexity / The Elephant in the Room / X3D Web Software Visualization in Action!	
517B	Research: Language Design — Transactions with Isolation and Cooperation / StreamFlex—High-throughput Stream Programming in Java / Can Programming be Liberated from the Two-Level Style?—Multi-Level Programming with DeepJava	Research: Software Design — The Causes of Bloat, The Limits of Health / Notation and Representation in Collaborative Object-Oriented Design / WebRB: Evaluating a Visual Domain-Specific Language For Building Relational Web-Applications	
516AB	Essay — The Transactional Memory / Garbage Collection Analogy — Dan Grossman	Panel (11:00–12:30)	Panel

Wednesday

Room	8:30–10:00	10:30–12:00		Lightning Talks	
516C	"No Silver Bullet" Reloaded — A Retrospective on "Essence and Accidents of Software Engineering" — Steven Fraser, Frederick P. Brooks, Jr., Martin Fowler, Ricardo Lopez, Aki Namioka, Linda Northrop, David Lorge Parnas, David Thomas			The Role of Objects in a Services-obsessed World — John Tibbetts, Ward Cunningham, Carl Lentz, Jeroen van Tyn	
510D / Tutorials	T35: Scripting Your (and Your Company's) Second Life — Cristina Lopes, William Cook (510A); T36: Agile in Face of Global Software Development — Jutta Eckstein (514A); T37: Use-Case Patterns and Blueprints — Gunnar Overgaard, Karin Palmkvist (514B); T38: Embedded Objects with C++: Idioms, Patterns and Architecture for Constrained Systems — Detlef Vollmann (515A); T39: Pattern-Oriented Software Architecture: A Pattern Language for Distributed Computing — Douglas Schmidt (515B)			T41: Behaviour Driven Development using JBehave — Elizabeth Keogh, Dan North (514A); T42: Fault Tolerant Software with Patterns — Robert Hanmer (514B); T43: Testing: It's Part of Your Job—Make it Effective! — Neil Harrison (515A); T44: Skills for the Agile Designer — Rebecca Wirfs-Brock (515B)	
Workshops	W15: Agility Unlimited? — Klaus Marquardt, Jens Coldewey, Johannes Link (514C)			W15: Agility Unlimited? — Klaus Marquardt, Jens Coldewey, Johannes Link (514C)	

Room	12:00–13:30	19:00–???
516AB	BoF: Collaborative Development Research using Jazz — Harold Ossher	Special Event

Thursday

Room	8:30–10:00	10:30–12:00	13:30–15:00	15:30–17:00	17:00–18:00
517A	Context, Perspective, and Programs — Gregor Kiczales	Onward! — No Ifs, Ands, or Buts: Uncovering the Simplicity of Conditionals; Epi-Aspects: Aspect-Oriented Conscientious Software	Meta-objects for the world around us — Pattie Maes	Awards	
517B	Research: Isolation and Repair — Using Early Phase Termination To Eliminate Load Imbalances At Barrier Synchronization Points; STARC: Static Analysis for Efficient Repair of Complex Data; Tracking Bad Apples: Reporting the Origin of Null and Undefined Value Errors	Research: Ownership — Inferring Aliasing and Encapsulation Properties for Java; Multiple Ownership; Ownership Transfer in Universe Types	Research: Language Specification — Lost in translation: Formalizing proposed extensions to C#; The Java Module System: core design and semantic definition; AWESOME: A Co-Weaving System for Multiple Aspect-Oriented Extension	Research: Runtime Techniques (15:30–17:30) — Scalable Omniscient Debugging; Using HPM-Sampling to Drive Dynamic Compilation; MOP: An Efficient and Generic Runtime Verification Framework; Making Trace Monitors Feasible	
517C		Invited Talk: Mark Bernstein — Intimate Information for Everyone's Everyday Tasks: What hyperfiction and new media theory teach us about programming (and teenage diaries)	Practitioner Reports — A Practical, High-Volume Software Product Line; Making Frameworks Work: A Project Retrospective	Invited Talk: Brian Foote — Big Balls of Muds: An introduction to Post-Modular Programming	
516C		Panel — The Future of SOA: What worked, what didn't and where is it going from here — Mamdouh Ibrahim, Kerrie Holley, Nicolai M. Josuttis, Brenda Michelson, Dave Thomas, John deVadoss			
510D / Tutorials	T45: Domain-Driven Design: Putting the Model to Work — Eric Evans (510B); T46: Real-time Programming on the Java Platform — David Holmes, Tony Printezis (514B); T47: Totally Awesome Computing: Python as a General-purpose Object-oriented Programming Language — Chuck Allison (514C); T48: Introduction to Software Product Line Development Methods — Charles Krueger (515B)		Lightning Talks — T21: Personas, Profiles, Actors, & Roles: Modeling users to target successful product design — Jeff Patton (510C); T49: Creating Plugins and Applications for the Eclipse Platform — Alex Romanov, Amir Kirsh (510A); T50: Domain-Driven Design: Strategic Design for Larger Systems — Eric Evans (510B); T51: Introduction to Ruby — Glenn Vanderburg (514A); T52: Building Service-Oriented Architectures with Web Services — Olaf Zimmermann (514B); T53: UML in Action — Mohamed E. Fayad (514C)		

Room	10:00–12:00	15:00–17:00	17:00–18:00
512EF	Demos (10:30–12:00)	Demos (15:00–16:30)	
512GH	Demos	Demos	
517D			Ice Cream Social

A Portable JavaScript Thread Library for Ajax Applications

We propose a portable JavaScript thread library to facilitate the development of Ajax applications. It enables programmers to write asynchronous programs using threads. Since the proposed library is implemented without modifying any existing systems, it is portable among popular Web browsers and has high anity with the existing event system.

Daisuke Maki, The University of Electro-Communications
Hideya Iwasaki, The University of Electro-Communications

Comprehending Implementation Recipes of Framework-Provided Concepts through Dynamic Analysis

Application developers often use example applications as a guide to learn how to implement a framework-provided concept. To ease applying this technique, we present a novel framework comprehension technique called FUDA. FUDA integrates dynamic slicing with clustering and data mining techniques to generate the implementation recipes of a desired concept.

Abbas Heydarnoori, University of Waterloo   **Krzysztof Czarnecki**, University of Waterloo

A Comparison of Compilation Techniques for Trace Monitors with Free Variables

A variety of different designs and optimisation strategies for trace monitoring have been proposed recently. Here, we examine trade-offs in simplicity of implementation and expressiveness of supported patterns, briefly discuss the underlying data structures of two mainstream implementations, and provide a short evaluation of the effectiveness of memory optimisations.

Pavel Avgustinov, University of Oxford   **Oege de Moor**, University of Oxford
Julian Tibble, University of Oxford

Structured Co-Evolution of Models and Web Application Platforms

Web applications exemplify the need for generative programming techniques. Some problems remain, including those that that arise from the interplay with versioning. This paper proposes addressing these problems with structured program transformations, and explores a framework for the co-evolution of platform artifacts and the models that generate them.

Adam Pingel, UCLA

Recursive Adaptive Computations Using Perobject Visitors

Adaptive Programming allows developers to write structure-shy programs. However, in Adaptive Programming, recursive computations are known to require a good deal of boiler plate code to express. This paper describes perobject visitors; a programming construct that allows developers to write recursive adaptive computations at a higher level of abstraction.

Ahmed Abdelmeged, Northeastern University   **Karl Lieberherr**, Northeastern University

A Rewriting Approach to the Design and Evolution of Object-Oriented Languages

Rewriting logic semantics provides an environment for defining formal, executable definitions of programming languages and program analysis tools. Multiple languages, including Java, Beta, and KOOL, have already been defined. At the same time, new tools and formalisms, based on rewriting logic and aimed specifically at programming languages, are being developed.

Mark Hills, University of Illinois   **Grigore Rosu**, University of Illinois

Optimizing Java Programs Using Generic Types

Our research involves improving performance of programs written in the Java programming language. By selective specialization of generic types, we can enable the compiler to eliminate typecasting, and provide type information to remove dynamic method lookup at runtime. An example of this specialization using Quicksort improved performance by over 20%.

Eli Mayfield, University of Minnesota, Morris   **Nathan Dahlberg**, University of Minnesota, Morris
Daniel Selifonov, University of Minnesota, Morris   **Elena Machkasova**, University of Minnesota, Morris
Kyle Roth, University of Minnesota, Morris

Chair: Ralph Johnson, University of Illinois

The original ooPSLA Conference organizers were a practical bunch. They knew theory was important, of course, but they were especially interested in applying theory in the real world. In fact, two of the words represented by "ooPSLA" are "Systems" and "Applications." The Practitioner Reports are where the ooPSLA community looks to hear about what's working (or not!) in leading-edge systems, applications, methodologies, frameworks, patterns, and management techniques. They are a key embodiment of the "reality check" that is an integral part of the ooPSLA Conference.

ooPSLA Practitioner Reports present actual experience and reflections, together with supporting evidence for any claims made. Of particular interest are reports that discuss both benefits and drawbacks of the approaches used. Reports may focus on a particular aspect of technology usage and practice, or describe broad project experiences. They may describe a particular design idea, or experience with a particular piece of technology. Some reports focus on people, process, or development challenges.

Tools	Tuesday 13:30–15:00	516AB

Effective Preparation for Design Review—Using UML Arrow Checklist Leveraged on the Gurus Knowledge

The process of design construction and design reviews is notorious in its endless debates and discussions. Sometime, these "gurus' style" debates are triggered by simple mistakes that could have being easily avoided. Precious time is lost, both because of the designer's lack of experience and the reviewer's overconfidence. If both sides were effectively prepared and armed with the same ammunition of knowledge, the "balance of terror" would reduce the will to argue and focus could have been maintained. Reinforcing the abilities in facing the intimidating guru's knowledge can be by formal encapsulation of that knowledge and systematic gurus' guidance to the novice-intermediate designer in his or her design steps. This paper presents a methodology that captures the recommended selection of entities' relationships using UML notation. The UML arrow methodology is comprised of checklist, which provides immediate yes/no feedback to simple guiding questions, and a governing iterative process. The consolidated questions encompass best practice design methodologies, composed, edited and corroborated by the company's experienced designers. It contains common process based on adaptations of the gurus' recommendations to the company's specific needs. The methodology was evaluated in practitioners' workshops as well as practical design sessions, based on data received by questionnaires. The results obtained from 62 participants reveal that (1) usage of this methodology lead to effective identification of inappropriate entities' relationships; (2) shortened the design review duration; (3) removed redundant technical remarks; (4) improved the preparations stages as well as brainstorming sessions, and (4) overall maintained a focused discussions regarding the problem.

Ethan Hadar, CA Labs

Understanding the Value of Program Analysis Tools

It is difficult to determine the cost effectiveness of program analysis tools because we cannot evaluate them in the same environment where we will be using the tool. This leaves program analysis tool evaluations being very subjective and usually dependent on the evaluators intuition. While we could not solve this problem, we suggest techniques to make the evaluations more objective. We use a comparative evaluation technique to make the ROI analysis more objective. We also show how to use coverage models to select several tools when they each find different kinds of issues. Finally, we suggest that the tool vendors include features that send issues found on developer's desktop to a central server so that we have the data available to run a continuous evaluation of the tool as it runs in our software process.

Ciera Christopher, Carnegie Mellon University   **I-Chin Chen**, eBay
Anoop Sharma, eBay

Domain-Driven Design   **Wednesday 15:30–17:00**   **516AB**

So We Thought We Knew Money

In Custom House's financial system, "money", "rate", "currency" and "markup" are among the key concepts that describe the domain. Without explicit modeling, those concepts appear in the code, but only in variable names of primitive types and sometime in method names for operations whose arguments and return types are all primitives. In this paper we present the experience gained when Custom House development team applies Domain-Driven Design to explicitly model domain fundamental concepts and abstract them into value objects. Using code snippets, We show how we discovered them, how we retrofitted a large existing code-base, and the effects on the system as it has evolved over time. We also report the pitfalls and stumbles during the process, which are not uncommon for projects that undergo ongoing development work. We show that, with all the incompleteness and imperfection, Domain-Driven Design has gradually accepted into real development process, making a crucial and continuous transformation.

Ying Hu, **Sam Peng**, Custom House Global Foreign Exchange

Anticorruption: A Domain-Driven Design Approach to More Robust Integration

When integrating with external or legacy systems, translating conceptual objects and actions from one model to another is a daunting task. In the absence of a well-thought-out design, when the pair of systems grows, knowledge of the external system can leak into the business domain, corrupting the domain layer and rendering it unmanageable. One solution found in Custom House is to introduce an anticorruption layer to isolate the two systems. Through the experience of redesigning the integration component, we show how we designed the anticorruption layer to encapsulate the translation logic; how the implicit model of the external system were discovered to reduce translation complexity; and the observer pattern that decoupled our business domain from the external system. After implementing this design, tasks related to integration takes no more than 10% of the total development effort—66% less than what was before. Keywords: Domain-Driven design, anti-corruption layer, domain model, integration, observer pattern

Sam Peng, **Ying Hu**, Custom House Global Foreign Exchange

Agile Enterprise Software Development Using Domain-Driven Design and Test First

In this paper we present the experience gained and lessons learned when the IT department at Statoil ASA, a large Oil and Gas company in Norway, used Domain-Driven design techniques to verify the software architecture chosen for the development of our group oil trading application. The hypothesis was that the use of object oriented techniques, domain driven design and a proper object-relational mapping tool would significantly improve the performance and reduce the code base compared to the current legacy systems. The legacy system is based on several Oracle databases serving a variety of clients written in Java, Gupta Centura Team Developer and HTML. The databases have a layer of business logic written in PL/SQL offering various system services to the clients. To validate our new object-oriented software architecture, we re-implemented one of the most computationally heavy and data intensive services using Test First and Domain Driven Design techniques. The resulting software was then tested on a set of servers with a representative subset of data from the production environment. We found through this exercise that using these techniques on our new software architecture drastically improved the performance of this service as well the quality of the resulting code when running atop our Oracle database. We also switched to an object database from Versant and achieved additional performance gains.

Einar Landre, Statoil

Reuse   **Thursday 13:30–15:00**   **517C**

A Practical, High-Volume Software Product Line

Many Software Product Line reports focus on the fact that you can achieve an ROI in 3-5 projects. This presentation asks the question "what do you have to do differently if you plan on generating 10,000 custom applications a year?" The answer includes an interesting combination of a custom specification methodology, an optimized framework, an application generation and deployment toolkit built on a databased repository, a feature selector, a decision support system, a constraint manager and an innovative set of tooling for automatically generating and accessing a databased concrete syntax for Domain Specific Languages that turns an abstract grammar into a set of tables with data access code, allowing for easier updates to DSL grammars while still getting the benefits of storing metadata in a database for automating metadata reuse. In this presentation we will look at our experiences in building the SystemsForge application generator, providing a fast paced but thorough review of the key design decisions we made and the surprises we encountered along the way.

Peter Bell, SystemsForge

Making Frameworks Work: A Project Retrospective

Software product lines have been popular for the past several years both in research and in the industry. The author recently spent three years as a developer in the industry working on a software product line project: more than two years as a framework developer and almost a year as a product developer on one of the first commercial applications of the product family. A pervading aspect of this project is the platform for the product line consisted of an enterprise object-oriented application framework built on a technology framework. This experience report discusses various issues that make framework development harder than regular development. The first part discusses how building product lines and frameworks entails increased coordination and communication between stakeholders and across the organization. The second part discusses the difficulty of building the right abstractions in a framework project, ranging from understanding the domain models to selecting and evaluating the framework architecture, to designing the right interfaces.

Marwan Abi-Antoun, Carnegie Mellon University

Chair: David F. Bacon, IBM Research

ooPSLA has been the forum for the introduction and dissemination of many key programming paradigms and methodologies. This year is no exception. This year's research papers present substantiated new research or novel technical results, advance the state of the art, and report on significant experience and experimentation.

Research: Growing Java	Tuesday 10:30–12:00	517B

Session Chair: David F. Bacon, IBM Research

The JastAdd Extensible Java Compiler

The JastAdd Extensible Java Compiler is a high quality Java compiler that is easy to extend in order to build static analysis tools for Java, and to extend Java with new language constructs. The compiler itself is built modularly, with a Java 1.4 compiler that is extended to a Java 5 compiler. Example applications that are built as extensions include an alternative backend that generates Jimple, an extension of Java with AspectJ constructs, and the implementation of a pluggable type system for non-null checking and inference. The system is implemented using JastAdd, a declarative Java-like language. We describe the compiler architecture, the major design ideas for building and extending the compiler, in particular, for dealing with complex extensions that affect name and type analysis. Our extensible compiler compares very favorably concerning quality, speed and size with other extensible Java compiler frameworks. It also compares favorably with quality and size compared with traditional non-extensible Java compilers, and it runs within a factor of four compared to javac.

Torbjörn Ekman, Oxford University   **Görel Hedin**, Lund University

Jeannie: Granting Java Native Interface Developers Their Wishes

Many programs are written in more than one language. Reasons for this include reuse of legacy libraries, access to platform functionality, and efficiency. Jeannie is a language design for cross-language integration, aiming at programmer productivity, static and dynamic error checking, portability, and efficiency. Jeannie combines two full-fledged languages: Java for high-level programming and C for low-level programming. Both Java code and C code are nested in each other in the same file, and compile down to traditional JNI. Jeannie performs static semantic checks and facilitates dynamic resource management and exception handling, two concerns that otherwise make JNI applications ugly and heavy-weight. Jeannie also lends itself for retargeting to alternative backends, such as JVM-specific native interfaces that are more efficient than the general JNI. This paper reports on our experiences from a prototype Jeannie compiler, and highlights lessons for designing and implementing foreign language interfaces.

Martin Hirzel, IBM Watson Research Center   **Robert Grimm**, New York University

ILEA: Inter-Language Analysis across Java and C

Many programs are written in more than one language. Reasons for this include reuse of legacy libraries, access to platform functionality, and efficiency. Jeannie is a language design for cross-language integration, aiming at programmer productivity, static and dynamic error checking, portability, and efficiency. Jeannie combines two full-fledged languages: Java for high-level programming and C for low-level programming. Both Java code and C code are nested in each other in the same file, and compile down to traditional JNI. Jeannie performs static semantic checks and facilitates dynamic resource management and exception handling, two concerns that otherwise make JNI applications ugly and heavy-weight. Jeannie also lends itself for retargeting to alternative backends, such as JVM-specific native interfaces that are more efficient than the general JNI. This paper reports on our experiences from a prototype Jeannie compiler, and highlights lessons for designing and implementing foreign language interfaces.

Gang Tan, Boston College   **Greg Morrisett**, Harvard University

Session Chair: Cliff Click, Azul Systems

Statistically Rigorous Java Performance Evaluation

Evaluating Java performance is non-trivial. Non-determinism at run-time causes the execution time of a Java program to differ from run to run of the same program. There are a number of sources of non-determinism such as Just-In-Time (JIT) compilation and optimization in the virtual machine (VM) driven by method sampling, thread scheduling, garbage collection, and various system effects. There exists a wide variety of Java performance evaluation methodologies that researchers and benchmarkers use to deal with the non-determinism in their experiments. All of these methodologies differ from each other in a number of ways. Some report for example average performance over a number of runs of the same experiment; others report the best or second best performance observed; yet others report the worst. Some iterate the benchmark multiple times within a single VM invocation; others consider multiple VM invocations and iterate a single benchmark execution; yet others consider multiple VM invocations and iterate the benchmark multiple times. Some eliminate the non-determinism due to JIT compilation by fixing the compiler work for each benchmarking experiment. This paper shows that these prevalent methodologies can be misleading, or can even lead to incorrect conclusions. The reason for not being solid is that these methodologies are not statistically rigorous. In this paper, we propose a practical and statistically rigorous Java performance evaluation methodology. The key idea is to deal with non-determinism by computing confidence intervals. We propose approaches to quantify startup performance as well as steady-state performance, and, in addition, we provide software to automatically obtain performance numbers in a rigorous manner. Although this paper focuses on Java performance evaluation, many of the issues addressed in this paper also apply to other programming languages and systems that build on a managed runtime system.

Andy Georges, Ghent University   **Lieven Eeckhout**, Ghent University
Dries Buytaert, Ghent University

MicroPause: Proactively Invoking Garbage Collection for Improved Performance

To date, the most commonly used criterion for invoking garbage collection (GC) is based on heap usage; that is garbage collection is invoked when the heap or an area inside the heap is full. This approach can suffer from two performance shortcomings, untimely garbage collection invocations and large volumes of surviving objects. In this work, we explore a new GC triggering approach called MicroPause that identifies proper locations to invoke garbage collection yielding high efficiency. MicroPause leverages two observations that (i) allocation requests occur in phases and (ii) the phase boundaries coincide with times when most objects also die. Thus, our technique does not untimely invoke garbage collection; therefore, it can significantly reduce the volume of recently created objects that have yet to die. We extended the HotSpot virtual machine from Sun Microsystems to support MicroPause and conducted experiments using 12 benchmarks. We found that MicroPause can reduce the overall GC time over that of the default HotSpot collector in every application (with the maximum reduction of 26%) under tight heap situations. As a result, MicroPause can improve the overall erformance of most benchmarks by as much as 14%.

Feng Xian, University of Nebraska-Lincoln   **Hong Jiang**, University of Nebraska-Lincoln
Witawas Srisa-an, University of Nebraska-Lincoln

Probabilistic Calling Context

Growing software complexity (1) often makes whole-program static analysis and exhaustive testing infeasible and (2) limits developers' ability to have complete program knowledge. Object-oriented programming features such as dynamic method dispatch obscure whole program behavior as well. Due to these factors, programmers are turning to dynamic analyses to assist with programming understanding. Previous work shows that *calling context* improves testing, debugging, and detecting security violations. However, computing calling context is very expensive in time and space, e.g., call stack walking, maintaining a location in the calling context tree, and interprocedural path profiles. This paper introduces a new online approach called *probabilistic calling context* (PCC) that efficiently maintains a probabilistically unique value representing the current context by computing an associative, non-commutative function at each call site. We demonstrate that this approach adds only 3% overhead to a Java virtual machine, and that application use case scenarios, such as detecting new contexts for coverage testing and checking for anomalous calling contexts at system calls, can efficiently query PCC values. We demonstrate in theory and practice that a 32-bit PCC value has relatively few conflicts for millions of unique contexts. This approach is efficient and accurate enough to use in deployed software for residual testing, bug detection, and intrusion detection.

Michael D. Bond, University of Texas at Austin   **Kathryn S. Mckinley,** University of Texas at Austin

Research: Inheritance and Visibility   Tuesday 15:30–17:30   517B

Session Chair: Yannis Smaragdakis, University of Oregon

Variant Path Types for Scalable Extensibility

Much recent work in the design of object-oriented programming languages has been focusing on identifying suitable features to support so-called scalable extensibility, where the usual extension mechanism by inheritance works in different scales of software components - that is, classes, groups of classes, groups of groups and so on. Mostly, this issue has been addressed by means of dependent type systems, where nested types are seen as properties of objects. In this work, we seek instead for a different solution, which can more simply apply to the Java approach of nested types as properties of classes. We introduce the mechanism of variant path types, which provides a flexible means to intra-group relationship (among classes) that has to be preserved through extension. Improving and extending existing works on groups and exact types, we feature the new notions of exact and inexact qualifications, providing rich abstractions to express various kinds of set of objects, thanks to a flexible subtyping mechanism. We formalize a safe type system for variant path types on top of Featherweight Java. Our development results in a complete solution for scalable extensibility, similarly to previous attempts based on dependent type systems.

Atsushi Igarashi, Kyoto University
Mirko Viroli, Alma Mater Studiorum—Università di Bologna

Dependent Classes

Virtual classes allow to define dependent abstractions: By being nested within an enclosing class they become attributes of its objects and their definition depends on and can vary with the type of their enclosing object. Expressing dependency via nesting, however, has two limitations: Abstractions that depend on more than one object cannot be modeled and a class must know all classes that depend on its objects. This paper presents the notion of a dependent class, a generalization of a virtual class that expresses similar semantics by parameterization rather than by nesting. This results in increased variability, less coupling between classes, and better extensibility. Besides, dependent classes nicely complement multi-methods in scenarios where multi-dispatched abstractions rather than multi-dispatched methods are needed. They can also be used to express more precise signatures of multi-methods and even extend their dispatch semantics. We present a formal semantics of dependent classes and a machine-checked type soundness proof in Isabelle/HOL, the first of this kind for a language with virtual classes and path-dependent types.

Vaidas Gasiunas, Darmstadt University of Technology   **Klaus Ostermann**, Darmstadt
Mira Mezini, Darmstadt

Component NextGen: A Sound and Expressive Component Framework for Java

Developing a general component system for an object-oriented language is a challenging design problem because inheritance across component boundaries can cause accidental method overrides. In addition, mutually recursive references across components are common in object-oriented programs--an issue that has proven troublesome in the context of component systems for functional and procedural languages. Our recent research has shown that a component framework can be constructed for a nominally typed object-oriented language supporting first-class generic types simply by adding appropriate annotations and syntactic sugar. The fundamental semantic building blocks for constructing, type-checking and manipulating components are provided by the underlying first-class generic type system. To demonstrate the simplicity and utility of this approach we have designed and implemented an extension of Java called Component NEXTGEN (CGEN). CGEN, which is based on the Sun Java 5.0 javac compiler, is backwards compatible with existing Java binary code and runs on current Java Virtual Machines. The primary contribution of this paper is an in-depth analysis of the subtle design issues involved in building a component framework for a nominally typed object-oriented language supporting first-class generics. In contrast to component systems for structurally typed languages, mutual recursion among components is accommodated in the type system and semantics without incorporating any special machinery. Our analysis includes a presentation of Core CGEN (CCG), a small, core language modeling the CGEN framework. It is based on Featherweight GJ and incorporates some ideas from MIXGEN. CCG adds the essential features to support components, but nothing more. Our discussion includes the type rules and semantics for CCG, as well as a proof of type safety.

James Sasitorn, Rice University   **Robert Cartwright**, Rice University

User-Changeable Visibility: Resolving Unanticipated Name Clashes in Traits

A trait is a unit of behaviour that can be composed with other traits and used by classes. Traits are an alternative to multiple inheritance. Conflict resolution of traits, while flexible, does not completely handle accidental method name conflicts: if a trait with method m is composed with another trait defining a different method m then resolving the conflict Mayo prove delicate or infeasible in certain cases. In this paper we present freezeable traits that provide an expressive composition mechanism to support unanticipated method composition conflicts. Our solution introduces private trait methods and lets the class composer change method visibility at composition time (from private to public and vice versa), something which is not possible in mainstream languages. Two class composers Mayo use different composition policies for the same traits. This approach respects the two main design principles of traits: the class composer is empowered and traits can be flattened away. We present an implementation of freezable traits in Smalltalk. As a side-effect of this implementation we introduced private (early-bound and invisible) methods to Smalltalk by distinguishing object-sends from self-sends. Our implementation uses compile-time bytecode manipulation and, as such, introduces no run-time penalties.

Stéphane Ducasse, LISTIC—Université de Savoie
Roel Wuyts, IMEC and Université Libre de Bruxelles

Alexandre Bergel, LERO, DSG Trinity College Dublin
Oscar Nierstrasz, SCG, University of Berne

Research: Language Design | Wednesday 8:30–10:00 | 517B

Session Chair: William Cook, The University of Texas at Austin

Transactions with Isolation and Cooperation

We present the TIC (Transactions with Isolation and Cooperation) model for concurrent programming. TIC adds to standard transactional memory the ability for a transaction to observe the effects of other threads at selected points. This allows transactions to cooperate, as well as to invoke non-repeatable or irreversible operations, such as I/O. Cooperating transactions run the danger of exposing intermediate state and of having other threads change the transaction's state. The TIC model protects against unanticipated interference by having the type system keep track of all operations that Mayo (transitively) violate the atomicity of a transaction and require the programmer to re-establish consistency. The result is a programming model that is both general and simple. We have used the TIC model to re-engineer existing lock-based applications including a substantial multi-threaded web mail server and a memory allocator with coarse-grained locking. Our experience confirms the features of the TIC model: it is convenient for the programmer, while maintaining the benefits of transactional memory.

Yannis Smaragdakis, University of Oregon
Tony Kay, University of Oregon
Reimer Behrends, University of Oregon
Michal Young, University of Oregon

StreamFlex—High-throughput Stream Programming in Java

The stream programming paradigm aims to expose coarse-grained parallelism in applications that must process continuous sequences of events. The appeal of stream programming comes from its conceptual simplicity. A program is a set of independent filters which communicate exclusively by the means of unidirectional data channels. The model reduces opportunities for data races as the output behavior of a filter is completely determined by the state of its input channels. Previous work has shown that stream programs can be implemented very efficiently on modern multiprocessors. StreamFlex is an extension to Java which marries streams with objects and allows programmers to integrate stream processors with traditional object-oriented components in the same virtual machine. StreamFlex targets high-throughput low-latency application with stringent quality-of-service requirements. To achieve these goals, we must both extend and restrict Java. In order to allow for program optimization and provide latency guarantees, the StreamFlex compiler restricts Java in that it imposes a stricter typing discipline on the code to be run in filters. On the other hand, StreamFlex extends the Java virtual machine with transactional channels and type-safe region-based allocation. The result is a rich language that can be implemented efficiently. In our experiments, we have been able to run a streaming task, concurrently with some plain Java threads, periodically every 80 microseconds with less than 2% deadline misses.

Jesper Spring, EPFL
Jean Privat, Purdue University
Rachid Guerraoui, EPFL
Jan Vitek, Purdue University

Can Programming be Liberated from the Two-Level Style?—Multi-Level Programming with DeepJava

Since the introduction of object-oriented programming, few programming languages have attempted to provide programmers with more than objects and classes, i.e., more than two levels. Those that introduce further levels almost exclusively aim at describing language properties -- i.e., their metaclasses exert linguistic control on language concepts and mechanisms -- often in order to make the language extensible. In terms of supporting logical domain classification levels, however, they are still limited to two levels. In this paper we conservatively extend the object-oriented programming paradigm to feature an unbounded number of domain classification levels. We can therefore avoid the introduction of accidental complexity into programs caused by accommodating multiple domain levels within only two programming levels. We present a corresponding language design featuring "deep instantiation" and demonstrate its features with a running example. Finally, we outline the implementation of our compiler prototype and discuss the potentials of further developing our language.

Thomas Kuehne, Darmstadt
Daniel Schreiber, Darmstadt University of Technology

Session Chair: David Holmes, Sun Microsystems

The Causes of Bloat, The Limits of Health

Applications often have large runtime memory requirements. In some cases, large memory footprint helps accomplish an important functional, performance, or engineering requirement. A large cache, for example, Mayo ameliorate a pernicious performance problem. In general, however, finding a good balance between memory consumption and other requirements is quite challenging. To do so, the development team must distinguish effective from excessive use of memory. We introduce health signatures to enable these distinctions. Using data from dozens of applications and benchmarks, we show that they provide concise and application-neutral summaries of footprint. We show how to use them to form value judgments about whether a design or implementation choice is good or bad. We show how being independent of any application eases comparison across disparate implementations. We demonstrate the asymptotic nature of memory health: certain designs are limited in the health they can achieve, no matter how much the data size scales up. Finally, we show how to use health signatures to automatically generate formulas that predict this asymptotic behavior, and show how they enable powerful limit studies on memory health.

Nick Mitchell, IBM T.J. Watson Research Center   **Gary Sevitsky**, IBM T.J. Watson Research Center

Notation and Representation in Collaborative Object-Oriented Design

While modeling tools and standard notations such as UML are available to designers, casual observation of collaborative and col-located object-oriented design suggests that teams tend to use physical mediums to sketch a plethora of informal diagrams in varied representations that often diverge from UML. Existing works focused on the eventual transformation of these sketches into finalized UML models. However, to fully understand collaborative design and develop tools for supporting its early stages, we need to understand the origins, roles, uses, and implications of these representations. To this end we conducted observational studies of collaborative design exercises at the OOPSLA DesignFest event, focused on the use of notations and diagrams. As expected, UML was not used in its ``traditional'' form, although many of its notations were borrowed as idioms. Our primary finding is that designs span multiple diagrams in representations that are often improvised in response to ad-hoc needs and the evolution of the design, and teams rely on additional communication mediums and on contextual information to compensate. The resulting artifacts are difficult to interpret without this additional information, and have limited documentation potential. Supporting tools should thus focus on preserving contextual information.

Uri Dekel, Carnegie Mellon University   **James Herbsleb**, Carnegie Mellon University

WebRB: Evaluating a Visual Domain-Specific Language For Building Relational Web-Applications

Many web-applications can be characterized as "relational". In this paper we introduce and evaluate WebRB, a visual domain-specific language for building such applications. WebRB addresses the limitations of the conventional "imperative-embedding" approach typically used to build relational web-applications. We describe the WebRB language, present extended examples of its use, and discuss the WebRB visual editor, libraries, and runtime. We then evaluate WebRB by comparing it to alternative approaches, and demonstrate its effectiveness in building relational web-applications.

Avraham Leff, IBM T. J. Watson Research Center   **James T Rayfield**, IBM T. J. Watson Research Center

Research: Type and Typestate   Wednesday 15:30–17:30   517B

Session Chair: Mandana Vaziri, IBM Research

Modular Typestate Checking for Aliased Objects

Objects often define usage protocols that clients must follow in order for these objects to work correctly. Aliasing makes it notoriously difficult to check whether clients and implementations are compliant with such protocols. Accordingly, existing approaches either operate globally or severely restrict aliasing.

We have developed a sound modular protocol checking approach, based on typestates, that allows a great deal of flexibility in aliasing while guaranteeing the absence of protocol violations at runtime. The main technical contribution is a novel abstraction, access permissions, that combines typestate and object aliasing information. In our methodology, developers express their protocol design intent through annotations based on access permissions. Our checking approach then tracks permissions through method implementations. For each object reference the checker keeps track of the degree of possible aliasing and is appropriately conservative in reasoning about that reference. This helps developers account for object manipulations that Mayoo occur through aliases. The checking approach handles inheritance in a novel way, giving subclasses more flexibility in method overriding. Case studies on Java iterators and streams provide evidence that access permissions can model realistic protocols, and protocol checking based on access permissions can be used to reason precisely about the protocols that arise in practice.

Kevin Bierhoff, Carnegie Mellon University   **Jonathan Aldrich**, Carnegie Mellon University

Type Qualifier Inference for Java

Java's type system provides programmers with strong guarantees of type and memory safety, but there are many important properties not captured by standard Java types. We describe JQual, a tool that adds user-defined type qualifiers to Java, allowing programmers to quickly and easily incorporate extra lightweight, application-specific type checking into their programs. JQual provides type qualifier inference, so that programmers need only add a few key qualifier annotations to their program, and then JQual infers any remaining qualifiers and checks their consistency. We explore two applications of JQual. First, we introduce opaque and enum qualifiers to track C pointers and enumerations that flow through Java code via the JNI. In our benchmarks we found that these C values are treated correctly, but there are some places where a client could potentially violate safety. Second, we introduce a readonly qualifier for annotating references that cannot be used to modify the objects they refer to. We found that JQual is able to automatically infer readonly in many places on method signatures. These results suggest that type qualifiers and type qualifier inference are a useful addition to Java. keywords: type qualifiers, type inference, readonly, immutability, JNI, field-sensitivity

David Greenfieldboyce, University of Maryland   **Jeffrey S. Foster**, University of Maryland

Establishing Object Invariants with Delayed Types

Mainstream object-oriented languages such as C# and Java provide an initialization model for objects that does not guarantee programmer controlled initialization of fields. Instead, all fields are initialized to a 0-equivalent default value on allocation. This is in stark contrast to functional languages, where all parts of an allocation are initialized to programmer-provided values. These choices have a direct impact on two main issues: 1) the prevalence of null in object oriented languages (and its general absence in functional languages), and 2) the ability to initialize circular data structures. This paper explores connections between these differing approaches and proposes a fresh look at initialization. Delayed types are introduced to express and formalize prevalent initialization patterns in object-oriented languages.

Manuel Fahndrich, Microsoft Research   **Songtao Xia**, Microsoft Research

Modular Verification of Higher-Order Methods with Mandatory Calls Specified by Model Programs

What we call a ``higher-order method'' (HOM) is a method that makes mandatory calls to other dynamically dispatched methods. Examples include template methods as in the Template method design pattern and notify methods in the Observer pattern. HOMs are particularly difficult to reason about, because standard pre- and postcondition specifications cannot describe the mandatory calls. For reasoning about such methods, existing approaches use either higher-order logic or traces, but both are unintuitive and verbose. We describe a simple, intuitive, and modular approach to specifying HOMs We show how to verify calls to HOMs and their code using first-order verification conditions, in a sound and modular way. Verification of client code that calls HOMs can take advantage of the client's knowledge about the mandatory calls to make strong conclusions. Our verification technique validates and explains traditional documentation practice for HOMs, which typically shows their code. However, specifications do not have to expose all of the code to clients, but only enough to determine how the HOM makes its mandatory calls.

Steve M. Shaner, Iowa State University   **David A. Naumann**, Stevens Institute of Technology
Gary T. Leavens, Iowa State University

Session Chair: Steve Backburn, Australian National University

Using Early Phase Termination To Eliminate Load Imbalances At Barrier Synchronization Points

We present a new technique, early phase termination, for eliminating idle processors in parallel computations that use barrier synchronization. This technique simply terminates each parallel phase as soon as there are too few remaining tasks to keep all of the processors busy.

Although this technique completely eliminates the idling that would otherwise occur at barrier synchronization points, it also changes the computation and therefore the result that the computation produces. We address this issue by providing probabilistic distortion models that characterize how the use of early phase termination distorts the result that the program produces. Our experimental results show that for our set of benchmark applications, (1) early phase termination can significantly improve the parallel performance, (2) the distortion is small, or can be made to be small with the use of an appropriate compensation technique, and (3) the distortion models provide accurate and tight distortion bounds. These bounds can enable users to confidently evaluate the effect of early phase termination and confidently accept results from parallel computations that use this technique if they find the distortion bounds to be acceptable.

Martin Rinard, MIT

STARC: Static Analysis for Efficient Repair of Complex Data

Data structure corruptions are insidious bugs that reduce the reliability of software systems. Constraint-based data structure repair is a promising idea that enables programs to recover from potentially crippling corruption errors. Prior work has feasibly repaird a variety of data structures that are relatively small in size (up to a thousand nodes). We present STARC, a framework for efficient large data structure repair using static analysis. Given a Java predicate method that describes the integrity constraints of a structure, STARC statically analyzes the method to identify: (1) the recurrent fields, i.e., fields that the predicate method uses primarily to traverse the structure; and (2) local field constraints, i.e., how the value of an object field is related to the value of a neighboring object fields. To repair a structure that violates the integrity constraints, STARC executes the predicate method on the structure and monitors its execution to identify corrupt object fields, which STARC then repairs using a systematic search of a neighborhood of the given structure. Each repair action is guided by the result of the static analysis, which enables efficient and effective repair. Experimental results show that STARC can repair structures that are up to 100 times larger than those possible with previous constraint-based repair algorithms. Using static analysis to scale the performance of constraint-based repair algorithms is a promising idea as it enables such algorithms to handle structures with tens of thousands of nodes.

Kathryn S. Mckinley, University of Texas at Austin   **Sarfraz Khurshid**, University of Texas at Austin
Bassem Elkarablieh, University of Texas at Austin   **Duy Vu**, University of Texas at Austin

Tracking Bad Apples: Reporting the Origin of Null and Undefined Value Errors

Despite extensive testing, deployed software still crashes. Other than a stack trace, these crashes offer little guidance to developers, making them hard to reproduce and fix. This work seeks to ease error correction by providing diagnostic information about the origins of null pointer exceptions and undefined variables. The key idea is to use *value piggybacking* to record and report useful debugging information in undefined memory. For example, instead of storing zero at a null store, store the *origin* program location, then correctly propagate this value through assignment statements and comparisons. If the program dereferences this value, report the origin. We describe, implement, and evaluate low-overhead value piggybacking for *origin tracking* of null pointer exceptions in deployed Java programs. We show that the reported origins add useful debugging information over a stack trace. We also describe, implement, and evaluate origin tracking of undefined values in C, C++, and Fortran programs in a memory error testing tool built with Valgrind. Together these implementations demonstrate that value piggybacking yields useful debugging information that can ease bug diagnosis and repair.

Michael D. Bond, University of Texas at Austin   **Stephen W. Kent**, University of Texas at Austin
Kathryn S. Mckinley, University of Texas at Austin   **Samuel Z. Guyer**, Tufts University
Nicholas Nethercote, National ICT Australia

Research: Ownership   **Thursday 10:30–12:00**   **517B**

Session Chair: Jan Vitek, Purdue University

Inferring Aliasing and Encapsulation Properties for Java

There are many proposals for language techniques to control aliasing and encapsulation in object oriented programs, typically based on notions of object ownership and pointer uniqueness. Most of these systems require extensive manual annotations, and thus there is little experience with these properties in large, existing Java code bases. To remedy this situation, we present Uno, a novel static analysis for automatically inferring ownership, uniqueness, and other aliasing and encapsulation properties in Java. Our analysis requires no annotations, and combines an intraprocedural points-to analysis with an interprocedural, demand-driven predicate resolution algorithm. We have applied Uno to a variety of Java applications and found that some aliasing properties, such as temporarily lending a reference to a method, are common, while others, in particular strict object ownership, are rare. As a result, we believe that Uno can be a valuable tool for discovering and understanding aliasing and encapsulation in Java programs.

Jeffrey S. Foster, University of Maryland   **Kin-keung Ma**, University of Maryland, College Park

Multiple Ownership

Existing ownership type systems require objects to have precisely one primary owner, organizing the heap into an ownership tree. Unfortunately, a tree structure is too restrictive for many programs, and prevents many common design patterns where multiple objects interact. Multiple Ownership is an Ownership Types system where objects can have more than one owner, and the resulting ownership structure forms a DAG. We give a straightforward model for multiple ownership, focusing in particular on how multiple ownership can support a powerful effect system that determines when two computations interfere - in spite of the DAG structure. We present a core programming language MOJO, (Multiple Ownership for Java-like Objects), including a type and effect system, and soundness proof. In comparison to other systems, MOJO imposes absolutely no restrictions on pointers, modifications or programs' structure, but in spite of this, MOJO's effects can be used to reason about or describe programs' behaviour.

Nicholas Cameron, Imperial College London   **James Noble**, Victoria University of Wellington
Sophia Drossopoulou, Imperial College London   **Matthew Smith**, Imperial College London

Ownership Transfer in Universe Types

Ownership simplifies reasoning about object-oriented programs by controlling aliasing and modifications of objects. Several type systems have been proposed to express and check ownership statically. For ownership systems to be practical, they must allow objects to migrate from one owner to another. This ownership transfer is common and occurs, for instance, in the Factory pattern, when data structures are merged, and during the initialization of data structures. However, existing ownership type systems either do not support ownership transfer at all or they are too restrictive, give rather weak static guarantees, or require a high annotation overhead. In this paper, we present UTT, an extension of Universe Types that supports ownership transfer. UTT combines ownership type checking with a modular static analysis to control references to transferable objects. UTT is very flexible because it permits temporary aliases, even across certain method calls. Nevertheless, it guarantees statically that a cluster of objects is externally-unique when it is transferred and, thus, that ownership transfer is type safe. UTT provides the same encapsulation as Universe Types and requires only negligible annotation overhead.

Peter Müller, ETH Zurich   **Arsenii Rudich**, ETH Zurich

Research: Language Specification   Thursday 13:30–15:00   517B

Session Chair: Dan Grossman, University of Washington

Lost in translation: Formalizing proposed extensions to C#

Current real-world software applications typically involve heavy use of relational and XML data and their query languages. Unfortunately object-oriented languages and database query languages are based on different semantic foundations and optimization strategies. The resulting "ROX impedance mismatch" currently makes life very difficult for developers. Microsoft Corporation is currently developing extensions to the .NET framework to facilitate easier processing of non-object-oriented data models. Part of this project (known as "LINQ") includes various extensions to the .NET languages to leverage this support. In this paper we consider the current proposals for C# 3.0, the next version of the C# programming language. We give both an informal introduction to the new language features, and also a precise formal account by defining a translation from C# 3.0 to C# 2.0. This translation also demonstrates how these language extensions do not require any changes to the underlying CLR.

Erik Meijer, Microsoft Corporation   **Mads Torgersen**, Microsoft Corporation
Gavin Bierman, Microsoft Research Cambridge

The Java Module System: core design and semantic definition

Java has no module system. Its packages only subdivide the class namespace, allowing only a very limited form of component-level information hiding and reuse. Two Java Community Processes have started addressing this problem: one describes the runtime system and has reached an early draft stage, while the other considers the developer's view and only has a straw-man proposal. Both are natural language documents, with its inevitable ambiguities. In this work we design and formalize a core module system for Java. Where the JCP documents are complete, we follow them closely; elsewhere we make reasonable choices. We define the syntax, the type system, and the operational semantics of an LJAM language, defining these rigorously in the Isabelle/HOL automated proof assistant. Using this formalization, we identify various issues with the module system, including incompatibilities with the existing Java intra-package visibility semantics. We highlight the underlying design decisions, and discuss various alternatives and their benefits. Our Isabelle/HOL definitions should provide a basis for further consideration of the design alternatives, for reference implementations, and for proofs of soundness.

Rok Strniša, University of Cambridge   **Matthew Parkinson**, University of Cambridge
Peter Sewell, University of Cambridge

AWESOME: A Co-Weaving System for Multiple Aspect-Oriented Extension

Different aspect extensions offer unique capabilities to deal with a variety of crosscutting concerns. Ideally, one should be able to use several of these extensions together in a single program. Unfortunately, each extension generally implements its own specialized weaver and the different weavers are incompatible. Even if the weavers were compatible, combining them is a difficult problem to solve in general, because each extension defines a new language with new semantics. In this paper we present a practical composition framework for building a multi-extension weaver. The multi-weaver is constructed by plugging together independently developed aspect mechanisms. To be general, the framework provides means for customizing the composition behavior. Furthermore, to be practically useful, the affect of the framework on the runtime performance of compiled aspect programs is negligible. To illustrate the architecture concretely, we demonstrate the construction of a weaver for a multi-extension AOP language that combines COOL and AspectJ. However, the method is not exclusive to COOL and AspectJ—t can be applied to combine any comparable reactive aspect mechanisms.

Sergei Kojarski, University of Virginia   **David H Lorenz**, University of Virginia

Research: Runtime Techniques   **Thursday 15:30–17:30**   **517B**

Session Chair: David Gay, Intel Research Berkeley

Scalable Omniscient Debugging

Omniscient debuggers make it possible to navigate backwards in time within a program execution trace, drastically improving the task of debugging complex applications. Still, they are mostly ignored in practice due to the challenges raised by the potentially huge size of the execution traces. This paper shows that omniscient debugging can be realistically realized through the use of different techniques addressing efficiency, scalability and usability. We present TOD, a portable Trace-Oriented Debugger for Java, which combines an efficient instrumentation for event generation, a specialized distributed database for scalable storage and efficient querying, support for partial traces in order to reduce the trace volume to relevant events, and innovative interface components for interactive trace navigation and analysis in the development environment. Provided a reasonable infrastructure, the performance of TOD allows a responsive debugging experience in the face of large programs.

Guillaume Pothier, DCC—University of Chile   **José Piquer**, DCC—University of Chile
Éric Tanter, DCC—University of Chile

Using HPM-Sampling to Drive Dynamic Compilation

All high-performance production JVMs employ an adaptive strategy for program execution. Methods are first executed unoptimized and then an online profiling mechanism is used to find a subset of methods that should be optimized during the same execution. This paper empirically evaluates the design space of several profilers for dynamic compilation and shows that existing online profiling schemes suffer from several limitations. They provide an insufficient number of samples, are untimely, and have limited accuracy. We describe and comprehensively evaluate HPM-sampling, a simple but effective profiling scheme for finding optimization candidates using hardware performance monitors (HPMs) that addresses the aforementioned limitations. We show that HPM-sampling is more accurate; has low overhead; and improves performance by 5.7% on average and up to 18.3% when compared to the default system in Jikes RVM.

Dries Buytaert, Ghent University   **Matthew Arnold**, IBM T.J. Watson Research Center
Andy Georges, Ghent University   **Lieven Eeckhout**, Ghent University
Michael Hind, IBM T.J. Watson Research Center   **Koen De Bosschere**, Ghent University

MOP: An Efficient and Generic Runtime Verification Framework

Monitoring-Oriented Programming (Not to be confused with "meta-object protocol") is a formal framework for software development and analysis, in which the developer specifies desired properties using definable specification formalisms, along with code to execute when properties are violated or validated. The MOP framework automatically generates monitors from the specified properties and then integrates them together with the user-defined code into the original system. The previous design of MOP only allowed specifications without parameters, so it could not be used to state and runtime verify safety properties referring to two or more related objects. In this paper we propose a parametric specification-formalism-independent extension of MOP, together with an implementation of JavaMOP that supports parameters. In our current implementation, parametric specifications are translated into AspectJ code and then weaved into the application using off-the-shelf AspectJ compilers; hence, MOP specifications can be seen as formal or logical aspects. Our JavaMOP implementation was extensively evaluated on two benchmarks, Dacapo and Tracematches, showing that runtime verification in general and MOP in particular are feasible. In some of the examples, millions of monitor instances are generated, each observing a set of related objects. To keep the runtime overhead of monitoring and event observation low, we devised and implemented a decentralized indexing optimization. Less than 8% of the experiments showed more than 10% runtime overhead; in most cases our tool generates monitoring code as efficient as the hand-optimized code. Despite its genericity, JavaMOP is empirically shown to be more efficient than runtime verification systems specialized and optimized for particular specification formalisms. Many property violations were detected during our experiments; some of them are benign, others indicate defects in programs. Many of these are subtle and hard to find by ordinary testing.

Feng Chen, University of Illinois   **Grigore Rosu**, University of Illinois

Making Trace Monitors Feasible

A trace monitor observes an execution trace at runtime; when it recognises a specified sequence of events, the monitor runs extra code. In the aspect-oriented programming community, the idea originated as a generalisation of the advice-trigger mechanism: instead of matching on single events (joinpoints), one matches on a sequence of events. The runtime verification community has been investigating similar mechanisms for a number of years, specifying the event patterns in terms of temporal logic, and applying the monitors to hardware and software. In recent years trace monitors have been adapted for use with mainstream object-oriented languages. In this setting, a crucial feature is to allow the programmer to quantify over groups of related objects when expressing the sequence of events to match. While many language proposals exist for allowing such features, until now no implementation had scalable performance: execution on all but very simple examples was infeasible. This paper rectifies that situation, by identifying two optimisations for generating feasible trace monitors from declarative specifications of the relevant event pattern. We restrict ourselves to optimisations that do not have a significant impact on compile-time: they only analyse the event pattern, and not the monitored code itself. The first optimisation is an important improvement over an earlier proposal in OOPSLA'05 to avoid space leaks. The second optimisation is a form of indexing for partial matches. Such indexing needs to be very carefully designed to avoid introducing new space leaks, and the resulting data structure is highly non-trivial.

Pavel Avgustinov, University of Oxford   **Julian Tibble**,PUniversity of Oxford
Oege De Moor, University of Oxford

Chair: Nadyne Mielke, Microsoft Corporation

After its remarkable success in previous years, ooPSLA is again hosting an ACM SIGPLAN Student Research Competition (sponsored by Microsoft Research). The first round of evaluation will be held jointly with the ooPSLA Poster Session. The ACM SIGPLAN Student Research Competition shares the Posters session's goal of facilitating interaction among researchers and attendees, to provide both sides with the opportunity to learn of ongoing, current research. Additionally, the Student Research Competition affords students with experience with both formal presentations and evaluations..

Student Research Competition	Tuesday 8:30–13:30	512C

CodeGenie: A Tool for Test-Driven Source Code Search

In CodeGenie, test cases are designed for a desired feature. The feature is then searched based on information available on the tests. Each matching code result is woven and tested locally using the original tests. The most suitable code is then reused. Woven code results can also be unwoven anytime.

Otavio A. L. Lemos, Universidade de Sao Paulo   **Joel Ossher**, University of California, Irvine
Sushil Bajracharya, University of California, Irvine

Compile-Time Execution Structure of Object-Oriented Programs with Practical Ownership Domain Annotations

Flexible ownership domain annotations can express and enforce design intent related to encapsulation and communication in object-oriented programs. Those annotations also enable obtaining a sound visualization of a system's execution structure at compile time. The visualization provides design intent, is hierarchical, and thus more scalable than existing compile-time approaches.

Marwan Abi-Antoun, Carnegie Mellon University

Exploiting Code Search Engines to Improve Programmer Productivity

Code Search Engines (CSE) can be used for several applications like searching relevant code samples, identifying hotspots, and finding bugs. However, there are two major limitations: returned samples are too many and they are partial and not-compilable. Our framework addresses the preceding limitations and helps exploiting CSEs for those applications.

Suresh Thummalapenta, North Carolina State University

Automatic Support for Model-Driven Specialization of Object-Oriented Frameworks

An appealing strategy for supporting specialization of an object-oriented framework is to adopt a domain-specific modeling approach, where a domain metamodel and a code generator are manually developed to support model-driven framework specialization. Our research advocates that this support can be automated having an additional specialization layer in the framework.

André L. Santos, University of Lisbon

Activating Refactorings Faster

Refactoring tools promise to increase programmers' coding speed, but programmers report that contemporary tools sometimes slow them down. This poster presents pie menus and selection cues, two fast new mechanisms for activating refactoring tools. These mechanisms were designed to make good on the promise to help programmers code faster.

Emerson Murphy-Hill, Portland State University

Round-Trip Engineering Using Framework-Specific Modeling Languages

We propose Framework-Specific Modeling Languages (FSMLs) which are languages designed for areas of concern of object-oriented frameworks. A framework-specific model expressed using an FSML describes how an application built on top of a framework is using the framework. The mapping of an FSML enables automatic forward-, reverse-, and round-trip engineering.

Michal Antkiewicz, University of Waterloo

Software Speculative Multithreading for Java

We apply speculative multithreading to sequential Java programs in software, working to achieve speedup on existing multiprocessors. Our Java bytecode interpreter and forthcoming JIT compiler implementations share a common speculation library. Initial profiling indicates three main optimizations: adaptive return value prediction, online fork heuristics, and nested method level speculation.

Christopher J. F. Pickett, McGill University

Chair: Brian Goetz, Sun Microsystems

ooPSLA attracts a unique mix of academics and practitioners who are interested in a wide variety of topics relating to object-oriented programming techniques, languages, and development methodologies. Tutorials are the centerpiece of ooPSLA's educational program; ooPSLA tutorials are half-day classes designed to help software professionals rapidly come up to speed on a specific technology area or methodology. Tutorials cover the range from best practices in standard technologies and methodologies to new ideas that may change how we develop software in the coming years. And they are taught by the best in the business.

Here, you can find instructors who are passionate about their subject teaching courses on topics everyone should learn. You'll find "big-picture" classes on topics such as service oriented architectures, system architecture, modeling, security, and agile development methodologies, as well as nuts-and-bolts classes on Ruby, Python, mobile applications, and concurrent programming. With 53 tutorials to choose from, you're sure to find at least a few that interest you.

T1: Software Evolution: Analysis and Visualization	Sunday 08:30–12:00	512C

Harald Gall, University of Zurich

Harald Gall is professor of software engineering at the University of Zurich, Department of Informatics. Prior to that, he was associate professor at the Technical University of Vienna in the Distributed Systems Group (TUV), where he also got his PhD (Dr. techn.) and master's degree (Dipl.-Ing.) in Informatics. His research interests are in software engineering with focus on software evolution, software architectures, reengineering, program families, and distributed and mobile software engineering processes. He was program chair of ESEC-FSE 2005 (European Software Engineering Conference and ACM SIGSOFT Symposium on the Foundations of Software Engineering), IWPC 2005 (International Workshop on Program Comprehension, 90 participants), and IWPSE 2004 (International Workshop on Principles of Software Evolution, 60 participants).

Michele Lanza, University of Lugano

Michele Lanza is assistant professor at (and founding member of) the faculty of informatics of the University of Lugano in Switzerland. His main research interests lie in software reengineering, reverse engineering, and software evolution with a special focus on software visualization and metrics. He actively participated in the writing of a landmark book in reengineering, Object-Oriented Reengineering Patterns by Demeyer, Ducasse, and Nierstrasz. It describes a series of lightweight techniques for understanding, assessing and restructuring complex software systems. He has mainly experience in building information visualization tools and approaches to deal with reverse engineering and evolution: His Ph.D. thesis, titled Object-Oriented Reverse Engineering-Coarse-grained, Fine-grained, and Evolutionary Software Visualization discusses several visualizations to understand complex software systems at various levels. He received the Ernst Denert Software Engineering Award in 2003 for the thesis1. He is the developer of CodeCrawler, an information visualization tool which implements polymetric views, semantically enriched visualizations of information, e.g. software. Prof. Lanza is involved in the visualization and the reverse engineering communities by contributing to the main venues such as SoftVis, VISSOFT, WCRE, ICSM, TSE, etc.

Gaining higher level evolutionary information about large software systems is a key challenge in dealing with increasing complexity and decreasing software quality. Software repositories such as modifications, changes, or release information are rich sources for distinctive kinds of analyses: They reflect the reasons and effects of particular changes made to the software system over a certain period of time. If we can analyze these repositories in an effective way, we get a clearer picture of the status of the software. For example, software repositories can be analyzed to provide information about the problems concerning a particular feature or a set of features. Hidden dependencies of structurally unrelated but over time logically coupled files exhibit a high potential to illustrate software evolution and possible architectural deterioration. In this tutorial, we describe the investigation of software evolution by taking a step towards reflecting the analysis results against software quality attributes. Different kinds of analyses (from architecture to code) and their interpretation will be presented and discussed in relation to quality attributes. The tutorial will touch issues such as meta-models for evolution data, data analysis and history mining, software quality attributes, and visualization of analysis results.

Tutorials

Stuart Charlton, BEA Systems

Stuart Charlton is an Enterprise Architect with BEA Systems Worldwide Consulting. Stuart specializes in the areas of architecture, SOA, data warehousing, and is an avid student of lean & agile approaches to business processes and product development. Prior to joining BEA, he was the lead integration architect for a major Canadian telecommunications company, and has been a consultant and trainer for over a dozen organizations in the United States, Canada, and Japan. He is the co-author of CodeNotes for J2EE, published by Random House in 2002, and has written for leading online publications. Stuart resides in Toronto, Canada.

Mark Baker, Coactus

Mark is best known in industry as the leading proponent for so-called "REST style Web services", which leverage the architectural characteristics of the Web itself to provide a pervasive services infrastructure for machine to machine integration. He is a Principal at Coactus Consulting, specializing in large scale (including enterprise) integration using the Web. Mark resides in Ottawa, Canada.

The heart of the World Wide Web's technical success arguably lies in an architectural style that was designed from the beginning to enable properties to emerge from certain constraints, embodied in the HTTP, HTML, and URI specifications. This style, known as Representation State Transfer (REST), is the Web's stylistic guide to distributed object integration, networked services, application mash-ups, and cross-organizational information exchange. REST is increasingly being used as a guide for business integration through services-oriented architecture (SOA), an area traditionally has been tackled through Intelligent Networks, Message-Oriented Middleware (MOM), SOAP Web Services, or Object Request Brokers. This tutorial will explore the Web's architecture, discuss common areas of confusion, and provide practical guidance on how to apply the Web to real-world business and consumer application challenges. Topics for discussion include: What makes the web's architecture different from SOAP Web Services, or from classic distributed objects? How does hypermedia become the "engine" of a networked system's state? How can the use of the Web architecture improve the interoperability, scalability, and manageability of both internal and cross-enterprise service networks? How can one enable secure and reliable transactions on the Web?

Jonathan Aldrich, Carnegie Mellon University

Software analysis tools are revolutionizing the quality of the software delivered by today's industry leaders. Prominent software vendors use static analysis to find bugs, eliminate security holes, and deliver high-quality patches to deployed software in a timely fashion. These analysis tools excel at finding exactly the classes of errors that are most difficult to find through testing, inspections and other mainstream quality assurance practices. This tutorial begins by discussing the reasons why traditional quality assurance practices such as testing and inspections are no longer adequate to deliver the dependability that is required to compete in today's software marketplace. It describes the fundamental advantages of static analysis technology and how the weaknesses of previous QA practices are addressed. Participants will learn the core concepts used in static analysis, including abstraction, soundness, false positives, and issues of scalability and adoptability. The ideas will be reinforced through discussion and exercises simulating analysis. Finally, the tutorial will provide guidance on how companies can integrate analysis tools into a comprehensive quality assurance strategy. OOPSLA attendees interested in a more in-depth, hands-on examination of the FindBugs tool may also want to register for the tutorial "Using FindBugs in Anger."

Peter Schuh, Independent Consultant

Peter Schuh is the author of Integrating Agile Development in the Real World, a field guide for anyone whose prime interest is the development and timely delivery of useful and usable software. He has held virtually every position on a project team, including project manager, programmer, DBA, business analyst, technical writer, and account manager. Most recently he has managed IT projects in the financial, leasing, healthcare and e-commerce industries. He has eight years experience successfully folding agile practices and techniques into non-agile project environments.

Agile practices and techniques have proven effective for delivering high-quality software that meets and even exceeds customer expectations. Executing on an agile plan, however, can still be a tricky business. In many environments—regardless of their development methodology—teams still need to make and meet commitments several months out, adjust to changing requirements and scope, and deliver regular and reliable status reports to upper management. The class will detail and explore pragmatic techniques for analysis, estimation and planning on agile projects both within and outside the accepted bounds of many agile methodologies. Specific topics discussed will include: solution sheets, the living plan, and fixed cost pricing. On the execution side, the class will detail and explore pragmatic techniques for iteration planning, weekly execution, and day-by-day status tracking, both within and outside agile project environments. Specific topics discussed will include: iterative development, task cycles, and burn-up reporting. Participants will leave with practices and techniques that they can adapt and apply quickly to their own teams and projects.

T5: High Quality Software Architecture — Sunday 08:30–12:00 — 518A

Michael Stal, Siemens AG Corporate Research and Technology

Michael Stal is a Principal Engineer at Siemens Corporate Research and Technology where he and his team are responsible for research and customer projects. His main research areas include Software Architecture and Distributed Systems. Michael is co-author of the Pattern-Oriented Software Architecture series. He is also a frequest speaker in various conferences and author of several articles.

It is commonly accepted that software architecture is a fundamental issue when developing a software system. It also accepted that a software system should provide the best quality possible. There are at least two questions in this context: What makes a high quality software system? And which approach should a development team follow when designing the architecture of such a software system? Basically, the tutorial tries to answer those two questions. It shows potential pitfalls when developing software systems. It illustrates properties of a high quality software system. And it provides recommendations how a software architect should participate in a software engineering project. The tutorial will cover the role of patterns and best practices, but also illustrate new architectural concepts such as Model-Driven Software Development and how they fit tinto the software architecture context. The tutorial will also provide a special focus on the challenges of Software Product Line Engineering. Attendees will get an introduction into the state-of-the-art of software architecture and learn how to systematically devleop high quality software systems.

T6: Principles of Aspect-Oriented Design in Java and AspectJ — Sunday, 08:30–12:00 — 518B

Dean Wampler, Object Mentor, Inc.

Dean Wampler is a consultant and mentor at Object Mentor, Inc. He is the founder of the AOSD advocacy site, aspectprogramming.com, as well as the open-source design-by-contract tool for Java, Contract4J (contract4j.org), which uses AspectJ. Dean is currently studying AOSD in dynamic languages.

How do you build agile, robust, and maintainable aspect-oriented software? This tutorial teaches how to do this with pragmatic aspect-oriented design (AOD), with examples in Java and AspectJ. We start by defining aspect-oriented software development (AOSD) and the problems it solves. Next we review the typical problems that developers encounter using aspects, followed by strategies that address these problems. The majority of the tutorial then covers extensions to standard object-oriented principles and patterns, all of which support designing aspect software that meets the same "production-quality" standards we expect of object software. The tutorial concludes with a look at AOSD in dynamic languages and a look forward at the potential of AOSD to improve architectures and frameworks.

T7: Unit Test Patterns and Smells; Improving Test Code and Testability Through Refactoring — Sunday 08:30–12:00 — 518C

Gerard Meszaros, ClearStream Consulting

Gerard Meszaros is Chief Scientist at ClearStream Consulting, a Calgary-based consulting company specializing is agile development processes. Gerard built his first unit testing framework in 1996 and has been doing automated unit testing ever since. He is an expert in test automation patterns, refactoring of software and tests, and design for testability. Gerard has applied automated unit and acceptance testing on projects ranging from full-on eXtreme Programming to traditional waterfall development and technologies ranging from Java, Smalltalk and Ruby to PLSQL stored procedures and SAP's ABAP. He is the author of the book xUnit Test Patterns - Refactoring Test Code.

XUnit is the generic name given to the family of tools/frameworks used by developers when developing automated unit tests. The xUnit family includes JUnit and NUnit and has been ported to or reimplemented in almost every modern programming language from Ada and Access to XML and XSLT. The xUnit community has now had enough experience to start cataloging "best practices" and "not so best practices" as patterns and smells. This tutorial introduces a number of these "test smells", describes their root causes, and suggests possible solutions expressed in the form of patterns. The case study based presentation is interspersed with short hands-on exercises and discussion. The tests smells fall into three categories: code smells are recognized by programmers as they read or write tests; behavior smells are noticed while running tests, and project smells are recognized by team leads or project managers. The patterns range from high level goals and principles, strategy-level patterns, test design patterns, design-for-testability patterns and language-specific test coding idioms. While the focus is on unit testing, many of the patterns also apply to acceptance (story or functional) testing and to executable specification (e.g. behavior-driven tools such as RSpec and JBehave.)

Tutorials

T8: From Use to User Interface: Collaboratively designing, prototyping, and testing user interface to help your users succeed — Sunday 13:30–17:00 — 512D

Jeff Patton, ThoughtWorks

Jeff Patton has designed and developed software for the past 12 years on a wide variety of projects from on-line aircraft parts ordering to electronic medical records. Jeff has focused on Agile approaches since working on an early XP team in 2000. In particular Jeff has focused on the application of user centered design techniques to improve Agile requirements, planning, and products. Some of his recent writing on the subject can be found at www.agileproductdesign. com and in Alistair Cockburn's Crystal Clear. Jeff's currently a proud employee of ThoughtWorks, and a columnist with StickyMinds.com and IEEE Software.

You may be using use cases, user stories, user scenarios, workflow models, or a variety of other approaches to work through how your software could be used. Up till now in the requirements process you've diligently kept user interface details out. But, now it's time to actually start estimating and building software. The developers on the team would like to know what the software looks like and how it behaves. As a matter of fact, so would you. In this tutorial you'll learn a simple approach to quickly and predictably move from a simple use case to a user interface you feel confident in. You'll work inside a team to collaboratively design and paper prototype your user interface. You'll validate your UI actually helps your users reach their goals by performing lightweight usability testing. And since UI is seldom right the first time, you'll learn how to iteratively improve your design. Along the way you'll learn the basics if user centered design and an alternative way to write use cases that are easy to read and convert to effective user interface design.

T9: Pick Up Your Pen! — Sunday 13:30–17:00 — 513B

Steve Metsker, Dominion Digital

Steve Metsker is the author of many articles and four books on object-oriented programming techniques. He is a frequent speaker at user groups and particularly likes helping people with their writing.

This hands-on session will help get you past the urge to write and on to successfully sharing your ideas through nonfiction print. We'll look at how to find time to write, how to brainstorm, and how to unblock. We'll do short exercises as we go. Some of the exercises require a computer, so please bring a laptop if you can! We will work in pairs, so if at least half of use have laptops, we'll be fine. After covering the basics of getting started with writing, we'll dive into writing itself: The importance of understanding thy premise; The mechanics of outlining; The challenge of smoothing an outline into prose. Of course, we'll do exercises along the way. If you have the urge to write and often feel like you should write more or you'll write more someday, today's the day to sign up for this tutorial and get writing!

T10: Object-Oriented Models of Requirements and High-level Software Design — Sunday 13:30–17:00 — 513F

Hermann Kaindl, Vienna University of Technology, ICT

Hermann Kaindl joined the Institute of Computer Technology at the Vienna University of Technology in early 2003. Prior to moving to academia as a full professor, he was a senior consultant with the division of program and systems engineering at Siemens AG Austria. There he has gained more than 24 years of industrial experience in software development. He is a Senior Member of the IEEE, a member of the ACM and INCOSE, and is on the executive board of the Austrian Society for Artificial Intelligence. Hermann Kaindl has previously held tutorials at CAiSE-00, RE-01, RE-02, HICSS-36, INCOSE-03, RE-03, CADUI-IUI-04, INCOSE-04, RE-04, HICSS-38, IRMA-05, INCOSE-05, AAAI-06, HCI-06, OOPSLA-06 and HICSS-40.

- How can the application domain and the requirements be better understood using object-oriented (OO) modeling? - How do scenarios / use cases fit together with functional requirements?

 - How can a model of the domain be used for a transition to a design model? This tutorial addresses these questions in the following manner. It shows how each requirement given in natural language can be viewed itself as an object and modeled as such. This approach facilitates both a hierarchical organization (by grouping requirements instances into classes and subclasses) and explicit association (by relating requirements through OO associations). While scenarios / use cases can somehow illustrate the overall functionality, additionally functional requirements for the system to be built should be formulated and related to them appropriately. All kinds of requirements make statements about the application domain, which should be first represented in a domain model of conceptual classes, in order to make the requirements better understandable. This tutorial explains a seamless transition to a high-level design model, where design classes are abstractions of implementation classes. The use of object-oriented models from a domain model and requirements to design will be illustrated with a running example each for presentation and for group exercises.

T11: Best Practices for Model-Driven Development   Sunday 13:30–17:00   514B

Markus Voelter, Independent Consultant

Markus Voelter works as an independent consultant and coach for software technology and engineering. He focuses on software architecture, middleware as well as model-driven software development. Over the last years, Markus has worked extensively in the area of model-driven software development in all kinds of projects - from embedded to enterprise, from small to large. Markus is the author of several magazine articles, patterns and books on middleware and model-driven software development (www.mdsd-book.org). He is a regular speaker at conferences world wide. You can find more information at http://www.voelter.de

Model-Driven Development (MDD) is well on its way into the mainstream of software development. Many developers have used Domain-Specific Languages (DSLs), models and code generators to simplify recurring development tasks. However, fully exploiting the power of DSLs and MDD requires more than just using a generator. Many of the benefits only materialize if development teams build their own languages and generators for their particular domain. This tutorial presents a set of best practices, including: * When and how to use custom meta models instead of general purpose meta-models such as the Unified Modeling Language (UML) * Various techniques of annotating, extending and specializing models * Tradeoffs between textual and graphical DSLs * When to use code generation, and when to use interpreters * Pragmatic use of model-to-model transformations * Handling variants in models, generators and transformations * How to sensibly integrate generated code with manually written code * Multiple viewpoint meta models * Modeling behavioral aspects of a system This tutorial is aimed at real developers working in real projects. The concepts will be illustrated with examples from my projects in the last year, based on tools from the Eclipse Modeling Project and openArchitectureWare.

T12: Software Architecture Refactoring   Sunday 13:30–17:00   518A

Michael Stal, Siemens AG Corporate Research and Technology

*Michael Stal is a…***See T5**.

Refactoring is an excellent practice for bottom-up and structure-preserving gardening activities in implementations. In an agile setting, refactoring denotes an essential tool, because the incremental integration of new use-cases keeps the tactical architecture aspects in flux which means that implementation artifacts require continuous modifications. Joshua Kerievsky came up with "Refactoring To Patterns" which identifies places in an implementation that use proprietary solutions for problems which should be better solved using patterns. If thinking that even further, we enter the world of architecture refactoring. In architecture refactoring we refactor the architecture itself when it reveals some problems ("smells") or needs to be changed for some reason such as opening it up for extension. The tutorial motivates and introduces the area of architectural factoring. It shows how architectural refactoring relates to architecture design and how and when it should be applied in a software engineering context. This includes the introduction of architecture qualities and patterns that build the foundation of high quality software architecture. The tutorial illustrates some typical examples for architecture refactoring such as breaking cycles, opening for extensions, or introducing strict layering. It also motivates how architecture refactoring goes hand in hand with code refactoring. At the end, an attendee will have an in-depth overview of architecture refactoring and how this relates to other software engineering activities.

T13: LINQ From the Source   Sunday 13:30–17:00   510A

Erik Meijer, Microsoft Corp

Erik Meijer is an architect in the SQL Server group at Microsoft. He is currently working with the C# and Visual Basic teams on LINQ, that is, language and type-system support for bridging the worlds of object-oriented (CLR), relational (SQL), and hierarchical (XML) data. Before joining Microsoft Erik was an academic language researcher, where he worked on functional languages such as Haskell98 and Mondrian.

Ted Neward, Neward & Associates

Ted Neward is the principal at Neward & Associates, a consulting firm specializing in enterprise consulting and technology research, focusing on Java, .NET, programming languages, virtual machines and XML services. He has worked closely with Microsoft, is a Java Community Process Expert Group member, a BEA Technical Director, Microsoft MVP Architect, and Pluralsight instructor.

Language Integrated Query (LINQ) is one of the hot technologies in the next version of Visual Studio. In this session, hosted by one of the designers of VB9, C# 3.0, LINQ to Objects, LINQ to XML, and LINQ to SQL, we will drill down into all the gory technical details behind LINQ, as well as its practical applicability to modern development projects, from large-scale enterprise systems to "tiny webapps on a big database" projects. This is a unique opportunity to get a first hand account of this new technology, the design process, and historical context.

Tutorials

T29: Using FindBugs in Anger — Sunday 13:30–17:00 — 518C

David Hovemeyer, York College of Pennsylvania

David Hovemeyer is an Assistant Professor of Computer Science at York College of Pennsylvania. FindBugs was a product of his Ph.D. research on static program analysis to find bugs at the University of Maryland. He continues to be actively involved in the FindBugs project, focusing on implementation of the underlying analyses.

William Pugh, Univ. of Maryland

William Pugh received a Ph.D. in Computer Science (with in minor in Acting) from Cornell University. He is currently a professor at the University of Maryland, College Park. Prof Pugh's current research focus is on developing tools to improve software productivity, reliability and education. Current research projects include FindBugs, a static analysis tool for Java, and Marmoset, an innovative framework for improving the learning and feedback cycle for student programming projects.

FindBugs is a open source static analysis tool that looks for defects in Java programs. This tutorial is designed to help attendees fully incorporate FindBugs into their software development process, with an emphasis on solving the issues that arise in using static analysis on projects with a large code base and multiple developers. We will go into substantial detail about how to perform differential analysis, which allows you to see which warnings have been introduced since the previous build, or since the last release to customers. We will also discuss persistent auditing, so that after reviewing a set of warnings and deciding which need to be fixed and which can be ignored, those audit results are applied to the results of each new analysis, so that you don't have to reexamine warnings already audited. We will also discuss how to apply annotations, including those defined by JSR-305, to allow FindBugs to better understand your own code and libraries, thus finding important bugs and suppressing unimportant issues. Finally, we will also discuss how to write your own bug detectors.

T14: Security Patterns and Secure Software Architecture — Monday 08:30–12:00 — 512F

Munawar Hafiz, University of Illinois at Urbana-Champaign

Munawar Hafiz is a graduate student at University of Illinois working with Professor Ralph Johnson. He has been working on security patterns for the last four years. His pattern documentation project at patternshare has been funded by Microsoft's Patterns and Practices group. He is currently working on a security pattern organization scheme.

Security patterns are valuable in all stages of secure software development because they document proven solutions of security problems and act as a vocabulary for exchanging ideas between security architects, developers and managers. The patterns solve problems of different granularity and their context ranges from business architecture to implementation-specific issues. Three books on security patterns books have recently been published, each with a fresh perspective. This tutorial provides the audience with a comprehensive look at the documented security patterns, covering about seventy security patterns. This tutorial also touches on the classification of security patterns, leading to that efficient pattern navigation. Through interactive demonstrations and practical examples of software architecture, this tutorial exposes the audience to broad ranges of security problems and their solutions.

T15: Java API Design — Monday 08:30–12:00 — 513A

Boris Bokowski, IBM

Boris Bokowski is a Software Developer with IBM Rational Software in Ottawa, Canada and a full-time Eclipse committer working on the Platform/UI team. He is part of the "API police" for the Eclipse Platform. Boris has over ten years of Java experience and has worked in many different domains like client-server protocols, compiler implementation, web application servers, and graphical user interfaces. He holds a PhD in computer science from Freie Universität Berlin, Germany.

Good and dependable APIs are critical to the long-term success of any larger software project. This tutorial is based on our experience with developing and evolving the Eclipse platform API over the past several years. It presents best practices for API specification, API evolution, and the API design process. In particular, it gives concrete technical advice on how to design and specify Java APIs, and how to evolve APIs in a way that maintains compatibility. The tutorial covers source compatibility and binary compatibility of changes to interfaces, classes, generics, and other language features up to Java 5. Stories from the Eclipse API trenches and the API compatibility quiz will engage participants and make this complex topic accessible.

T16: Test-Driven Development—Hands-on!   Monday 08:30–12:00   513B

Niclas Nilsson, factor10

Niclas is a software developer, consultant, educator, and writer, with a deep passion for the software development craft. He started working as a developer in 1992 and, drawing from experience, he knows that some choices makes significant difference in software development, like languages, tools and processes. This is the reason behind his affection for dynamic languages, test-driven development, code generation, and agile processes. He has been test-driven for over five years and has so far used test-driven development in Java, C++ and Ruby. Blog at http://niclasnilsson.se

Jimmy Nilsson, factor10

Jimmy Nilsson has written numerous technical articles and two books. He is training and speaking at conferences, but above everything else, he is a developer with almost 20 years of experience. He has been working with .NET since the preview version and has been a TDD practitioner and evangelist for years. Blog at: http://jimmynilsson.com/blog/

Curious about Test-Driven Development? Heard a lot about it, but never had a chance to really try it? This tutorial will give you hands-on experience with Test-Driven Development in Java and/ or C#. Bring your laptop with either a new version of Eclipse or Visual Studio (2003 or 2005) installed and you'll get a real Test-Driven kick-start. We will provide you with the additional tools/ frameworks you'll need. You will get a feeling for what it's like to make changes to a system that has a lot of tests (no more "this looks ugly, but I don't have the guts to change it"). You'll also experience how writing test code before writing the production code changes the way you think about and design your production code. The tutorial will introduce you to Test-Driven Development, refactoring, unit-testing frameworks, and mock frameworks. You'll have the opportunity to try these for yourself in C# or Java, with the presenters helping out and answering your questions.

T18: Software Architecture Documentation in the Real World   Monday 08:30–12:00   518A

Markus Voelter, Independent Consultant

Markus Voelter works as… **See T11**.

Michael Kircher, Siemens Corporate Technology

Michael Kircher is working as Senior Engineer at Siemens Corporate Technology. His main focus is on software architecture, distributed object computing, software product-line engineering, and patterns. In the role as software architect, reviewer, and consultant he led and supported a multitude of development projects within Siemens AG. He is author of several articles, papers, and books on the above mentioned topics.

Documentation is a relatively unpopular aspect of software development and its importance is much debated. However, if you want to develop software architectures that should last for a long time - e.g. for platforms or product lines - documenting these architectures is essential. And done right, it can even be rewarding and fun! In this tutorial we will introduce best practices for documenting software architectures in real projects. These best practices include * Using the pattern form as a vehicle to write compelling explanations and capturing rationales * Using pattern languages to describe complex, multi-aspect architectures * Using tutorials and walk-throughs to describe the programming model to application developers * Using Structured Glossaries as a way of defining the meaning of terms and the relationships between terms * Using architectural diagrams (Visio/Powerpoint, UML Class Diagrams, Sequence Diagrams) to illustrate architectures * Using architecture meta models for formally describe software architectures We will also talk a little bit about which channel (e.g. paper, online, video) is best suited for the various kinds of documentation, as well as about some best practices for typesetting and document layout. The toolbox provided by this session makes architecture documentation much more productive.

Tutorials

David Holmes, Sun Microsystems Australia

David Holmes is a Senior Java Technologist at Sun Microsystems, working in the JVM Real-time group. His work with Java technology has focused on concurrency and synchronization support in the language and virtual machine and he has previously worked on a real-time Java virtual machine. David was a member of the JCP Expert Group for JSR-166 "Concurrency Utilities", that shipped in the Java 5.0 release. He has presented tutorials on concurrent Java programming and design at numerous international object-oriented programming conferences over the past ten years. David is a contributing author to "Java Concurrency in Practice", written by Brian Goetz and the other JSR-166 EG members. Along with Ken Arnold and James Gosling, David is also a co-author of the book "The Java Programming Language" - Third and Fourth Editions. David completed his Ph.D. at Macquarie University, Sydney, in 1999, in the area of synchronization within object-oriented systems.

Joe Bowbeer, Mobile App Consulting

Joe Bowbeer is a Java Micro Edition specialist whose fascination with concurrent programming began in his days at Apollo Computer. He was a member of the JCP Expert Group for JSR-166 "Concurrency Utilities", that shipped in the Java 5.0 release and is a contributing author to "Java Concurrency in Practice", written by Brian Goetz and the other JSR-166 EG members. Joe is a veteran OOPSLA attendee and has co-presented concurrency talks at both OOPSLA and JavaOne.

Come learn the fundamentals of concurrent programming in the Java programming language. Historically, concurrent programming has mostly been the domain of systems programmers rather than application developers. However, Java's built-in support for concurrency has turned many application developers, willingly or not, into concurrent programmers. But concurrent programming poses many traps for the unwary, and analyzing, testing, and debugging concurrent programs can be significantly more complicated than with sequential ones. This tutorial introduces the basic concept's of multithreaded concurrent programming in Java. It describes Java's threading model and core API's, language features for providing thread-safety through mutual exclusion, and the primitive API's from which thread communication patterns can be implemented. It describes concepts and mechanisms for building thread-safe classes, and introduces simple design patterns and techniques for constructing concurrent applications in Java. Throughout there is a focus on resolving the competing design forces (safety, liveness, performance, reusability) that are present in concurrent software design problems.

Martin Odersky, EPFL

Martin Odersky is a professor at EPFL in Lausanne, Switzerland. His research interests focus on programming languages, more specifically languages for object-oriented and functional programming. His research thesis is that the two paradigms are just two sides of the same coin and should be unified as much as possible. To prove this he has experimented with a number of language designs, from Pizza to GJ to Functional Nets. He has also influenced the development of Java as a co-designer of Java generics and as the original author of the current javac reference compiler. His current work centers around the Scala programming language, which unifies FP and OOP while staying completely interoperable with Java and .NET.

Ted Neward, …**See T13**.
Gilles Dubochet, EPFL

Gilles Dubochet is a graduate student in Prof. Martin Odersky's programming methods laboratory. He graduate from EPFL in 2005 after completing his Master project at the university of Edinburgh under the supervision of Prof. Philip Wadler. His current interests include staged compilation, meta-programming and, more generally, compiler design.

Scala is a new general-purpose programming language that is fully interoperable with Java. It smoothly integrates features of object-oriented and functional languages. Scala allows a terseness of style comparable to scripting languages but has at the same time a strong and static type system. In this way, it can express common programming patterns in a concise, elegant, and type-safe way. Scala also allows for creation of Domain-Specific Languages that hide much of the complexity of traditional APIs from users. Combined with its ability to plug into existing Java environments, this makes it possible to construct higher-level interfaces to existing Java libraries (servlets, Spring beans, EJB, Swing, and more). Its Actors-based threading approach, combined with its functional preference for immutable values makes concurrent programming far simpler than what's present in Java. And its built-in XML support makes Scala natural for XML processing tasks, including Web Services. This tutorial will give an introduction to the Scala programming language, highlighting its main innovative features: closures, pattern matching, type abstraction, and mixins.

T22: Building Embedded and Stand-alone Domain Specific Languages: Principles and Practise
Monday 13:30–17:00 **512F**

Eric Van Wyk, University of Minnesota

Eric Van Wyk is an assistant professor at the University of Minnesota and is a McKnight Land-Grant professor. His research interests include techniques for the declarative specification and implementation of languages and language extensions.

This tutorial provides an introduction to the principles and practice of building domain-specific languages (DSLs). We consider DSLs as stand-alone languages and as language fragments that can be embedded into general purpose languages such as Java. The tutorial covers the fundamental aspects of any DSL design: concrete syntax, abstract syntax, semantics, and code generation. It discusses in more detail techniques for generating scanners and parsers from specifications. A significant portion of the tutorial covers embedded DSLs and a tool, ableJ, that allows programmers to import DSLs as language extensions into an extensible specification of Java. For example, one extension embeds the SQL database query language into Java and statically checks for syntax and type errors in SQL queries; another adds types for physical units such as meters, feet/second, etc and type rules to ensure that values of such types are not used incorrectly. Finally, the tutorial highlights the distinguishing characteristics of many open source tools that one may use to build DSLs and considers visual DSLs that can be constructed using the Eclipse MDD tools and the Microsoft DSL Toolkit. Further information is available at www.melt.cs.umn.edu.

T23: Using Java ME to Program Your Mobile Phone
Monday 13:30–17:00 **513A**

Jonathan Knudsen, Sun Microsystems, Inc.

Jonathan Knudsen is the author of half dozen books and over one hundred articles on Java technology and Lego robotics. He has been working with Java technology since late 1996. http://kickbutt.jonathanknudsen.com/ http://jonathanknudsen.com/

This tutorial provides an overview of the world of Java ME and provides a swift course in the basics of Java ME application programming. Participants will learn how to create an application, how to build a user interface, how to connect to the Internet, and more. Based on material from the book _Kicking Butt with MIDP and MSA_, this tutorial will encourage participants to learn Java ME programming in a hands-on spirit of fun. Participants should have a recent version of Netbeans and Netbeans Mobility Pack for CLDC already installed on their laptops in order to build and test mobile applications. Get Netbeans here: http://www.netbeans.org/ Get Mobility Pack here: http://www.netbeans.org/products/mobility/

T24: The Art of Telling Your Design Story
Monday 13:30–17:00 **513B**

Rebecca Wirfs-Brock, Wirfs-Brock Associates

Rebecca Wirfs-Brock, president of Wirfs-Brock Associates and IEEE Software's Design Columnist, is a well-known and respected object practitioner. She invented the way of thinking about objects known as Responsibility-Driven Design and is the lead author of Object Design: Roles, Responsibilities, and Collaborations (2003) and the classic Designing Object-Oriented Software (1990). Through her writing, teaching, consulting, and speaking she popularizes the use of informal techniques and thinking tools for designers and analysts.

Do you have trouble communicating design ideas, getting buy-in to new approaches, or informing others about your design? The best way to present your design isn't the same way you came up with it. As Francis Galton, a 19th century geneticist remarked, "It often happens that after being hard at work, and having arrived at results that are perfectly clear and satisfactory to myself, when I try to express them I feel that I must begin by putting myself upon quite another intellectual place. I have to translate my thoughts into a language that does not run very evenly with them." This tutorial presents tips, techniques, and guidelines for communicating your designs to others. To be an effective communicator, you need to know what belongs together and what deserves special emphasis. By choosing what to emphasize, understanding what's fundamental, and using progressive realization techniques, you can unfold a design in successively interesting parts. In this tutorial we'll present options for drawing and explaining your design using informal as well as formal notation. We'll demonstrate some fun ways to get people to simulate object interactions by tossing koosh balls, and we'll have time to plot out your own design's storyline.

Tutorials

T25: Integrating Architecture-Centric Methods into Object-Oriented Analysis and Design   Monday 13:30–17:00   513C

Raghvinder Sangwan, Pennsylvania State University

Raghvinder S. Sangwan, Assistant Professor of Information Science, holds a Ph.D. in Computer and Information Sciences from Temple University. He teaches analysis, design, and development of software systems, their architecture, and automatic and semi-automatic approaches to assessment of their design and code quality. Prior to joining Penn State University, he worked as a lead architect for Siemens on geographically distributed development projects, building information systems for integrated health networks. He is a technical consultant for Siemens Corporate Research investigating approaches to managing global software development projects. Dr. Sangwan's research includes requirements engineering, software design, software architecture and product line engineering.

Architecture has been established as a key to developing software systems that meet quality expectations of their stakeholders. Object-Oriented Analysis and Design (OOAD) methodologies, however, treat architecture only indirectly or implicitly. The quality of systems developed using such methodologies, thus, depends largely on the skill level and experience of its architect. It has been suggested, therefore, that augmenting these methodologies with software architecture-centric methods such as the Quality Attribute Workshop (QAW) and Attribute Driven Design (ADD) can provide explicit and methodical guidance to an architect in creating systems with desirable qualities. In this tutorial, we will first go through the exercise of applying OOAD techniques (use case analysis, domain modeling, and component-based design) for creating the architecture for a system from the building automation domain. We then use the techniques prescribed by the architecture-centric methods (quality attribute workshop and attribute driven design) for creating the architecture for the same system. These two exercises are used to clearly demonstrate the shortcomings of OOAD. We finally demonstrate how OOAD can be augmented with architecture-centric methods to overcome these shortcomings.

T26: Why Users Say, "Start with the Screen!": Effective Test-Driven Development, Presentation Layer-First   Monday 13:30–17:00   513D

Bobby Norton, ThoughtWorks

Bobby Norton first started programming BASIC on an Apple IIE at age 9, and has since designed and developed software in C, C++, Java, and C# working in several domains for clients such as the FBI, NASA, GE, the US military. His desire to take a more active role in the agile methods community prompted him to join the ThoughtWorks Toronto office last year as a senior consultant and developer. He has appeared as a speaker at several invited talks, most recently last year's Agile Vancouver, and serves as a trainer in Object Bootcamp sessions offered by ThoughtWorks.

Chris Stevenson, ThoughtWorks

Chris Stevenson has 14 years of development experience using Java, .NET, and Ruby. He is an experienced agile coach and leader of small teams, particularly those implementing Extreme Programming. Chris has worked with teams literally all over the world: Australia, the UK, India, the U.S., and most recently, Canada, where he serves as a senior consultant at ThoughtWorks. Chris is an experienced trainer and presenter, but he is best known in the open source world for his work on jTDS and AgileDox, CruiseControl.NET, PicoContainer, NanoContainer, Groovy, NMock and JMock. His maintains a blog at skizz.biz.

Your user stories, use cases, or other requirements artifacts define high-level interactions with the interface. The team has developed a user interface prototype that the users love. Perhaps you even have automated acceptance tests to define success criteria for the features. It's time to start designing and implementing the feature. Let's assume you're using test-driven development (TDD). What's the first test to write? Should we start at the data access layer and build up? Maybe the domain model and build out? In this tutorial, participants will learn through development examples and discussion the tools and techniques to start design and implementation at the presentation layer, and then systematically work down to service, domain model, and data access layers. We'll then produce a distribution ready for deployment using automated acceptance testing and continuous integration. The resulting code is loosely coupled, self-documenting, highly-tested, and reflective of the needs and domain of the users. Through end-to-end examples in C# and Java, participants will gain experience with: Using presentation patterns such as Supervising Controller, Passive View, and the venerable Model-View-Controller; writing interaction-based tests using mock objects; techniques for effective test-driven design of layered architectures; automating unit and acceptance tests.

T27: SOA in Reality — Monday 13:30–17:00 — 518A

Nicolai Josuttis, self employed

Nicolai Josuttis (www.josuttis.com, nj@it-communication.com) is an independent system architect, technical manager, author, and consultant. He designs mid-sized and large software systems for the telecommunication, traffic, finance, and manufacturing industries. He is well known both in the C++ Community and to attendees at various conferences. He not only speaks and writes with authority (being the author of 'SOA in Practice', 'The C++ Standard Library' and 'C++ Templates') but is also an innovative presenter. He is a partner of IT-communication. com. Currently, he is a team lead for the realization of a SOA at an international mobile phone company running with Millions of service calls per day.

SOA (Service Oriented Architecture) is hype. According to Gartner, by 2008, SOA will provide the basis for 80% of development projects. However, we see all the effects of a hype of a very new topic. There is not even a common understanding about the core terms such as "SOA" and "service". In this tutorial, Nicolai Josuttis tries to separate the SOA hype from reality. As a team leader of the realization of a SOA at a world-wide mobile phone company with a heterogeneous infrastructure, about 30 service participants, and about 300 services in production, he knows about the big difference between was is being said and promised about SOA and what it means in practice to run a SOA in a large and heterogeneous environment. Starting with a short clarification of the fundamental SOA concepts, he will present the most important steps and traps of bringing SOA into production. Facts and knowledge based on a broad understanding of large and distributed systems spiced with important insights based on his experience. Note that this tutorial covers the concepts and isn't just a tutorial about WebServices (although there is a lot to say about WebSerives as a preferred SOA implementation strategy).

T28: Java Concurrency Utilities in Practice — Monday 13:30–17:00 — 518B

David Holmes, *Sun Microsystems Australia*

*David Holmes is a…***See T19 above**.

Joe Bowbeer, Mobile App Consulting

*Joe Bowbeer is a…***See T19 above**

Find out how to apply the java.util.concurrent utilities to solving concurrency problems in your applications. The Java programming language has turned a generation of applications programmers into concurrent programmers through its direct support of multithreading. However, the Java concurrency primitives are just that: primitive. They can be used to build complex concurrent classes and applications, but doing so takes great care as concurrent programming poses many traps for the unwary. Designing concurrent programs that can successfully resolve the competing forces of safety, liveness, efficiency, coordination, and reusability, is a task that can be greatly aided through the use of a good concurrency utility library. This tutorial gives an overview of the Java concurrency utilities and through a series of applied examples shows in detail how these utilities can be used to effectively build and manage concurrent applications. It demonstrates design patterns for effective thread creation and communication using the Executor framework and Futures; shows how to use concurrent data structures and atomic objects to achieve thread-safety; and how to coordinate concurrent activities through the use of "synchronizers" like latches and barriers.

T40: Software Security: Building Security In — Monday 13:30–17:00 — 518C

Gary Mcgraw, Cigital

Gary McGraw is the CTO of Cigital, Inc., a software security and quality consulting firm with headquarters in the Washington, D.C. area. He is a globally recognized authority on software security and the author of six best selling books on this topic. The latest, Software Security: Building Security In was released in 2006, with Exploiting Online Games slated for release this year. His other titles include Java Security, Building Secure Software, and Exploiting Software; and he is editor of the Addison-Wesley Software Security series. Dr. McGraw has also written over 90 peer-reviewed scientific publications, authors a monthly security column for darkreading.com, and is frequently quoted in the press. Besides serving as a strategic counselor for top business and IT executives, Gary is on the Advisory Boards of Fortify Software and Raven White. His dual PhD is in Cognitive Science and Computer Science from Indiana University where he serves on the Dean's Advisory Council for the School of Informatics. Gary is an IEEE Computer Society Board of Governors member and produces the monthly Silver Bullet Security Podcast for IEEE Security & Privacy magazine.

Software security has come a long way in the last few years, but we've really only just begun. In this tutorial, I will present a detailed approach to getting past theory and putting software security into practice. The three pillars of software security are applied risk management, software security best practices (which I call touchpoints), and knowledge. By describing a manageably small set of touchpoints based around the software artifacts that you already produce, I avoid religious warfare over process and get on with the business of software security. That means you can adopt the touchpoints without radically changing the way you work. The touchpoints I will describe include **o** Code review using static analysis tools **o** Architectural risk analysis **o** Penetration testing **o** Security testing **o** Abuse case development **o** Security requirements Like the yin and the yang, software security requires a careful balance-attack and defense, exploiting and designing, breaking and building-bound into a coherent package. Create your own Security Development Lifecycle by enhancing your existing software development lifecycle with the touchpoints.

Tutorials

T30: Incremental Releases Users & Stakeholders Will Love — Tuesday 08:30–12:00 — 515C

Jeff Patton, ThoughtWorks

*Jeff Patton has…***See T8**.

One of the benefits of Agile Software Development is "early and continuous delivery of valuable software." Dividing development work up into small pieces (user stories or backlog items) then building the most valuable parts first sounds like a simple idea, but there's often a bit more to it. Sometimes it takes a few not-so-valuable parts to allow users to take advantage of the most valuable parts. At times the most valuable parts may show well, but without a sufficient amount of software implemented, users may be unwilling to set aside legacy software or even manual processes to actually put the new software into use and earn the return on investment incremental release should bring. In this tutorial participants will learn the basics of planning incremental releases that are useful to their users. We'll discuss strategies for splitting user stories into the small but useful parts that allow releases to contain more user stories. You'll learn more about the "myth of the finished user story" and why, if you're not cautious, your stories may inflate while you're not looking.

T31: Green Bar for C++ - Unit Testing and Refactoring C++ — Tuesday 13:30–17:00 — 512C

Peter Sommerlad, IFS Institute for Software at HSR Rapperswil

Peter Sommerlad is professor for software engineering and head of Institute for Software at HSR Hochschule für Technik, Rapperswil. Peter is co-author of the books Pattern-oriented Software Architecture Vol.1 and Security Patterns as well as author and shepherd of many other patterns for software and security. His current research interests are in refactoring for non-java languages in Eclipse (e.g. C++, Ruby, Python) with his long-term goal of making software simpler by decremental development: Refactoring software to 10% its size with better architecture, testability and quality and even improved functionality.

Automated unit testing supports high quality of program code, even under inevitable change and refactoring. As a side effect, unit tested code often has a better structure. Java developers are used to unit testing because of JUnit and its tight integration into IDEs. C++ programmers lacked the tool support for easy-to-use unit testing, even though the language's complexity asks for it. Refactoring and simplifying C++ code without tests is hazardous. This tutorial introduces C/C++ programmers to unit testing and teaches them to use the CUTE unit testing framework. Specialties of using C++ in contrast to Java are explained, both that are advantageous to unit testing and refactoring, as well as those that are worse than with Java. The job of retro-fitting automated tests to an existing code base is also addressed. CUTE can be used with a corresponding Eclipse CDT plug-in (better) or with MS Visual Studio with support for automatic navigation to failing test cases. Any other standard C++ compiler should be fine as well. In addition C++ automated refactoring is taught and exercised based on Eclipse CDT's refactoring extensions implemented by IFS Institute for Software. Detailed instructions on how to prepare your own laptop will be given at http://wiki.hsr.ch/PeterSommerlad/wiki.cgi?OopslaCuteTutorial

T32: Writing Adaptable Software: Mechanisms for Implementing Variabilities in Code and Models — Tuesday 13:30–17:00 — 514C

Markus Voelter, Independent Consultant

*Markus Voelter works as…***See T11**.

Today, a software system is often a member of a system family. Members have many things in common and differ in a set of well-defined respects. These differences are called variabilities and must be implemented in the software system. This tutorial provides an overview over techniques available for implementing such variabilities. Various alternatives exist in programming languages such as preprocessors (as in C/C++), static metaprogramming (e.g. C++ templates), aspect-oriented programming (e.g. AspectJ and CaesarJ), polymorphism and design patterns (such as Bridge, Strategy or Interceptor), or reflection and dynamic metaprogramming (as in Ruby). In addition, variabilities can also be handled in models in the context of model-driven development, for example by connecting structural models with varability models, model weaving and AO techniques for model-to-model and model-to-code transformations. The first part of the tutorial is a discussion of the different kinds of variability as well as notations for describing each form (feature modelling, graph-based models). We will then show examples for each of the implementation alternatives listed above and discuss which of them are capable of handling structural and/or non-structural variabilities. Finally, we will elaborate on the tradeoffs with regards to performance, flexibility and complexity of each alternative.

T33: Software Best-Practices: Agile Deconstructed
Rigid Methods Reconstructed — Tuesday 13:30–17:00 — 515B

Steven Fraser, Cisco Research Center

Steven Fraser recently joined Cisco Research in San Jose, California, as a Director (Engineering). Previously, Steven was a member of Qualcomm's Learning Center in San Diego, California. Steven also held a variety of technology management roles at Bell-Northern Research, NT, and Nortel including: Process Architect, Senior Manager (Disruptive Technology and Global External Research), and Design Process Engineering Advisor. In 1994 he spent a year as a Visiting Scientist at the Software Engineering Institute (SEI) at Carnegie Mellon University (CMU) collaborating with the Application of Software Models project on the development of team-based domain analysis (software reuse) techniques. Fraser was the General Chair for XP2006 and is the Corporate Support Chair for OOPSLA'07 and Tutorial Chair for both XP2008 and ICSE 2009. Fraser holds a doctorate in EE from McGill University in Montréal—and is a member of the ACM and a senior member of the IEEE.

Part I: Software Best Practices: Agile Deconstructed

Part I of the tutorial is an introduction to software best practices including: coding standards, continuous builds, project management, retrospectives, test automation, risk assessments, and requirements engineering. While many of these best practices have been incorporated into agile methodologies, this presentation does not promote a specific agile methodology. The intended audience for the session includes software developers and managers evaluating and applying software practices to the development of large systems.

David Lorge Parnas, University of Limerick

*David Lorge Parnas is…***See Page 17**.

Part II: Rigid Methods Reconstructed: The Real Software Best Practices

If you want agile software, you have to apply highly disciplined (rigid) methods. This statement, which appears to be nonsense, recognizes that flexibility is not an accident. It requires careful analysis of changes and careful structuring of the software. Part II of the tutorial tells you what you must do.

T34: The WS-* Jungle — Tuesday 13:30–17:00 — 515C

Michael Stal, Siemens AG Corporate Research and Technology

*Michael Stal is a…***See T5**.

Jörg Bartholdt, Siemens

Jörg Bartholdt works as software architect at Siemens Corporate Technology, Germany. He supports Siemens business units in designing distributed software systems. His main areas of interest are service-oriented architectures, middleware, persistency and security.

SOAP WebServices started with the promise of being "simple". One can see that this is not true because the abbreviation "Simple Object Access Protocol" was turned into a plain name without any association intended anymore. Even worse, the number of WS-* standards increases and makes the overview more confusing. Which of them do I need and can be used today in productive applications? What about their maturity? How well are they supported in terms of implementations and global players? We will quickly lay the foundation of SOAP and WSDL and then have a look at various WS-* (e.g. WS-Addressing, WS-Security, WS-Trust, WS-Transaction, WS-ReliableMessaging, WS-Management): Who is supporting the standard/draft? How mature is it for production? Which implementations can I use?

T35: Scripting Your (and Your Company's) Second Life — Wednesday 08:30–12:00 — 10A

Cristina Lopes, University of California, Irvine

Cristina Lopes is an Associate Professor in the Bren School of Information and Computer Sciences at the University of California, Irvine. Prior to joining the Faculty in ICS, she was at Xerox PARC, where she co-founded the group that developed Aspect-Oriented Programming (AOP) and AspectJ. Her research focuses on language and tool support for better software practices, especially those pertaining to pervasive computing systems. Dr. Lopes holds B.S. and M.S. degrees in Electrical and Computer Engineering from Instituto Superior Tecnico, in Lisbon, and a Ph.D. in Computer Science from Northeastern University.

William Cook, The University of Texas at Austin

William Cook is an Assistant Professor in the Department of Computer Sciences at the University of Texas at Austin. His research is focused on object-oriented programming, programming languages, modeling languages, and the interface between programming languages and databases. Prior to joining UT in 2003, Dr. Cook was Chief Technology Officer and co-founder of Allegis Corporation. He was chief architect for several award-winning products, including the eBusiness Suite at Allegis, the Writer's Solution for Prentice Hall, and the AppleScript language at Apple Computer. He completed his Ph.D. in Computer Science at Brown University in 1989.

The virtual world Second Life (SL) is attracting a lot of attention both from people drawn to the stunning 3D visuals and its diverse communities, and companies drawn to SL's potential to become the "next phase of the Internet's evolution"—to quote IBM's CEO. Whether this prediction turns out to be true or not, SL is, no doubt, the most intriguing computing system that has emerged since the advent of the Web and its browsers. SL is a virtual world where (almost) everything is user-programmable. As such, the limit of what can be done is people's imaginations. This tutorial covers the basics of SL and of programming in LSL, SL's event-based, C-like, object-anchored scripting language. The tutorial starts from the very beginning, but, since we expect attendees to be familiar with C-like languages, we will cover advanced topics such as the integration of SL with the Web.

Tutorials

T36: Agile in Face of Global Software Development   Wednesday 08:30–12:00   514A

Jutta Eckstein, IT communication

Jutta Eckstein, a partner of IT communication, is an independent consultant and trainer from Braunschweig, Germany. Her know-how in agile processes is based on over ten years experience in developing object-oriented applications. She has a unique experience in applying agile processes within medium-sized to large mission-critical projects. This is also the topic of her book 'Agile Software Development in the Large'. Besides engineering software she has been designing and teaching OT courses in industry. Having completed a course of teacher training and led many 'train the trainer' programs in industry, she focuses also on techniques which help teach OT and is a main lead in the pedagogical patterns project. She has presented work in her main areas at ACCU (UK), JAOO (Denmark), OOPSLA (USA), XP (Europe) and Agile (USA). She is a member of the board of the AgileAlliance and a member of the program committee of many different European and American conferences in the area of agile development, object-orientation and patterns.

There are not many projects left that are made at home without some form of outsourcing, offshoring or nearshoring. Thus, global software development seems to be a fact in state-of-the-art software development. However, more and more projects are discovering the success factor of agile software development methods. Although agile software development methods suggest - among other things - an emphasis on face-to-face communication, several projects have tried meanwhile to combine global with agile software development. In this session Jutta shares experiences in following an agile approach in a global development environment: What are the possibilities to overcome the challenges that global software development provides, and what are the success factors for implementing an agile software development process within such constraints? In this talk, Jutta will report from her experiences in bringing these two trends together. Her own experiences are mainly based on large global agile projects in embedded and commercial software development.

T37: Use-Case Patterns and Blueprints   Wednesday 08:30–12:00   514B

Gunnar Overgaard, SEB AB

Gunnar Overgaard has been using, mentoring, and teaching use cases and participating in the development of the concept since 1987. He has also participated in the development of UML since 1997. Gunnar has given hundreds of classes and presentations, both for industry and academia. He is one of the authors of the book Use Cases: Patterns and Blueprints, Addison-Wesley, 2004.

Karin Palmkvist, Generic Integration AB

Karin Palmkvist has been working with use cases since the late 1980s, as a system analyst, mentor, teacher, and speaker at conferences and seminars. She also participated in the development and standardization of the UML within the OMG. Karin is one of the authors of the book Use Cases: Patterns and Blueprints.

Use-case modeling is a well-established technique for capturing requirements stating how a system is to be used. However, many struggle with how to produce complete and correct models. As a result, they find themselves solving more or less the same modeling problems over and over again. In this tutorial, we show how using patterns and blueprints for use-case models help to avoid these difficulties. We present a collection of useful patterns and blueprints and illustrate how they can be used to facilitate the development of accurate and understandable use-case models.

T38: Embedded Objects with C++: Idioms, Patterns and Architecture for Constrained Systems   Wednesday 08:30–12:00   515A

Detlef Vollmann, vollmann engineering gmbh

Detlef Vollmann has 25 years experience in software engineering, about 20 years in object technology, and more than 15 years experience with embedded systems. As an independent consultant, he supports several Swiss companies with the design of object-oriented systems for small and large systems. Since 1991, he has authored and taught courses in C++, object-oriented technologies, software architecture and embedded computing for major Swiss companies and international conferences.

Embedded software is different from other software, usually involving hard real-time constraints and very limited memory. But the challenges for embedded systems are even greater: they must be very reliable (99.999% often is not good enough) and need to deal with a variety of different memory types (standard RAM, EEPROM, Flash, buffered RAM). Another frequent design problem is that the system must work with different kinds of

processing tasks, including interrupt handling, hardware control, an d application processes, all interconnected by a selection of communication means. Many designers believe that objects have no role to play in such heavily constrained systems, but nothing could be farther from the truth. With the correct tools, object-oriented systems can perform better with tight resources than traditionally designed systems. C++ is such a tool that combines the full power of object-oriented design and programming with a maximum of performance while still allowing for the strict control necessary, e.g., for hard real-time requirements. This tutorial presents and discusses various techniques for designing and implementing typical embedded systems, leveraging object-oriented mechanisms while minimizing resource consumption. C++ will be used as the implementation language, and real-word examples will be shown.

T39: Pattern-Oriented Software Architecture: A Pattern Language for Distributed Computing
Wednesday 08:30–12:00   **515B**

Douglas Schmidt, Vanderbilt University

Douglas Schmidt is a Professor of Computer Science and Associate Chair of the Computer Science and Engineering program at Vanderbilt University. He has published 9 books and over 350 technical papers that cover a range of research topics, including patterns, optimization techniques, and empirical analyses of software frameworks and domain-specific modeling environments that facilitate the development of distributed compting middleware and applications. In addition to his academic research, Dr. Schmidt has over fifteen years of experience leading the development of ACE, TAO, CIAO, and CoSMIC, which are widely used, open-source middleware frameworks and model-driven tools.

Developing software for distributed computing applications that effectively utilizes concurrency over high-speed, low-speed, and mobile networks is hard; developing high quality reusable distributed applications is even harder. Many patterns in the software literature focus on distributed computing. Until recently, however, there has been no holistic view of distributed computing that emphasizes how groups of patterns complete and complement each other. Building complex distributed systems has therefore been a craft that many have tried, but few have mastered. To address this issue, this tutorial describes a pattern language that links hundreds of patterns relevant for distributed computing, including: . Object interaction . Interface and component partitioning . Application control . Resource management . Concurrency and synchronization This pattern language has been used successfully by the speaker on production distributed applications and middleware at hundreds of commercial companies for telecommunication systems, network management for personal communication systems, electronic medical imaging systems, real-time avionics and aerospace systems, and automated stock trading. The material presented in this tutorial appears in the book "Pattern-Oriented Software Architecture: A Pattern Language for Distributed Computing", Wiley & Sons, 2007 by Frank Buschmann, Kevlin Henney, and Douglas Schmidt.

T41: Behaviour Driven Development using JBehave
Wednesday 13:30–17:00   **514A**

Elizabeth Keogh, Thoughtworks Ltd.

Liz Keogh is an experienced Agile developer, mentor, and team-leader. She has been the lead developer on JBehave since 2005 and is responsible for the example project used to drive development of the Story framework. She is experienced in using BDD techniques on real-world projects and has successfully mentored newcomers to TDD and Agile using the same examples as this tutorial. Her blog is at: http://sirenian.livejournal.com Most recently Liz presented the Agile Haiku Workshop, exploring emotional communication without emotional language, at XP2005. She is also a published poet.

Dan North, Thoughtworks Ltd.

Dan is an experienced Agile developer, coach, mentor and leader. The story of how he developed the techniques of BDD and founded JBehave can be found here: http://dannorth.net/introducing-bdd Dan is a experienced presenter, most widely on subjects such as BDD, NLP, and the use of natural language in software. Most recently he presented "The Yawning Crevasse of Doom", or how to bridge the communication gap between business and developers, at QCon 2007 with Martin Fowler.

The bridge between developer jargon and business language can be hard to build. Using BDD techniques, developers can describe their software in terms that the customer understands. Acceptance tests become stories with reusable scenarios and steps, making the domain language executable. Unit tests become descriptions of behaviour and responsibility. This approach can lead to better design and code readability, aid in teaching and mentoring, give clarity to the customer and improve communication across the team. Emphasis will be on Behaviour Driven techniques, with JBehave used as a supporting tool.

T42: Fault Tolerant Software with Patterns
Wednesday 13:30–17:00   **514B**

Robert Hanmer, Alcatel-Lucent

Robert S. Hanmer is a Consulting Member of Technical Staff at Alcatel-Lucent. His work has included development, architecture and evaluation of highly reliable systems for the telephone network, especially in the areas of reliability and performance. He has also been active in the software patterns community since the mid-1990's, writing patterns and organizing pattern conferences. He has authored or co-authored many journal articles and book chapters primarily in the area of fault tolerant design. He has also written a book on software fault tolerance. He has been introducing newcomers to patterns and to writing patterns since 1995.

We rely more and more on computer software to conduct our business, maintain our links with society, and enhance our lives. We want those computer systems, whether they are web servers, ATMs, the internet or the phone system, to be ready to process our requests when we want them. But we are the people creating these systems; so we should know something about basic principles of fault tolerance. Fault tolerance is something that is designed into software systems. It is not free and it is more than a way of coding or a quality methodology. The ability to tolerate faults is important to computer systems because it allows them to detect, isolate, contain and process errors at runtime before they cause the system to fail. This tutorial looks at designing software to be fault tolerant. There are many techniques that individual architects and designers can put into their software allowing it to tolerate faults. These techniques have been used in many systems, from many different application domains, for many years. This tutorial lays the foundation for fault tolerant design. It starts with the basics and progresses to defining key design elements for fault tolerant software. A pattern-based approach is used.

Tutorials

Neil Harrison, Utah Valley State College

Neil Harrison is an assistant professor of computer science at Utah Valley State College in Orem, Utah. Before that, he was a distinguished member of technical staff at Avaya Labs, where he developed and tested communications software, and wrote testing tools and techniques. He is the co-author of the book, "Organizational Patterns of Agile Software Development". He has published numerous articles on software patterns, effective organizations, and software testing. He is acknowledged as the world's leading expert on pattern shepherding and the pattern conference (PLoP) shepherding award is named after him.

For developers, testing is hard. Even worse, it can be remarkably ineffective: we often invest a great deal of time and effort in test development and execution, only to let significant bugs get through to the customer. Developers are generally poor testers, but developers MUST test—they can't simply push all testing off to system or acceptance testers. Fortunately, there are things you can do to make your development testing much more effective. The answer lies in the way you approach writing your tests. Through a combination of objective and subjective techniques, you can write tests that are more likely to uncover bugs in your software. The best part is that it doesn't take a lot of effort: you will be working smarter, not harder. In addition, the basics of these techniques are easily learned. In this tutorial, you will learn how to write tests that give better coverage, find more errors, and are simply more effective for the time invested. You will also learn key mindsets that will make you more aware of where bugs may lurk. You will learn how to integrate these techniques with test-driven development (TDD), which makes TDD not only an important development tool, but a powerful quality assurance tool at the same time. Prior knowledge of TDD is not required.

Rebecca Wirfs-Brock, Wirfs-Brock Associates

*Rebecca Wirfs-Brock, president of…**See T24**.*

Agile designers need to quickly see the essence of a problem, shape reasonable solutions, and communicate complex ideas. When things don't go according to plan, they must react, readjust their thinking, and try again. This tutorial introduces techniques and vocabulary for articulating the nature of design problems and their solutions: Problem frames identify the typical structure of software tasks. Designer stories set the stage for collaborative design. Role stereotypes are useful for assigning or assessing object behaviors. Control styles help characterize and communicate predominant collaboration patterns. Trust regions can aid in spotting places where defensive programming is needed. Seasoned designers strike a balance and know how to get along in a team environment. They know the differences between core and revealing design tasks and plan accordingly. When unanticipated problems crop up, they adapt their work rhythms. And when conflict happens, they know they need to think critically and hold their own in a heated discussion.

Eric Evans, Domain Language, Inc

Eric Evans is a specialist in domain modeling and design in large business systems. Since the early 1990s, he has worked on many projects developing large business systems with objects and has been deeply involved in applying Agile processes on real projects. Out of this range of experiences emerged the synthesis of principles and techniques shared in the book "Domain-Driven Design," Addison-Wesley 2003. Eric now leads "Domain Language", a consulting group which coaches and trains teams to make their development more productive through effective application of domain modeling and design.

Large information systems need a domain model. Development teams know this, yet they often end up with little more than data schemas which do not deliver on the productivity promises for object design. This tutorial delves into how a team, developers and domain experts together, can engage in progressively deeper exploration of their problem domain while making that understanding tangible as a practical software design. This model is not just a diagram or an analysis artifact. It provides the very foundation of the design, the driving force of analysis, even the basis of the language spoken on the project. The tutorial will focus on three topics: The conscious use of language on the project to refine and communicate models and strengthen the connection with the implementation. A subtly different style of refactoring aimed at deepening model insight, in addition to making technical improvements to the code. A brief look at strategic design, which is crucial to larger projects. These are the decisions where design and politics often intersect. The tutorial will include presentation and discussion of selected patterns from the book "Domain-Driven Design," Addison-Wesley 2003, and reenactments of domain modeling scenarios.

T46: Real-time Programming on the Java Platform — Thursday 08:30–12:00 — 514B

David Holmes, Sun Microsystems Australia

*David Holmes is a…***See T19**.

Tony Printezis, Sun Microsystems

Tony Printezis is a Staff Engineer at Sun Microsystems, located in Burlington, MA. He has been contributing to the Java HotSpot Virtual Machine, as well as the Java Real-Time System, since the beginning of 2006. Before that, he spent over 3 years at Sun Labs. He spends most of his time working on dynamic memory management for the Java platform, concentrating on performance, scalability, responsiveness, parallelism, and visualisation of garbage collectors. He joined Sun Microsystems in 2002 after a 2.5-year research collaboration, while he was a member of the faculty of the Department of Computing Science of the University of Glasgow. He obtained a PhD in 2000 and a BSc (Hons) in 1995, both from the University of Glasgow in Scotland.

Real-time programming and Java are not normally synonymous. Real-time programming is often associated with specialized application domains such as avionics, process control and communication systems. Since the advent of the Real-time Specification for Java in 2000, the Java platform has extended its reach into these specialized domains. Meanwhile, mainstream Java application domains, such as banking and finance, have been pushing the limits on traditional Java platform technology. Often, what is sought is not raw performance, but predictability of response times - and this is what Real-time Java was designed to provide. One of the catalysts in the recent surge in interest in mainstream real-time Java programming is the development of Real-time Garbage Collection. RTGC provides a bridge between traditional Java development and the more complex memory management model provided within the RTSJ. This tutorial explores the concepts and API's behind the RTSJ and provides a primer on real-time garbage collection offerings. It covers priority-based scheduling and the use of real-time threads and asynchronous event handlers; asynchronous events and timers; plus alternative memory management using scoped-memory, and no-heap threads.

T47: Totally Awesome Computing: Python as a General-purpose Object-oriented Programming Language — Thursday 08:30–12:00 — 514C

Chuck Allison, Utah Valley State College

Chuck Allison has over 20 years of industrial software development experience, and is an associate professor of Computer Science at Utah Valley University. He was a contributing member of the C++ Standards Committee throughout the 1990s and designed std::bitset. He was an editor of one sort or another for the C/C++ Users Journal from 1992-2003, is the founding editor of The C++ Source and a columnist for Better Software Magazine. Chuck is the author of two C++ books and a regular lecturer at Software Development Conference.

Python is well-known as a scripting language and is often used in server-side web programming. What many people don't know is that it is a full-powered object-oriented programming language, with some functional programming support thrown in for good measure. Python is a portable, modular, very-high-level language. Programs in Python are typically 3-10 times smaller in code size than their Java or C++ equivalents, and take a fraction of the time to develop. The Python library is among the most robust of any available—whatever you need, chances are there's a Python library for you. (As they say, the Python release comes with "batteries included".) Another lesser-known fact is that Python evolved from a language designed to teach programming to children, hence its easy learning curve and its elegant appearance. Python interfaces nicely with C++ and Java, so you can call upon their facilities in the rare case that the standard Python distribution doesn't already solve your problem. This is a soup-to-nuts tutorial for attendees already familiar with most any modern programming language. Topics covered will include: A First Look, Types and Operations, Text Processing, Lists and Dictionaries, Tuples and Files, Statements and Control Structures, Functions and Functional Programming, Modules and Packages, Object-oriented Programming.

T48: Introduction to Software Product Line Development Methods — Thursday 08:30–12:00 — 515B

Charles Krueger, BigLever Software

Dr. Charles Krueger, Founder and CEO of BigLever Software, is a thought leader in the software product line field, with 20 years of experience in software development practice. He has proven expertise in helping companies establish highly acclaimed software product line practices, including Software Product Line Hall of Fame inductees LSI Logic and Salion. A frequent speaker at International Software Product Line Conferences, Dr. Krueger presented at SD Best Practices 2006, is an invited author for the CACM, and was recently featured in Dr. Dobb's and eWeek. He received his PhD in computer science from Carnegie Mellon University.

Methods and tools for software development tend to focus on individual products. However, most development organizations must create a product line - a portfolio of closely related products with variations in features and functions - rather than a single product. This mismatch has led to the emergence of software development methods, tools and techniques focused specifically on the challenges of software product line development. This tutorial explores the latest generation of software product line development methods that are yielding order-of-magnitude improvements in time-to-market, engineering cost, product quality, and portfolio scalability. It features best methods from recent industry case studies including model-driven, aspect-oriented, minimally-invasive, and agile strategies. The case studies featured in the tutorial, including Software Product Line Hall of Fame inductees LSI Logic/Engenio and Salion, provide insight into how a new generation of pragmatic methods are enabling companies of all types and sizes to realize a new level of benefits, in terms of both technical and business impact.

Tutorials

T21: Personas, Profiles, Actors, & Roles: Modeling users to target successful product design

Thursday 13:30–17:00 **510C**

Jeff Patton, ThoughtWorks

*Jeff Patton has designed…***See T8**.

Leveraging what you understand about your application's users is central to a user centered design approach. User personas have been touted as solution for improving the quality of user experience. But what exactly is a persona? How doe you create one? And, how does a persona differ from other popular approaches to modeling users such as actors, user roles, or user profiles?

In this tutorial you'll how user models fit into a holistic user centered design process. You'll learn about various types of user models. Through discussion and practice you'll learn how to create simple role models, user profiles, and personas. You'll learn how to leverage these user models to identify valuable features for your product, and overarching characteristics the design of your product must have to be successful for your target users.

This tutorial is fast paced, information dense, and full of collaborative activities to give you practice with the concepts and techniques introduced.

T49: Creating Plugins and Applications for the Eclipse Platform

Thursday 13:30–17:00 **510A**

Alex Romanov, Radview and Tel-Aviv University

Alex Romanov, a Senior Software Engineer at Radview Software Ltd., leading the next generation of open source, Eclipse based products for load testing of Web applications. Alex is an expert in Eclipse, and has a blog called "On Everything And Nothing In Particular" on the subject. In his spare time Alex is studying for Masters degree in the University of Tel-Aviv. Alex's research interests include Software Verification and Log Analysis.

Amir Kirsh, Comverse and Academic College of Tel-Aviv Yaffo

Amir Kirsh, a senior project manager in Comverse Network Systems, dealing with IMS and Quad Play solutions. Staff member at the Academic College of Tel-Aviv Yaffo, teaching Object Oriented and advanced programming courses. Amir is also an author of a very popular text book Object Oriented Programming in C++. Amir has a vast experience in both server side development (from ACE and C++ to J2EE) and with client side applications development (from WinAPI, MFC, Delphi and VB to C# and Eclipse).

Eclipse is known as a rich IDE, but is in fact much more than that. Eclipse is an extensible platform for the creation of standalone applications and for plugins for other Eclipse based applications (including the Eclipse IDE itself.) In this tutorial we will go through all phases of building an Eclipse-based application, from requirements to implementation. This is an entry level tutorial for Eclipse plugin development and as such will include a structured introduction on Eclipse Framework, Eclipse UI capabilities and the plugin architecture. The tutorial will cover the following topics: building Eclipse Plugins, Rich Client Platform (RCP), SWT and JFace, Eclipse MVC architecture using the Graphical Editing Framework (GEF) and more. The tutorial aimed for both SW engineers and managers, and is recommended for anyone wishing to understand the possibilities Eclipse provides for development of SW applications.

T50: Domain-Driven Design: Strategic Design for Larger Systems

Thursday 13:30–17:00 **510B**

Eric Evans, Domain Language, Inc

*Eric Evans is a…***See T45**.

Some design decisions have an impact on the trajectory of the whole project. Modeling is most needed in complex circumstances, yet the typical dynamics of large projects too often derail it or disconnect it from the real design. A related issue: modeling is best carried out by small, dynamic teams with a lot of autonomy, yet creating large systems requires coordination and project-spanning decisions. Managers and developers alike need to pay close attention to this intersection of design, project organization, and politics. This tutorial will introduce them to a suite of techniques

for that purpose. First, distilling a shared vision of the system's core and the roles of its parts can focus development effort on real business assets, and tell when 'good enough good enough' versus when to push for excellence. Then, "context mapping" addresses a vital fact of life: different groups model differently. Ignoring these realities leads to dumbed-down models and costly, buggy integrations, and disruption of project plans where they depend on other teams. Finally, who makes such decisions and how? Architecture teams disconnected from daily development make decisions with unintended consequences. Without prescribing a particular organization, we will consider guidelines for strategic decision making.

T51: Introduction to Ruby   Thursday 13:30–17:00   514A

Glenn Vanderburg, Independent Consultant

Glenn Vanderburg has over 20 years of experience as a software developer, working in diverse environments using a wide variety of languages and tools, including Java, C and C++, Perl, Tcl, and more. His career spans large enterprises, universities, and startups. He caught the Ruby bug in 2000, and has never enjoyed programming so much.

Ruby began catching the eye of western programmers in 2000 (after originating in Japan six years earlier) and its popularity has been growing steadily ever since. Recently, of course, the popularity of Ruby on Rails has increased interest in Ruby (and Rails' creator is the first to credit Ruby with making Rails possible). Ruby hits a lot of sweet spots: it's powerful, concise, clean, expressive, and has a wonderful community. This tutorial is an introduction to Ruby, intended to give you a head start on learning the language. It's targeted especially at those with a background in statically typed languages such as Java, C#, and C++, but any programmers with an understanding of OO will feel at home.

T52: Building Service-Oriented Architectures with Web Services   Thursday 13:30–17:00   514B

Olaf Zimmermann, IBM Research

Olaf Zimmermann is an Open Group Master Certified and IBM Senior Certified IT Architect and Research Staff Member. His areas of expertise include distributed computing and Service-Oriented Architecture (SOA) in general and J2EE/Web services in particular. Over recent years, Olaf has conducted numerous SOA/Web services engagements, and educated practitioners around the world on this technology. He is an author of the Springer text book "Perspectives on Web Services" (ISBN 3-540-00914-0). Olaf also contributed to several IBM ITSO Redbooks such as "Web Services Wizardry with WebSphere Studio Application Developer", SG24-6292-00. Olaf holds an honours degree in Computer Science from the Technical University in Braunschweig, Germany.

Service-Oriented Architecture (SOA) and Web services technologies have matured into highly attractive architecture and implementation alternatives for building distributed systems. SOA concepts and Internet protocol-based implementation stacks are a powerful combination that is well-suited for crafting heterogeneous business-to-business and enterprise application integration solutions. In this tutorial, we introduce SOA as an architectural style defined by patterns such as Enterprise Service Bus (ESB) and service composition. We present two industry case studies that demonstrate where and how these patterns can be applied in practice. Next, we present selected elements of the Web services technology standards stack from an application programmer's perspective, for example SOAP and WSDL. In a third module, we design and develop a complete sample application applying the introduced concepts and technologies, making use of the Web services editors and code generators in the Eclipse Web Tools Project (WTP). We conclude with a discussion of the key architectural decisions on SOA and Web services development projects - for example REST vs. SOAP message exchange; service interface creation process and granularity concerns; security, reliability and other quality-of-service factors; server-side deployment and client-side invocation guidelines. This discussion centers around the lessons learned on large-scale industry case studies.

T53: UML in Action   Thursday 13:30–17:00   514C

Mohamed E. Fayad, San Jose State University & vrlSoft, Inc.

Mohamed Fayad is a Full Professor of Computer Engineering at San Jose State University & a founder of vrlSoft, Inc, Silicon Valley, CA. He is a columnist for The Communications of the ACM, writing the column Thinking Objectively, and was guest editor on nine theme issues. He has written and published articles in many journals and magazines. He has given tutorials and presented various seminars in the USA and over 30 different countries.. Fayad has received an MS and a Ph.D. in computer science from the University of Minnesota at Minneapolis. He is the lead author of several Wiley books.

This unique tutorial details experience gained from over 100 architected and developed real systems using Unified Modeling Language models and diagrams. It includes practical knowledge that has not yet been published. There will be pertinent examples and working exercises. As the tutorial progresses, it will also focus on true analysis (problem understanding), finding ultimate design (solution) and creative heuristics and guidelines. Over twenty concepts and ten models are introduced and discussed at length. Active participation will help students quickly understand guided design examples. Each model discussed is carefully chosen to help cover a broad range of application areas. UML has become part of the mainstream software development protocol. This course will equip teams with the tools they need to apply UML correctly and to coordinate and manage the development of complex software systems. Participants will apply their skills through a number of interactive, mini design sessions, where the instructor helps them overcome common obstacles.

Workshops

Chair: Michael Kircher, Siemens Corporate Research

ooPSLA workshops provide a creative and collaborative environment where attendees meet to surface, discuss, and solve challenging problems related to a variety of research topics. Workshops provide a great opportunity for researchers and practitioners to establish as well as foster communities on these topics.

The topics of workshops as well as their formats are diverse. For example, workshops may provide an opportunity for people working in a particular area to coordinate efforts and to establish a collective plan of action, to collaborate on a book, to seek contribution, and to discuss and share ideas on a hot new emerging technology.

W1: Process in oo Pedagogy— The sixth "Killer Examples" workshop	Sunday 08:30–17:00	512B

http://www.cse.buffalo.edu/faculty/alphonce/OOPSLA2007/

The "Killer Examples" series of workshops are highly interactive workshops whose goals are to bring together educators and developers to share their design pattern and object-oriented expertise, and to provide a forum for discussion of teaching techniques and pedagogical goals. These workshops have been an annual occurrence at OOPSLA since 2002. The theme of this year's workshop is "process", for teaching, learning and programming. While there is a formal application procedure to guarantee admission to the workshop, we do accept walk-ins if space permits and the walk-ins are determined to have adequate interest and background in the workshop theme to be able to contribute positively to the discussions.

Carl Alphonce, University at Buffalo, SUNY
Jürgen Börstler, Umeå University
Michael Caspersen, University of Aarhus

Adrienne Decker, University at Buffalo, SUNY
Michael Kölling, University of Kent

W2: Managing Complexity	Sunday 08:30–17:00	512H

http://www.kmarquardt.de/workshops/ManagingComplexity/index.htm

Projects, once stranded, suffocate from their own weight. The weight of the project is the complexity that it has created, or that has been burdened upon it. Complexity of the problem, of the organization, of the chosen solution, of the environment and of the team dynamics. While complexity cannot be avoided, it can be influenced and managed. This workshop explores how complexity arrives at a project, how it can be measured, what heuristics indicate risk, and how complexity can be managed and overcome.

Klaus Marquardt, Dräger Medical
Lise Hvatum, Schlumberger

Jens Coldewey, Coldewey Consulting

W3: 1st International Workshop on In Process Software Engineering Measurement and Analysis (ISEMA 2007)	Sunday 08:30–17:00	514C

http://www.case.unibz.it//index.php?option=com_content&task=view&id=132&Itemid=74

Improving the software engineering development process requires collection of data, but collection of data interferes with how developers work. At present, most of the software engineering tools, data collection, and analysis techniques available use manual data collection, despite known problems with reliability, correctness, and timeliness of the data. To overcome such limitations and reduce interference with the development process, software engineering researchers must develop tools and data analysis techniques that collect data without human interactions. Such tools produce very detailed and extensive data, but lack the filtering and classification that humans perform on manually collected data. This unfiltered data requires the development of new analysis techniques and new prediction models to use it effectively. This workshop focuses on defining research challenges created by in process software measurement and analysis of the software development process using tools that do not affect or modify the process but extract data automatically from it.

Philip Johnson, University of Hawaii

Alberto Sillitti, Free University of Bolzano

W4: The First International Workshop on Patterns Languages: Addressing Challenges
Sunday 08:30–17:00   **515B**

http://www.engr.sjsu.edu/~fayad//workshops/PLAC07/

Pattern languages have emerged as a promising classification technique and in providing ways to build frameworks. However, there area number of problems, such as: 1. Context's missing indicators/guidelines for in-context patterns selection within the pattern language. 2. Classifications of patterns' rationale within the pattern language structure is also missing. 3. Traceability is lost, especially when dealing with deeper levels of pattern language implementation. 4. No systematic way for compositing these patterns, similar or different, to build software architectures 5. There is a loss of generality in traditional pattern languages. 6. Pattern languages struggles and conflicts in providing full software maintainability and stability. 7. How pattern languages deal with the problem they address is neither straightforward nor easy. 8. There is no set classification in pattern languages. The workshop will address pattern languages' challenges and debateissues related to pattern language creation and development, selection process, composition, reuse, and impacts

Mohamed E. Fayad, San Jose State University & vrlSoft, Inc.
Chia-chu Chiang, University of Arkansas at Little Rock
Huascar A. Sanchez, Independent Consultant

Pablo Chacin, Technical University of Catalonya
A. Kannammal, Coimbatore Institute of Technology, TamilNadu

W5: Fifth International Workshop on SOA & Web Services Best Practices
Sunday 08:30–17:00   **516D**

http://boss.bekk.no/oopsla2007/

In this 5th International Workshop on SOA and Web Services, we will explore best-practices in the adoption, design, implementation, management and monitoring of SOA- and Web services-related methods, tools and technologies. SOA and Web services are supported by nearly all major software vendors and industry analysts. However, there are many challenges in the area of engineering SOA. In this workshop, we aim to share experience, assess the state-of-the-art and the state-of-the-practice, consolidate successful techniques, and identify the most promising application areas and open issues for future work. The workshop brings industry and academia together to publish a compendium of best practices.

This workshop is a continuation of a highly successful series of workshops at OOPSLA 2006, OOPSLA 2005, OOPSLA 2004, and OOPSLA 2003.

Olaf Zimmermann, IBM Research
Anders Aas Bjerkestrand, Bekk Consulting

Dr. Amir Zeid, British University in Egypt
Lars Arne Skaar, TietoEnator Banking Solutions

W6: The 7th OOPSLA Workshop on Domain-Specific Modeling
Sunday 08:30–17:00   **519A**

http://www.dsmforum.org/events/DSM07/

An upward shift in abstraction leads to a corresponding increase in productivity. In the past this has occurred when programming languages have evolved towards a higher level of abstraction. Today, domain-specific modeling languages provide a viable solution for continuing to raise the level of abstraction beyond coding, making development faster and easier. In domain-specific modeling (DSM) the models are constructed using concepts that represent things in the application domain, not concepts of a given programming language. The modeling language follows the domain abstractions and semantics, allowing developers to perceive themselves as working directly with domain concepts. Together with frameworks and platforms, DSM can automate larger portion of software production. Workshop topics: **o** Industry/academic experience reports **o** Creation of metamodel-based languages **o** Novel approaches for code generation from domain-specific models **o** Issues of support/maintenance for systems built with DSMs **o** Evolution of languages **o** Metamodeling frameworks and languages **o** Tool support

Juha-pekka Tolvanen, MetaCase
Jeffrey G. Gray, University of Alabama at Birmingham

Matti Rossi, Helsinki School of Economics
Jonathan Sprinkle, University of California, Berkeley

W7: Eclipse Technology Exchange 2007
Sunday 08:30–17:00   **519B**

http://www.cs.mcgill.ca/~martin/etx2007

The Eclipse platform (http://www.eclipse.org) is designed for building integrated development environments (IDEs) for object-oriented application development. The theme of the workshop is the use of the Eclipse open source as a code base for teaching and research. The central activity of the workshop will be sharing the results of such projects, including discussion of potential new uses of Eclipse and of improving the core Eclipse technology with respect to ease and efficiency of use. One goal of the workshop would be to promote growth of the Eclipse community, and to increase interaction between its members. Another goal would be to expand the use of Eclipse technology and improve the usability of its core technology.

Li-te Cheng, IBM Research
Alessandro Orso, Georgia Institute of Technology

Martin Robillard, McGill University

Workshops

Mini-PLoP Bootcamp: Pattern Writing Bootcamp — Sunday 08:30–17:00 — 512E

http://refactory.com/OOPSLA_Workshop.html

Patterns have been and still are a mainstay at OOPSLA, for example ten per cent of workshops and tutorials at OOPSLA 2006 had the word "pattern" in their names. This workshop provides an introduction to patterns and pattern writing, a "bootcamp" aimed at those who are new to patterns. The bootcamp provides an overview of patterns in a training, mentoring, experiencing way, where participants will be immersed in patterns and emerge with an enlarged perspective and a better ability to understand writing, reading, and using patterns.

Robert Hanmer, Alcatel-Lucent   **Linda Rising**, Independent consultant

W6: The 7th OOPSLA Workshop on Domain-Specific Modeling — Monday 08:30–17:00 — 519A

http://www.dsmforum.org/events/DSM07/

An upward shift in abstraction leads to a corresponding increase in productivity. In the past this has occurred when programming languages have evolved towards a higher level of abstraction. Today, domain-specific modeling languages provide a viable solution for continuing to raise the level of abstraction beyond coding, making development faster and easier. In domain-specific modeling (DSM) the models are constructed using concepts that represent things in the application domain, not concepts of a given programming language. The modeling language follows the domain abstractions and semantics, allowing developers to perceive themselves as working directly with domain concepts. Together with frameworks and platforms, DSM can automate larger portion of software production. Workshop topics: o Industry/academic experience reports o Creation of metamodel-based languages o Novel approaches for code generation from domain-specific models o Issues of support/maintenance for systems built with DSMs o Evolution of languages o Metamodeling frameworks and languages o Tool support

Juha-pekka Tolvanen, MetaCase   **Matti Rossi**, Helsinki School of Economics
Jeffrey G. Gray, University of Alabama at Birmingham   **Jonathan Sprinkle**, University of California, Berkeley

W8: Integration of Open Source Components into Large Software Systems — Monday 08:30–17:00 — 512A

http://www.carleton.ca/tim/oopsla

Developing large software systems has largely become an exercise in integration. About 85% of code that goes into the software of a typical system is written by others, and the main role of businesses is to write the "glue" that holds the externally developed components together. While in the past, businesses were largely concerned with the integration of commercial off-the-shelf (COTS) components, many of these components will now come as free/open source software (F/OSS) components. The use of open source components provides new strategic options for reducing the exposure to risk and cost of development, while significantly increasing the available solutions. Models for the integration of COTS components do not necessarily apply to open source components. A particular focus in this workshop will be on the shift away from COTS to F/OSS components, and what new opportunities and issues are introduced by it.

Michael Weiss, Carleton University   **Peter Carbone**, Nortel
Tony Bailetti, Carleton University

W9: Versions, Releases, and Distribution — Monday 08:30–17:00 — 512B

http://www.kmarquardt.de/workshops/VersionsReleasesDistribution.htm

Software developers rightfully focus on the activities needed to polish the software for its first release. The management of future releases, version identification, compatibility checks, and update strategies are typically treated as an afterthought, but insufficiencies or inconsistencies here have all the potential to make your life miserable once the software has hit the market. This workshop explores the path of the developed software to the customer, and strives to improve it. Special attention is paid to combine knowledge from different project kinds to facilitate mutual learning experiences.

Klaus Marquardt, Dräger Medical   **Lise Hvatum**, Schlumberger

W10: The Popularity Cycle of Graphical Tools, UML, and Libraries of Associations

Monday 08:30–17:00 **512C**

http://www.codefarms.com/OOPSLA07/workshop

A cycle has been observed: As our programs grow complex, graphical tools become popular - but then a new paradigm emerges, makes the tools obsolete, and we return to the more compact textual programming. Could this happen to UML and what this new paradigm would be? Could we design generic associations which are missing in our class libraries today? Could the existing OO languages support such reusable associations? The workshop will combine three views: High level design (MDA), low level implementation (libraries of associations), and the features which OO languages need in order to support libraries of associations and design patterns.

Jiri Soukup, Code Farms Inc. **Martin Soukup**, Nortel Networks

W11: The First Workshop on Programming Languages and Integrated Development Environments (PLIDE)

Monday 08:30–17:00 **512E**

http://lamp.epfl.ch/~mcdirmid/PLIDE

The proposed WIDE workshop will bring together practitioners and researchers who are active in or have experience in IDE language support. By sharing and discussing techniques in this field, we can improve our own efforts by (a) not independently repeating each others' mistakes and (b) learning what others have already discovered. Hopefully, our discussion will help to transform the black-art of crafting IDE language support into a more established engineering that is well documented and eventually taught in compiler courses. At the very least, we can make contacts and form a small community around this very important field.

Sean Mcdirmid (Ecole Polytechnique Federale de Lausanne, EPFL) **Julian Dolby**, IBM T.J. Watson Research Center
Robert Fuhrer, IBM T.J. Watson Research Center **Eugene Vigdorchik**, Jet Brains

W12: No Silver Bullet - a Retrospective on the Essence and Accidents of Software Engineering

Monday 08:30–17:00 **512G**

http://mysite.verizon.net/dennis.mancl/oopsla07/index.html

Twenty years ago, Fred Brooks wrote a landmark article on software engineering: "No Silver Bullet - Essence and Accident in Software Engineering." Brooks presented a set of useful constructive criticisms of the state of the art in software development. What if we had a chance to rewrite Brooks' article today? What have we learned about effective software development techniques over the last 20 years? Do we have some experiences that reinforce or contradict Brooks' thesis? Workshop participants will be challenged to try to refactor a "classic."

Steven Fraser, Cisco Systems **Bill Opdyke**, Motorola
Dennis Mancl, Alcatel-Lucent Bell Labs

W13: Semantic-Based Systems Development

Monday 08:30–17:00 **512H**

http://www.fluidbusiness.org/events/oopsla2007/

Software systems are intrinsically complex from a number of perspectives. The level of complexity is increasing due to the growing need to integrate different and diverse systems in order to achieve organizational goals (evolving toward ecosystems as an ideal). Effective integration Mayo be argued as important as, from a knowledge-based perspective of organizations, the effectiveness of services rendered by resources depends upon how they are combined and applied. Consequently, systems development and re-engineering must focus on ways of dealing with the complexity of modern software systems in an effective manner. Since software systems, in essence, model real world phenomena, it is necessary to adopt modeling and development techniques founded on semantics. Broadly speaking, semantics enable the precise mapping between complex real world phenomena and their modeled counterparts and/or enable the (dynamic) mapping/integration between different representations (and understandings) of real world phenomena. With that in mind, this workshop aims to bring together researchers and practitioners with diverse cultural and professional backgrounds in order to discuss and analyze the different perspectives, issues and challenges of Semantic-Based Systems Development.

Sergio De Cesare, Brunel University **Carsten Holtmann**, Karlsruhe University
Grant Holland, Sun Microsystems **Mark Lycett**, Brunel University

Workshops

W14: The First International Workshop on Unified Data Mining Engine: Addessing Challenges   Monday 08:30–17:00   514C

http://www.engr.sjsu.edu/~fayad/workshops/UDME07/

Building such a Unified Data Mining Engine (UDME) is not an easy exercise, specifically, when several factors can undermine their quality success, such as cost, time, and lack of systematic approaches. We would like to architect and develop a UDME, that has the some or all of the following properties: 1. Ease of use, 2. No Need of Expert to run the tool 3. Easy to add new functionality 4. Easy to interface 6. Multiple algorithms 7. Fewer resources 8. Stable 9. Isolation of Application logic 10. Minimum Maintenance Cost The workshop will address the unified data mining engine' challenges, and also debate several issues that are related to the architecture and development of the UDME.

Mohamed E. Fayad, San Jose State University & vrlSoft, Inc.   **Somenath Das**, eBay, Inc.
Tarek Helmy, King Fahd University of Petroleum and Minerals   **Eduardo M. Segura**, San Jose State University & vrlSoft, Inc.

Mini-PLoP: Writers Workshop for Papers in Software Development   Monday 08:30–17:00   517C

http://refactory.com/OOPSLA_Workshop.html

There has been recognition that the writers' workshop process is beneficial for the production of high quality papers ever since the first patterns conference or PLoP was held in 1994. In this OOPSLA workshop, attendees will participate in a writers' workshop of their paper as well as writers' workshops for papers of other attendees in their group. Participants who do not have a paper to workshop are also invited. Everyone will be expected to read all the papers in their groups in advance. The organizers of this workshop were all involved in PLoP '06 and would like to continue a close association between this important patterns activity and OOPSLA. Our hope is that the relationship will benefit both OOPSLA, by attracting authors of papers that are works-in-progress, and PLoP, by allowing authors who might not be able to attend PLoP to participate in the writers' workshop process.

Robert Hanmer, Alcatel-Lucent   **Joseph W. Yoder**, The Refactory Inc & Joe Yoder Enterprises
Linda Rising, Independent consultant

W15: Agility Unlimited?   Wednesday 08:30–17:00   514C

http://www.coldewey.com/publikationen/conferences/oopsla2007/agilityUnlimited.html

When agile methods arose, they were tried in small, collocated teams developing medium critical client or server software mostly with internal teams or effort-based contracts. Even today many proponents and skeptics of agile development believe that these parameters limit its use. Others have tried to cross these boundaries and have applied the agile value system to large teams, distributed development, highly critical or embedded systems or in an environment that doesn't support decision-making. Some of them have succeeded, others have failed. Most of them had to adapt agile techniques to their specific context. This workshop tries to deepen the understanding of agile values, principles and techniques by exploring these experiences. We invite practitioners and academics who have experience with testing the boundaries of agility or are interested in their findings.

Klaus Marquardt, Dräger Medical   **Johannes Link**, Software Activist
Jens Coldewey, Coldewey Consulting

ACM / SIGPLAN / SIGSOFT

ACM, the Association for Computing Machinery http://www.acm.org, is an educational and scientific society uniting the world's computing educators, researchers and professionals to inspire dialogue, share resources and address the field's challenges. ACM strengthens the profession's collective voice through strong leadership, promotion of the highest standards, and recognition of technical excellence. ACM supports the professional growth of its members by providing opportunities for life-long learning, career development, and professional networking.

The ACM Special Interest Group on Programming Languages explores programming language concepts and tools, focusing on design, implementation, and efficient use. Its members are programming language users, developers, implementers, theoreticians, researchers, and educators. The monthly newsletter ACM SIGPLAN Notices publishes several conference proceedings issues, regular columns, and technical correspondence. This SIG offers an additional newsletter, FORTRAN Forum, on a subscription-only basis. SIGPLAN sponsors four major annual conferences: the ooPSLA Conference on object-oriented Technology, the Conference on Programming Language Design and Implementation (PLDI)-the major professional conference in the field, the Symposium on Principles of Programming Languages (POPL), and the International Conference on Functional Programming (ICFP).

The ACM Special Interest Group on Software Engineering seeks to improve our ability to engineer software by stimulating interaction among practitioners, researchers, and educators; by fostering the professional development of software engineers; and by representing software engineers to professional, legal, and political entities.

Contact ACM to Join

Voice:
1 800 342-6626 (US/Canada)
+1 212 626-0500 (Global)

Fax:
+1 212 944-1318 (Global)

Email:
acmhelp@acm.org

Web:
http://www.acm.org/
http://www.sigplan.org/
http://www.sigsoft.org/

ooPSLA 2007 Committees

Conference Committee

General
Richard P. Gabriel, *IBM Research*
Research
David F. Bacon, *IBM Research*
Onward!
Cristina Videira Lopes, *University of California, Irvine*
Essays
Guy L. Steele Jr., *Sun Microsystems Laboratories*
Treasurer
Dirk Riehle, *SAP Research, SAP Labs LLC*
Communications
Eugene Wallingford, *University of Northern Iowa*
Corporate Support
Steven Fraser, *Cisco Research*
Invited Talks and Presentations
Robert Biddle, *Carleton University*
Demonstrations
Brian Sletten, *Zepheira, LLC*
DesignFest®
John Brant, *The Refactory*
Don Roberts, *University of Evansville*
Doctoral Symposium
Elisa Baniassad, *The Chinese University of Hong Kong*
Educators Symposium
Joe Bergin, *Pace University*
Steve Metsker, *Dominion Digital*
Lightning Talks
Cédric Beust, *Google*
David Leibs, *AMD*
Panels
Joe Yoder, *The Refactory*
Posters
Nadyne Mielke, *Microsoft Corporation*

Practitioner Reports
Ralph Johnson, *University of Illinois*
Student Research Competition
Nadyne Mielke, *Microsoft Corporation*
Student Volunteers
Michael Richmond, *IBM Research*
Technology
Ken Bauer, *Bauer Alonso Consulting*
Belia Romero, *Bauer Alonso Consulting*
Tutorials
Brian Goetz, *Sun Microsystems*
Workshops
Michael Kircher, *Siemens Corporate Research*
Advertising
Gail E. Harris, *Instantiated Software Inc.*
Graphic Design
Rebecca Rikner
Local Arrangements
Martin Robillard, *McGill University*
Steering Committee Chair
Mamdouh Ibrahim, *IBM*
ACM Liaison
Brooke Hardy, *ACM*
Operations
Ms. Mike Thomas, *Meeting Strategies Worldwide, Inc.*
Mary Cameron, *Meeting Strategies Worldwide, Inc.*
Registration
Carole Mann, *Registration Systems Lab*
Submission System
Richard van de Stadt, *Borbala Online Conference Systems*

Program Committees

Research Program Committee
David F. Bacon, *IBM Research (Chair)*
Andrew Begel, *Microsoft Research (USA)*
Steve Blackburn, *Australian National University (Australia)*
Siobhán Clarke, *Trinity College Dublin (Ireland)*
Cliff Click, *Azul Systems (USA)*
Charles Consel, *INRIA (France)*
William Cook, *The University of Texas at Austin (USA)*
Lidia Fuentes, *Universidad de Malaga (Spain)*
David Gay, *Intel Research Berkeley (USA)*
Dan Grossman, *University of Washington (USA)*
Tim Harris, *Microsoft Research, Cambridge (UK)*
Matthias Hauswirth, *University of Lugano (Switzerland)*
Michael Hicks, *University of Maryland College Park (USA)*
David Holmes, *Sun Microsystems (Australia)*

Arno Jacobsen, *University of Toronto (Canada)*
Eric Jul, *DIKU (Denmark)*
Gregor Kiczales, *University of British Columbia (Canada)*
Cristina Videira Lopes, *University of California, Irvine (USA)*
Ana Milanova, Rensselaer *Polytechnic Institute (USA)*
Robert O'Callahan, *Mozilla (New Zealand)*
Radu Rugina, *Cornell University (USA)*
Koushik Sen, *University of California, Berkeley (USA)*
Yannis Smaragdakis, *University of Oregon (USA)*
Guy L. Steele Jr., *Sun Labs (USA)*
Mandana Vaziri, *IBM Research (USA)*
Eelco Visser, *Delft University of Technology (Netherlands)*
Jan Vitek, *Purdue University (USA)*
Amiram Yehudai, *Tel Aviv University (Israel)*

Onward! Program Committee

Cristina Videira Lopes, *University of California, Irvine (Chair)*
Ron Goldman, *Sun Labs (USA)*
Andre van der Hoek, *University of California, Irvine (USA)*
Brian Kernighan, *Princeton University (USA)*
Doug Lea, *State University of New York, Oswego (USA)*
Wolfgang De Meuter, *Vrije Universiteit (Belgium)*
Mira Mezini, *Technische Universität Darmstadt (Germany)*
Harold Ossher, *IBM Research (USA)*
Alexander Repenning, *University of Colorado (USA)*
Dirk Riehle, *SAP Research, SAP Labs LLC (USA)*
Dave Thomas, *Bedarra Labs (Canada)*

Essays Program Committee

Guy L. Steele Jr., *Sun Labs (Chair)*
Robert Cartwright, *Rice University (USA)*
William Cook, *The University of Texas at Austin (USA)*
Mira Mezini, *Technische Universität Darmstadt (Germany)*
Simon Peyton-Jones, *Microsoft Research, Cambridge (UK)*
Richard Schmitt, *West Virginia Wesleyan College (USA)*

Doctoral Symposium Committee

Elisa Baniassad, *The Chinese University of Hong Kong (HK)*
Siobhán Clarke, *Trinity College (Ireland)*
Todd Millstein, *University of California, Los Angeles (USA)*
James Noble, *Victoria University of Wellington (New Zealand)*
Martin Rinard, *Massachusetts Institute of Technology (USA)*
Douglas C. Schmidt, *Vanderbilt University (USA)*

Educators Symposium Committee

Joseph Bergin, *Pace University (Chair)*
Steve Metsker, *Dominion Digital (Chair)*
Jutta Eckstein, *IT-communication (Germany)*
Ed Gehringer, *North Carolina State University (USA)*
Neil Harrison, *Utah Valley State College (USA)*
Jim Heliotis, *Rochester Institute of Technology (USA)*
Kevlin Henney, *Curbralan (UK)*
Joshua Kerievsky, *Industrial Logic (USA)*
Grigori Melnik, *University of Calgary (Canada)*
Rick Mercer, *University of Arizona (USA)*
Venkat Subramaniam, *Agile Developer (USA)*
Eugene Wallingford, *University of Northern Iowa (USA)*

Panels Committee

Joseph Yoder, The Refactory (Chair)
Ademar Aguiar, Universidade do Porto (Portugal)
Elisa Baniassad, The Chinese University of Hong Kong (HK)
Steven Fraser, Cisco Systems (USA)

Posters Committee

Nadyne Mielke, *Microsoft Corporation (Chair)*
Yvonne Coady, *University of Victoria (Canada)*
Martin Lippert, *University of Hamburg (Germany)*
Doug Lea, *State University of New York, Oswego (USA)*
Rick Mercer, *University of Arizona (USA)*
Laurie Williams, *North Carolina State University (USA)*

Practitioner Reports Committee

Ralph Johnson, *University of Illinois (Chair)*
Alan Knight, *Cincom Systems (USA)*
Amanda Silver, *Microsoft Corporation (USA)*
Donald Smith, *Eclipse Foundation (USA)*
Olaf Zimmermann, *IBM Research (USA)*

Student Research Competition

Nadyne Mielke, *Microsoft Corporation(Chair)*
Gavin Bierman, *Microsoft Research (USA)*
Michael Richmond, *IBM Research (USA)*
Eugene Wallingford, *University of Northern Iowa (USA)*

Tutorials Committee

Brian Goetz, *Sun Microsystems (Chair)*
Shail Arora, *Gradepoint, Inc. (USA)*
Joe Bowbeer, *independent consultant (USA)*
Steve Metsker, *independent consultant (USA)*
Ted Neward, *independent consultant (USA)*
Bill Pugh, *University of Maryland (USA)*
John Rose, *Sun Microsystems (USA)*
Russ Rufer, *independent consultant (USA)*
Brian Sletten, *Zepheira, LLC (USA)*
Glenn Vanderburg, *independent consultant (USA)*
Eugene Wallingford, *University of Northern Iowa (USA)*

Workshops Committee

Michael Kircher, *Siemens Corporate Research (Chair)*
Jutta Eckstein, *IT-communication (Germany)*
Aniruddha Gokhale, *Vanderbilt University (USA)*
Prashant Jain, *IBM India Research Lab (India)*
Gail Murphy, *University of British Columbia (Canada)*

Steering Committee

Mamdouh Ibrahim, *Chair*
Ralph Johnson, *OOPSLA 2005 Chair*
Peri Tarr, *OOPSLA 2006 Chair*
Richard P. Gabriel, *ooPSLA 2007 Chair*
Gail E. Harris, *ooPSLA 2008 Chair*
Kathleen Fisher, *SIGPLAN Chair*
Chandra Krintz, *SIGPLAN Vice Chair*
Brooke Hardy, *ACM Liaison*

Most Influential 1997 OOPSLA Paper

Kathleen Fisher (chair), AT&T Labs (USA)
Toby Bloom, MIT (USA)
Ben Zorn, Microsoft Research (US)
William Cook, The University of Texas at Austin
Peri Tarr, IBM Research (IBM)
Mary Loomis, MSCI Barra (USA)

Dynamic Languages Symposium Committee
Gilad Bracha, *Cadence Design Systems (USA)*
Johan Brichau, *Université Catholique de Louvain (Belgium)*
William Clinger, *Northeastern University (USA)*
William Cook, *University of Texas at Austin (USA)*
Pascal Costanza, *Vrije Universiteit Brussel, Belgium (co-chair)*
Stéphane Ducasse, *Université de Savoie (France)*
Brian Foote, *Industrial Logic (USA)*
Robert Hirschfeld, *Hasso-Plattner-Institut Potsdam, (co-chair)*
Jeremy Hylton, *Google (USA)*
Shriram Krishnamurthi, *Brown University (USA)*
Michele Lanza, *University of Lugano (Switzerland)*
Michael Leuschel, *Universität Düsseldorf (Germany)*

Henry Lieberman, *MIT Media Laboratory (USA)*
Martin von Löwis, *Hasso-Plattner-Institut Potsdam, Germany*
Philippe Mougin, *OCTO Technology (France)*
Oscar Nierstrasz, *University of Berne (Switzerland)*
Kent Pitman, *PTC (USA)*
Ian Piumarta, *Viewpoints Research Institute (USA)*
Nathanael Schärli, *Google (Switzerland)*
Anton van Straaten, *AppSolutions.com (USA)*
Dave Thomas, *Bedarra Research Labs (Canada)*
Dave Ungar *(USA)*
Allen Wirfs-Brock, Microsoft (*USA*)
Roel Wuyts, IMEC & Unversité Libre de Bruxelles (Belgium)

Additional Research and Onward! Reviewers

Mercedes Amor	Philip Haller	Polyvios Pratikakis
Matthew Arnold	Christian Hammer	Tony Printezis
Sushil Bajracharya	Michael Hind	Jean Privat
Reimer Behrends	Tony Hoare	Xin Qi
Michael Bond	Terry Hon	Mukund Raghavachari
Chandrasekhar Boyapati	Shan Shan Huang	Nasko Rountev
John Boyland	Ali Ibrahim	Barbara Ryder
Martin Bravenboer	Bart Jacobs	Jennifer Sartor
Shane Brennan	Ranjit Jhala	Vibha Sazawal
Yrom-David Bromberg	Pallavi Joshi	Reza Sherafat
Matthew Chalmers	Maria Jump	Mati Shomrat
Satish Chandra	Tony Kay	Wolf Siberski
Sigmund Cherem	Andrew Kennedy	Arjun Singh
Jeremy Condit	Julien Lancia	Murray Spork
Christoph Csallner	Doug Lea	Saurabh Srivastava
Sam Davis	Daan Leijen	Christopher Stone
Amit Deshpande	Benjamin Lerner	Nikhil Swamy
Danny Dig	Yin Liu	Pablo Snchez
Julian Dolby	Alexey Loginov	Peter Thiemann
Paul Dourish	David Mandelin	Eli Tilevich
Zoe Drey	Maria-Cristina Marinescu	Michael Toomim
Andrew D. Eisenberg	Vijay Menon	Shmuel Tyszberowicz
Burak Emir	Julien Mercadal	Ellen Van Paesschen
Rob Ennals	Vinod Muthusamy	Carlos Varela
Yishai A. Feldman	Aleks Nanevsky	Martin Vechev
Stephen Fink	Bonnie Nardie	Gina Venolia
Cormac Flanagan	Immad Naseer	Mathieu Verbaere
Matthew Flatt	Iulian Neamtiu	Joost Visser
Matthew Fluet	Srinivas Nedunuri	Westley Weimer
Jeff Foster	James Noble	Adam Welc
Daniel Frampton	Jacques Noyé	Ben Wiedermann
Vladimir Gapeyev	Nate Nystrom	Benjamin Wiedermann
Robin Garner	Nathaniel Nystrom	Eric Wohlstadter
Arbi Ghazarian	Manuel Oriol	Tobias Wrigstad
Ryan M. Golbeck	Yossi Peery	Alex Wun
Michael Gong	Igor Peshansky	Tao Xie
Christian Grothoff	Monica Pinto	Eran Yahav
Sam Guyer	Filip Pizlo	Charles Zhang

Conference Chair: Gail E. Harris, Instantiated Software Inc.

Wow, what a program we've had this year! I was impressed by the innovative and practical solutions that ooPSLA attendees have found to the current challenges in software development and engineering. This proves, for the 22nd time, how relevant ooPSLA is to both academia and to industry.

At the same time, I have also developed a deeper appreciation for the challenges that still remain. We still need to improve our programming techniques and languages to reduce errors and security vulnerabilities. We have yet to make optimal use of multi-core processors, let alone efficiently make use of thousands of them assembled into ultra large systems (ULS). On the other end of the spectrum, there are the ubiquitous computers and embedded-devices that are showing up in everything from your cell phone (or is it now a PDA, or an MP3 player or your medic alert bracelet with your medical history) to your groceries (which will confer and calculate the healthiness of your order). I can't wait till next year to see how you have researched these challenges and applied the solutions to the "real world."

So mark your calendars for Oct 19–23, 2008, and get ready to meet at the Nashville Tennessee Convention Center, in music-rich Nashville Tennessee for the 23rd ooPSLA. On behalf of the entire ooPSLA 2008 committee, we look forward to seeing you next year.

ooPSLA 2008 Important Dates

March 19, 2008
Submission deadline for Research Program, Onward!, Development Program,
Educators' Symposium and proposals for Tutorials, Panels, Workshops, and DesignFest®

July 2, 2008
Submission deadline for Development Program Briefs, Doctoral Symposium,
Student Volunteers

Watch http://oopsla.org for further details.

OOPSLA 2008 Contact Information

Program Chair: Gregor Kiczales, University of British Columbia, papers@oopsla.org
Conference Chair: Gail E. Harris, Instantiated Software Inc., chair@oopsla.org

For information, please contact:

ACM Member Services Department
1-800-342-6626 (US & Canada)
1-212-626-0500 (Global)
E-mail: info@oopsla.org

OOPSLA
2008 NASHVILLE