

---

## Doctoral Symposium

Monday, May 5

*Dick Hamlet, Portland State University, USA*  
*Mary Lou Soffa, University of Pittsburgh, USA*

The Doctoral Symposium is a forum for Ph.D. students to publicly discuss their research goals, methods, and results at an early stage in their research. The Symposium aims to provide useful guidance for completion of the dissertation research and for initiation of a research career by experienced faculty members in a supportive setting. The Symposium and ICSE also provide an opportunity for student

participants to interact with established researchers and others in the wider software engineering community. Attendance is limited to students who have applied and been accepted to this event.

Funding for the Doctoral Symposium has been graciously provided by Northrop Grumman.

---

## New Software Engineering Faculty Symposium (NSEFS 03)

*Leon J. Osterweil, University of Massachusetts, USA*

Monday, May 5

The challenge of becoming a new Assistant Professor at a research university is one that most people accept with enthusiasm and energy, but also with some trepidation. One's tenure track appointment is a signal accomplishment and the culmination of many years of hard work. But, simultaneously, it signals the start of another round of hard work and challenges that are new and often unfamiliar. Among these new challenges are establishing an independent research program, learning how to teach, learning how to mentor, and deciding how to balance career and personal life. For a new software engineering faculty member there are additional challenges such as obtaining financial support, supervising the development of software systems, and dealing with skeptical faculty colleagues from other disciplines.

The purpose of NSEFS 03 is to help new software engineering faculty members to feel more comfortable and

confident in dealing with these many challenges. NSEFS 03 will feature presentations by leading academic software engineering researchers who will provide advice and guidance based upon their personal experiences and insights into the contemporary academic software engineering community. There will also be ample time for informal and small group interactions with the presenters and other attendees.

NSEFS 03 is designed to be a service to those who are within two years of (either before or after) their initial tenure track academic appointment, although the symposium may be of significant interest and value to others. Enrollment is limited.

Funding for the New Software Engineering Faculty Symposium has been graciously provided by Nokia and BMW.

---

## Pioneers Symposium

Sunday, May 4

*Larry Votta, Sun Microsystems, USA*  
*Stuart Faulk, University of Oregon, USA*

The Pioneers Symposium will explore the question "What does it take to do Software Engineering research of enduring value?" This symposium will provide a forum in which Software Engineering's next generation has the opportunity to learn from some of the field's preeminent contributors what it takes to recognize and to do work that exhibits this kind of lasting value. The Symposium is primarily targeted to graduate students and new faculty beginning their research or careers. The speakers and attendees will explore what is needed to do research that will have lasting value or, as David Parnas has put it, "research that will still be relevant 25 years from now". E.g., What distinguishes such work?

What does a researcher need to know and do to produce such work?

The invited speakers are Barry Boehm, Victor Basili, Michael Jackson, Nancy Leveson, and David Parnas. These are intellectual pioneers, distinguished not just for their seminal work, but for blazing trails that others have continued to find worthwhile to follow.

Funding for the Pioneers Symposium has been graciously provided by Sun Microsystems and the National Science Foundation.

---

# Managing Commitments and Risks: Challenges in Agile Development:

## A Co-operative Learning Experience

Tuesday, May 6

*Jyrki Kontio, Helsinki University of Technology, Finland*  
*Jan Rydén, TietoEnator, Espoo, Finland*

*Magnus Höglund, TietoEnator, Espoo, Finland*  
*David Raffo, Portland State University, USA*  
*Craig Larman, Valtech, USA*

This co-operative learning event is entertaining and eye-opening. In it, participants experience human and organizational dimensions of software development. The purpose is to highlight and explore the challenges in commitment management and risk management in agile software development.

Many practical challenges in software engineering are not limited to technological issues. Managerial issues, communications, personnel relationships, and skills management often have a substantial impact on a software project's success. This event explores and studies these issues of software management from three specific perspectives:

how commitments are made and managed, how risks are managed, and what specific challenges face the agile software development context.

The event is based on interactive discussion and involves role-playing. All participants take part in working groups in solving the problem that is presented to them. The participants all take roles as consultants to a company ("Mobile Widget Inc."). Each team will be given a problem scenario and their task is to prepare a presentation for the Mobile Widget Inc. executives on how they plan to solve the problem.

---

## Workshops

Funds to support student participation at workshops have been graciously provided by IBM.

### **W1: SELMAS'03 - 2nd International Workshop on Software Engineering for Large-Scale Multi-Agent Systems**

**Saturday, May 3 - Sunday, May 4**

*Jose Alberto R. P. Sardinha, PUC-Rio - Brazil*

*Alessandro Garcia, PUC-Rio - Brazil*

*Carlos José P. de Lucena, PUC-Rio - Brazil*

*Alexander Romanovsky, University of Newcastle, Australia*

*Donald Cowan, University of Waterloo, Canada*

*Paulo S.C. Alencar, University of Waterloo, Canada*

*Jaelson F. B. Castro, Universidade Federal de Pernambuco - UFPE*

Advances in networking technology have revitalized the investigation of agent technology as a promising paradigm to engineer complex distributed software systems. Nowadays, agent technology has been applied in a wide range of application domains, including e-commerce, human-computer interfaces, telecommunications, and concurrent engineering. Software agents can be viewed as complex objects with an attitude. Like objects, agents provide a specific set of services for their users. However, software agents are inherently more complex abstractions, so that development of multi-agent systems (MASs) poses other challenges to software engineering. A single agent is driven by beliefs, goals, plans, and various behavioral properties, such as autonomy, adaptation, interaction, collaboration, learning and mobility. Each of these features adds complexity to system modeling, design and implementation. In addition, a large-scale MAS encompasses multiple types of agents, each of them having distinct agency properties, and it needs to satisfy multiple stringent requirements, such as reliability,

security, adaptability, interoperability, scalability, maintainability, and reusability. Many existing agent-oriented solutions are far from ideal; in practice, they are often built in an ad-hoc manner and are error-prone, not scalable, and not generally applicable to large MAS.

The workshop will cover a wide range of topics of software engineering for large-scale multi-agent systems, from theoretical foundations to empirical studies.

### **W2: EDSER-5 - 5th International Workshop on Economics-Driven Software Engineering Research: The Search for Value in Engineering Decisions**

**Saturday, May 3 - Sunday, May 4**

*Jyrki Kontio, Helsinki University of Technology, Finland*

*Shawn Butler, Carnegie Mellon University, USA*

The EDSER-5 workshop aims at improving the state-of-the-art and state-of-practice in economics and business driven software engineering in order to ensure that the engineering decisions made during software development fully address the relevant economic and business issues. While these technical engineering decisions form the foundation of software engineering, these decisions should be guided by the objective of adding value to stakeholders.

The workshop will give an up-to-date view of the recent research in this area and it will also act as a forum where practitioners can exchange views with leading researchers.

---

# Workshops

## **W3: CBSE6 - 6th Workshop on Component-Based Software Engineering: Automated Reasoning and Prediction** **Saturday, May 3 - Sunday, May 4**

*Ivica Crnkovic, Mälardalens University, Sweden*

*Heinz Schmidt, Monash University, Australia*

*Judith Stafford, Tufts University, USA*

*Kurt C. Wallnau, CMU Software Engineering Institute, USA*

Component-based technologies and processes have been deployed in many organizations and fields over the past several years. Measurable gains in design flexibility and development productivity have been demonstrated where software component technology has been combined with software architecture and product line practices. However, modeling, reasoning about, and predicting component and system properties remains challenging in theory and in practice. CBSE6 builds on previous workshops in the ICSE/CBSE series, this year thematically centered on automated composition theories. Composition theories support reasoning about, and predicting, the runtime properties of assemblies of components.

In this workshop emphasis is placed on composition theories that are well founded theoretically, automated by tools, and/or supported by evaluation. Both empirical and formal theories of composition are of interest. Issues related to composition theory and practice include: determining what properties are of interest in a given domain, how to model these properties, how to reason about and with property models, how to measure component and assembly properties, how to verify the measurements and predictions, and how to communicate property values and composition theories to component users. Resolving these issues requires collaborative work of researchers in software engineering, computer science, and other disciplines. The primary goal of CBSE6 is to achieve better understanding of the state of the art in automated compositional reasoning and prediction. While emphasizing state of the art, the workshop aims at bridging theory and practice.

## **W4: Bridging the Gaps Between Software Engineering and Human-Computer Interaction**

**Saturday, May 3 - Sunday, May 4**

*Rick Kazman, CMU Software Engineering Institute, USA*

*Len Bass, CMU Software Engineering Institute, USA*

*Jan Bosch, University of Groningen, The Netherlands*

Almost half of the software in systems being developed today and over a third to fifty percent of the effort expended throughout the software life cycle relate to the systems' user interface. For this reason, problems and methods from the field of human-computer interaction (HCI) have tremendous impact on the overall process of software engineering (SE), and vice versa. Yet despite these powerful reasons to practice

and apply effective SE and HCI methods, there still exist major gaps of understanding both between suggested practice and actual practice and between the best practices of the two fields. The respective curricula seldom reference each other, and do not teach the respective communities how to interact with one another. The architectures, processes, methods and vocabulary being used in each community are often foreign to the other community. As a result, product quality is not as high as it could be, and (avoidable) re-work is frequently necessary.

This workshop will bring together practitioners and academics in the two fields in an attempt to enumerate and understand these gaps between SE and HCI, with an eventual goal of proposing practical ways-shared processes, shared architectures, shared notations, etc.-to bridge these gaps. The workshop will focus on: increasing awareness of the issues in the world at large; designing joint or related curricula; creating unified tools, methods, and processes; and influencing regulations and/or conventions of practice. The tangible results of the workshop will be a practical program of education, research, and public relations focused on changing the way that people think about these two fields, and the way that the fields are actually practiced.

## **W6: Workshop on Software Variability Management** **Saturday, May 3**

*Peter Knauber, Mannheim University of Applied Sciences, Germany*

*Jan Bosch, University of Groningen, The Netherlands*

In a variety of approaches to software development, software artifacts are used in multiple contexts or for various purposes. The differences lead to so-called variation points in the software artifact. During recent years, the amount of variability that has to be supported by a software artifact is growing considerably and the management of variability is developing as a main challenge during development, usage, and evolution of software artifacts.

So far, variability management is recognized as a crosscutting concept that has a key role in various areas but it is poorly understood as an issue in its own right. In different facets, variability management is part of many recent development approaches, including but not limited to object-oriented frameworks, design patterns, domain-oriented languages, generative programming, generic components, domain and requirements analysis, and software product families (also called software product lines).

Successful management of variability in software artifacts leads to more customizable software products that are in turn likely to result in higher market success: in the information systems domain, the products are more easily adaptable to

---

## Workshops

the needs of different user groups; in the embedded systems domain, the software can be more easily configured to work with different hardware and environmental constraints.

This workshop will explicitly address the management of variability in software artifacts from its different perspectives, bringing together representatives from different academic and industrial communities to share their experiences and ideas.

### **W7: WADS - Workshop on Software Architectures for Dependable Systems**

**Saturday, May 3**

*Rogério de Lemos, University of Kent at Canterbury, UK  
Cristina Gacek, University of Newcastle upon Tyne, UK  
Alexander Romanovsky, University of Newcastle upon Tyne, UK*

Architectural representations of systems have been shown to be effective in assisting the understanding of broader system concerns by abstracting away details of the system. Dependability has become an important aspect of computer systems since everyday life increasingly depends on software. Although there is a large body of research in dependability, architectural level reasoning about dependability is only just emerging as an important theme in software engineering. The complexity of emerging applications and the trend of building trustworthy systems from existing, untrustworthy components are urging that dependability concerns be considered at the architectural level.

This workshop will bring together members of the communities of software architectures and dependability to discuss the state of research and practice when dealing with dependability issues at the architectural level.

### **W8: SCESM - 2nd International Workshop on Scenarios and State Machines: Models, Algorithms, and Tools**

**Saturday, May 3**

*Sebastian Uchitel, Imperial College, UK  
Francis Bordeleau, University of Carleton, Canada  
Alexander Egyed, Teknowledge Corporation, USA  
Martin Glinz, University of Zürich, Switzerland  
Jeff Kramer, Imperial College, UK  
Ingolf Krüger, University of California at San Diego, USA  
Axel van Lamsweerde, University of Louvain, Belgium  
Stefan Leue, University of Freiburg, Germany  
Wilhelm Schäfer, University of Paderborn, Germany  
Tarja Systä, University of Tampere, Finland  
Jon Whittle, NASA Ames, USA  
Albert Zündorf, University of Braunschweig, Germany*

Behavior modeling plays an important role in the engineering of software-based systems; it is the basis for systematic approaches to requirements capture, specification,

design, simulation, code generation, testing, and verification. A range of notations, techniques and tools supporting behavior modeling for these development tasks exists.

Two complementary approaches for modeling behavior have proven useful in practice: state- and scenario-based modeling. UML statecharts have become popular as a technique for describing the intended behavior of class instances in object-oriented systems. State-based formalisms are also widely used for modeling distributed and real-time systems, in particular because the corresponding models can be rigorously analyzed using model checking. Practitioners also use scenario-based notations and tools extensively; here, the focus of concern shifts from the complete behavior specification for individual components or objects to the (partial) specification of component collaboration.

Use cases, interaction and sequence diagrams play an important role in scenario-based requirements elicitation. The International Telecommunication Union message sequence chart standard defines a scenario-notation for detailed specification of telecommunication system behavior. The use of state machines and scenarios, however, is not limited to capturing intended system behavior; reverse engineering techniques use them extensively to capture the behavior of existing sub-systems. Although there has been much research on both scenarios and state machines the relation between them has yet to be fully understood and, more importantly, exploited. The complementary nature of scenarios and state-machines suggests several avenues for combining the strengths of both modeling approaches. Scenarios can, for instance, be viewed as partial descriptions that are generalized through state machine specifications. Alternatively, scenarios can be thought to provide collaboration views while state machines stress local/component views. Finally, scenarios can be seen as use case realizations that aid in recognizing the operations and associations of classes and in specifying the behavior of objects as state machines. Scenarios can also be viewed as the source of test-cases, used to validate an implementation.

This second workshop on scenarios and state machines has been motivated by the very successful first workshop on this topic at ICSE'02. Exploring the relation between scenarios and state machines can lead to new areas of research and to tools that can exploit the best of both worlds.

---

## Workshops

### **W9: 3rd Workshop on Open Source Software Engineering Saturday, May 3**

*Joseph Feller, University College Cork, Ireland  
Brian Fitzgerald, University of Limerick, Ireland  
Scott A. Hissam, CMU Software Engineering Institute, USA  
Karim Lakhani, MIT Sloan School of Management, USA*

Building on the success of the 1st and 2nd Workshops on Open Source Software Engineering (ICSE'01 and ICSE'02), the 3rd Workshop on Open Source Software Engineering will bring together researchers and practitioners for the purpose of discussing the diverse array of techniques, as well as supporting tools and social/organizational contexts, that can be observed in the domain of open source software (OSS).

Despite the constantly growing body of research on OSS, our knowledge of the open source family of software engineering processes is far from complete, a point raised repeatedly by the participants of the 2nd workshop in this series. To address this knowledge gap, this workshop invites participants to bring to the conversation empirical descriptions and contribute to informed discussions of the key tools, techniques, and contexts currently used by OSS development communities. Discussions will address the processes that are involved in critical software engineering activities, such as requirements gathering, analysis and design, leadership and decision making; coordination and collaboration; configuration management; testing and quality assurance; release management; debugging; documentation and support; translation and localization.

### **W10: SEHAS'03 - Software Engineering for High Assurance Systems: Synergies between Process, Product, and Profiling Friday, May 9 - Saturday, May 10**

*Martin Feather, Jet Propulsion Lab, USA  
Constance Heitmeyer, Naval Research Lab, USA  
Nancy Mead, CMU Software Engineering Institute, USA  
Allen Nikora, Jet Propulsion Lab, USA*

This two-day workshop will bring together researchers and practitioners interested in the development of an engineering method for constructing and evaluating software for high assurance systems. A high assurance system is a system where compelling evidence is required that the system delivers its services in a manner satisfying certain critical properties, such as security, safety, fault-tolerance, and survivability. Examples of high assurance systems include safety-critical medical systems, control systems for nuclear plants, and aerospace systems. The workshop participants will explore the opportunities for, and benefits of, synergies between different themes, each addressing aspects of the problem of high assurance software development. These themes are: Process, Product, and Profiling.

The purpose of this workshop is to find synergies between the themes and discover where crossover work can lead to advances that might otherwise go unexplored.

### **W11: ACSE 2003 - 3rd International Workshop on Adoption-Centric Software Engineering Friday, May 9**

*Robert Balzer, Teknowledge Corporation, USA  
Jens Jahnke, University of Victoria, Canada  
Marin Litoiu, IBM Canada Ltd., Canada  
Hausi A. Müller, University of Victoria, Canada  
Dennis B. Smith, CMU Software Engineering Institute, USA  
Margaret-Anne Storey, University of Victoria, Canada  
Scott R. Tilley, Florida Institute of Technology, USA  
Ken Wong, University of Alberta, Canada*

Understanding adoption of software engineering tools and practices is critical for the software and information technology sectors, which are continually challenged to increase their productivity. This workshop will bring together researchers and practitioners who investigate innovative solutions to software engineering adoption issues. The key objective of this workshop is to explore approaches where software engineering tools and practices are implemented as extensions of Commercial Off The Shelf Software (COTS) products and middleware technologies that work in conjunction with software engineering tools as well as mined components. The workshop aims to advance the understanding and evaluation of adoption of software engineering tools and practices.

### **W12: RAMSS - Workshop on Remote Analysis and Measurement of Software Systems Friday, May 9**

*Alessandro Orso, Georgia Institute of Technology, USA  
Adam A. Porter, University of Maryland, USA*

The manner in which software is produced and used is changing radically. Not long ago, software systems had only a few users and ran on a limited number of mostly disconnected computers. Today the situation is quite different. Nowadays the number of software systems, computers, and users has dramatically increased. Moreover, most computers are connected through the Internet.

This situation has opened the way for new development paradigms, such as the open-source model, shortened development lead times, and spurred the development and acceptance of increasingly distributed, heterogeneous computing systems.

Although these changes raise new issues for software engineers, they also represent new opportunities to greatly improve the quality and performance of software systems. Consider, for example, software analysis and measurement tasks such as testing and performance optimization. Usually,

---

## Workshops

these activities are performed in-house, on developer platforms, using developer-provided inputs, and at great cost. As a result, these activities often do not reflect actual in-the-field performance, which ultimately leads to the release of software with missing functionality, poor performance, errors that cause in-the-field failures and, more generally, users' dissatisfaction.

This workshop will bring together researchers and practitioners interested in exploring how the characteristics of today's computing environment (e.g., high connectivity, substantial computing power for the average user, higher demand for and expectation of frequent software updates) can be leveraged to improve software quality and performance. In particular, the workshop will discuss how software engineers can shift substantial portions of their analysis and measurement activities to actual user environments, so as to leverage in-the-field computational power, human resources, and user data to investigate the behavior of systems after deployment and to improve quality and performance.

### **W13: GSD 2003 - International Workshop on Global Software Development Friday, May 9**

*Daniela Damian, University of Victoria, BC Canada  
Filippo Lanubile, University of Bari, Italy  
Heather L. Oppenheimer, Lucent Technologies, USA*

Increased globalization of software development creates software engineering challenges due to the impact of temporal, geographical and cultural differences, and requires development of techniques and technologies to address these issues. The goal of this workshop is to provide an opportunity for researchers and industry practitioners to explore both the state-of-the-art and the state-of-the-practice in global software development.

The workshop is a continuation of the last five ICSE workshops on the same topic. Our workshop at ICSE last year, after changing the title from the technology-focused "Software Engineering over the Internet" to the more general

"Global Software Development," generated increased interest from software practitioners and fruitful discussions between industry and academia.

This year at ICSE'03, the workshop will continue and draw upon this interaction between academia and industry in addressing the issues of Global Software Development.

### **W14: WODA 2003 - Workshop on Dynamic Analysis Friday, May 9**

*Jonathan E. Cook, New Mexico State University, USA  
Michael D. Ernst, Massachusetts Institute of Technology, USA*

This workshop will focus on sharing ideas on and brainstorming new approaches to effective dynamic analysis. It will cover a topical spectrum, possibly including enabling technologies; framework and common tool support; event type definition, classification, and specification; symbolic and theoretically exact reasoning techniques; statistical and probabilistic reasoning techniques; research foundations; relationships to static analysis; relationships to testing; and other potential topics.

### **W15: STRAW '03 - 2nd International Workshop on Software Requirements to Architectures Friday, May 9**

*Daniel M. Berry, University of Waterloo, Canada  
Roel Wieringa, University of Twente, The Netherlands  
Rick Kazman, CMU Software Engineering Institute, USA*

There is a clear relationship between requirements engineering and architecture design in software engineering. However, for the most part, the two disciplines have evolved independently from each other, and promising areas of mutual interest remain to be explored. The Second International Workshop on Software Requirements and Architectures (STRAW '03) will bring together researchers from the requirements engineering and architecture communities to exchange views and results that are of mutual interest, and to discuss topics for further research.

---

# Tutorials

Funds to support student attendance at tutorials have been graciously provided by Microsoft Research and Daimler Chrysler.

## **F3: Documenting Software Architectures: Views and Beyond** **Sunday, May 4**

*Paul Clements, CMU Software Engineering Institute, USA*  
*David Garlan, Carnegie Mellon University, USA*  
*Reed Little, CMU Software Engineering Institute, USA*  
*Robert Nord, Siemens Corporate Research, USA*  
*Judith Stafford, Tufts University, USA*

A documented software architecture supports communication among system stakeholders such as managers, marketers, engineers, and maintainers. It serves as a base for education of members of the development team, system analysts, and newly hired developers, perhaps even new architects. It also serves as a base for early analysis of how well the system will fulfill behavioral and quality requirements. It is imperative to know how to write documentation of an architecture so that interested parties can use it, maintain it, and build a system from it. Software projects increase the risk of failure to meet schedules, behavioral requirements, and/or quality goals without a blueprint that is clear, concise, and useful.

This full-day tutorial presents the “Views and Beyond” approach for software architecture documentation developed at the Software Engineering Institute. The approach is based on the simple principle that documenting an architecture is a matter of choosing and then documenting the relevant views of that architecture, and then documenting information that applies across all the views.

The primary aim of this tutorial is to teach software architects and developers how to write down a software architecture for a software system. Secondary aims include teaching what constitutes good documentation of a software architecture and why it is worth the effort to create and maintain architecture documentation.

## **F4: Computing System Dependability** **Sunday, May 4**

*John Knight, University of Virginia, USA*

Computer systems provide us with a wide range of services upon which we have come to depend. Many computers are embedded in more complex devices that we use such as automobiles, aircraft, appliances, and medical devices. Others are part of sophisticated systems that provide us with a variety of important facilities such as financial services, telecommunications, transportation systems, and energy production and distribution networks. Without these computer systems, many devices and services that are important either would not operate or would be much less useful. In some cases, the failure of a computer system can lead to extensive damage. This is a full-day tutorial at the introductory level that will familiarize computer engineers, software engineers, managers, and researchers with the basic elements of the

theory and practice of computing system dependability. The fundamental principles and basic concepts of dependability will be presented along with various dependability analysis techniques. Several practical techniques that help improve dependability will be introduced.

Fundamental topics covered will include the basic principles and definitions, types of fault, fault avoidance, fault elimination, and fault tolerance, system hazard analysis, event-tree analysis, failure modes and effects analysis, fault-tree analysis, software specification and verification, Byzantine agreement, fail-stop machines, real-time systems, and distributed systems.

Attendees will acquire a familiarity with the field so that they will be able to make appropriate choices in selecting techniques to apply within their own work and in their own future study. Attendees will be assumed to have a working knowledge of basic computer organization, programming in a high-level language, and the software development lifecycle.

## **F5: Goal-Oriented Requirements Engineering: From System Objectives to UML Models to Precise Software Specifications** **Sunday, May 4**

*Axel van Lamsweerde, University of Louvain, Belgium*

Requirements engineering (RE) is the branch of systems engineering concerned with the elicitation of the objectives to be achieved by the system envisioned, the operationalization of such objectives into specifications of services and constraints, the assignment of responsibilities for the resulting requirements to agents such as humans, devices and software, and the evolution of such requirements over time and across system families. Getting adequate, consistent and complete requirements is difficult and critical. Recent surveys have confirmed the growing recognition of RE as an area of primary concern in software engineering research and practice.

This full-day tutorial first briefly introduces RE by defining its scope and the products and processes involved. A multi-view framework is then presented for system modeling in the specific context of engineering requirements. The framework integrates AND/OR goal diagrams for the refinement/abstraction of functional and non-functional objectives in the current system and in the system-to-be; UML class diagrams for modeling the conceptual objects concerned by such objectives; context diagrams for modeling agents in the system-to-be together with their responsibilities and interfaces; UML use case diagrams for modeling the software services that operationalize the functional goals assigned to software agents; and UML sequence and state diagrams for requirements mining and behavior modeling. Lightweight techniques for precise specification of the goals, objects, agents and operations from these various models are also

---

## Tutorials

discussed briefly. The next part of the tutorial is devoted to RE-specific model analysis; in particular, we review techniques for goal refinement, operationalization and assignment; analysis of obstacles to goal satisfaction for more robust system building; and management of conflicting goals among multiple viewpoints. The final part of the tutorial puts all previous pieces together by presenting a constructive method for elaborating a full, robust and consistent system model from which precise software specifications are derived. The presentation will strongly rely on running examples in two different domains: an information system and an embedded control system. The method and associated tool will be shown in action on a system simplified from a real project.

This full-day tutorial is aimed at both industrial and academic participants who wish to get a comprehensive overview of state-of-the-art techniques available in this critical area. No specific prerequisite is required. Tutorial attendants will get deep insights on the specific problems raised when engineering high-quality requirements, have a good sense of the range of current technical solutions to these problems, and increase their practical ability to build and analyze complex system models and software specifications.

### **F6: Mastering Design Patterns** **Sunday, May 4**

*Craig Larman, Valtech, USA*

During both initial software development and subsequent modification, a major cost that is often not appreciated is the quality of the design. These include the scalability, ease of comprehension, flexibility, and robustness in the face of change. Various studies indicate that after initial release, at least 50% of effort and cost is spent in modification. To save money, it is rational to take a long-term view of product development and invest in skillful designs that reduce these costs. This tutorial will provide designers with the skills needed to create high-quality object-oriented designs that support reduced modification costs and increased comprehensibility.

Design patterns are about the reuse of excellent, established design ideas, rather than code- best-practice formulas from experienced object-oriented developers. There are well-established, popular and useful design patterns, such as the "gang-of-four" (GoF) design patterns described in the hugely popular and influential book "Design Patterns," the annual PLOP workshop patterns, and other sources, such as the Pattern-Oriented Software Architecture series. In this information-intensive tutorial, participants will learn how to apply the majority of high-frequency GoF design patterns.

By attending this tutorial, you will learn to apply patterns for: varying algorithms, including Strategy; varying instance behavior, including Decorator; access control, including Façade and Singleton; organization of work, including

Command; structural decomposition, including Composite; varying event response, including Observer; varying interfaces, including Adapter; varying instance creation, including Abstract Factory; varying implementations, including Bridge; and more, including coverage of idioms for package design.

This full-day tutorial is intended for object-oriented developers and architects with at least six months of object-oriented programming experience.

### **F7: VBE-Value-Based Software Engineering** **Monday, May 5**

*Barry Boehm, University of Southern California, USA*  
*Kevin Sullivan, University of Virginia, USA*

Attendees of this full-day tutorial will gain an appreciation for the increasing importance of value-based over value-neutral software engineering methods, an understanding of the major Value-Based Software Engineering (VBSE) concepts and techniques, and experience in applying the concepts and techniques to a representative software engineering case study. The tutorial is intended for software engineering practitioners and managers.

The tutorial will cover the following topics: participants' summary of their learning objectives (or value propositions) for the tutorial; an overview of VBSE techniques, and how they address current and emerging software engineering challenges, opportunities, and problems areas; a VBSE case study of a single-thread Internet-based order processing system; seven key elements of VBSE, including benefits realization analysis; stakeholders' value proposition elicitation and reconciliation; cost/benefit and business case analysis, continuous risk and opportunity management, concurrent software and system engineering, value-based monitoring and control, and change as opportunity.

Additional topics include life cycle processes for implementing VBSE; a VBSE research roadmap and summary of recent research results; and a summary of emerging new research results which to exploit tools such as real options theory, portfolio theory, and statistical decision theory to understand the value provided by alternative software process and product decisions under conditions of uncertainty and change. These emerging methods can also help determine the value of buying information (via prototyping, benchmarking, COTS evaluation, architecture analysis, testing, etc.) to reduce risk in uncertain decision situations.

### **F8: Usage-Centered Software Engineering** **Monday, May 5**

*Larry Constantine, Constantine and Lockwood, Ltd., USA*  
*Lucy Lockwood, Constantine and Lockwood, Ltd., USA*

Software engineering is often weakest when it comes to addressing the critical areas of user requirements, usability, user interfaces, and interaction design. Often these concerns

---

## Tutorials

are minimized, ignored, or relegated to other disciplines and other professionals as a responsibility outside the scope of software engineering. This tutorial for practicing software engineering, design, and development professionals and technically oriented managers shows how user requirements, usability, and user interface design can be made the very core of a successful software engineering process.

Usage-centered design is a systematic, model-driven approach to visual and interaction design with an established record of effectiveness over a wide range of project scopes within a wide variety of settings and areas of application. This full-day tutorial will introduce the models and methods of usage-centered design and explore the integration of usage-centered approaches into software engineering practice. Agile approaches to modeling will be stressed, with the focus on use cases, which are central to usage-centered design and serve as a common thread throughout an integrated usage-centered software engineering process. Although emphasis is placed on agile usage-centered, this approach also serves as an effective bridge to more elaborate structured forms of usage-centered modeling.

Usage-centered design is guided by three core models of users, tasks, and user interface contents. Users are modeled by the roles they play in relation to the planned system, tasks are modeled using a specialized form of use cases known variously as task cases or essential use cases, and interface contents are modeled in the form of abstract prototypes.

The emphasis in this tutorial is on practical application and skill-building. Participants will emerge with a broad understanding of usage-centered design and how it can be incorporated into the overall software engineering process. They will gain experience in modeling users, tasks, and user interfaces employing simplified agile modeling techniques to quickly organize information about users and user tasks into concise models for driving the design and constructing object-oriented software.

### **F10: Pattern-Oriented Distributed System Architectures Monday, May 5**

*Doug Schmidt, Vanderbilt University, USA*

Developing software for distributed systems that effectively utilizes concurrency over high-speed, low-speed, and mobile networks is hard. This full-day tutorial describes how to apply architectural and design patterns together with middleware frameworks and components to alleviate the complexity of developing concurrent distributed systems. These patterns, frameworks, and components have been used successfully by the speaker and others on actual distributed systems at hundreds of commercial companies in domains ranging from datacom and telecom, medical engineering, aerospace, defense, distributed interactive simulations, and financial services. The tutorial examines

reusable solutions abstracted from production systems to illustrate the key technical design and implementation issues.

Attendees of this tutorial will learn, from examples, how to significantly simplify and enhance the development of middleware and application software that effectively utilizes concurrency and distribution via the use of advanced design techniques — such as patterns, layered, modularity, and data/control abstraction; object-oriented language features — such as abstract classes, inheritance, dynamic binding, and parameterized types; middleware — such as object-oriented frameworks for infrastructure middleware (e.g., ACE and JVMs) and distribution middleware (e.g., J2EE, .NET, and CORBA ORBs); and advanced operating system mechanisms — such as event demultiplexing, multi-threading, multi-processing, and explicit dynamic linking.

The material presented in this tutorial is based in part on the book “Pattern-Oriented Software Architecture: Patterns for Concurrent and Distributed Objects”, Wiley & Sons, 2000, which is the second volume in the Pattern-Oriented Software Architecture (POSA) series co-authored by Frank Buschmann, Hans Rohnert, and Michael Stal.

### **F11: Industrial-Strength Software Product-Line Engineering Monday, May 5**

*John Klein, Avaya CRM, USA*

*Barry Price, Avaya Labs, USA*

*David Weiss, Avaya Labs, USA*

Software product-line engineering is one of the few approaches to software engineering that shows promise of improving software productivity by factors of 5 to 10. However, there are still few examples of successful application of product-line engineering on a large scale partly because of the complexity of the problem of initiating a product-line engineering project and of the many factors that must be addressed for such a project to be successful. Practitioners and researchers alike are confronted with issues such as the following: How have others been able to succeed in creating and maintaining a software product-line? What approaches have been tried? How do I construct a convincing business case for product-line engineering? Whom do I have to convince? What are the key activities needed to start a product-line engineering project and to have it be successful? How do I recognize when my organization is ready to or needs to start product-line engineering? How do I measure the effectiveness of product-line engineering in my organization? How long will it take me to show some return? What does a product-line organization look like? How is it different from what my organization looks like now? What resources will I need? What training will my people need? This one-day tutorial draws on experiences in introducing and sustaining product-line engineering in Lucent Technologies and in Avaya to answer such questions.

Participants will gain an inside look at an organization that is transitioning into product-line engineering on a large

---

# Tutorials

scale. Success in such a project depends on both technical issues in creating and evolving a product-line, and organizational issues in transforming an organization to be compatible with the product-line engineering process. Participants will be shown by example what events and actions must take place at what times to be successful in introducing product-line engineering. We expect attendees to leave the tutorial with many ideas on how they can start introducing product line engineering into their organizations in a systematic way.

## **H1: Feature-Oriented Programming for Product Lines** **morning, Monday, May 5**

*Don Batory, University of Texas at Austin, USA*

Feature Oriented Programming (FOP) includes a design methodology and a set of tools for program synthesis. The goal of FOP is to specify a target program in terms of the features that it offers, and to synthesize an efficient program that meets these specifications. FOP has been used to develop product-lines in disparate domains, including compilers for extensible Java dialects, fire support simulators for the U.S. Army, high-performance network protocols, and program verification tools.

GenVoca is a simple mathematical model of FOP that is based on step-wise refinement, a methodology for building programs by adding one feature at a time. The incremental units of implementation/design are refinements that encapsulate the implementation of an individual feature. GenVoca models of product-lines treat programs as values and refinements as functions (that map input programs to output programs with augmented features). Application designs are equations - compositions of functions and values - that are amenable to optimization and analysis. FOP and Aspect Oriented Programming (AOP) are complementary, as both aspects and feature refinements encapsulate cross-cuts, i.e., fragments of multiple classes. The primary difference is emphasis: FOP follows a more traditional OO design approach to define cross-cuts, which focuses on how algorithms compose to build complete systems, rather than the AOP emphasis on join-points, point-cuts, and advice to modify existing programs.

This half-day tutorial focuses on AHEAD (Algebraic Hierarchical Equations for Application Design), the successor to GenVoca. AHEAD allows a single refinement to encapsulate simultaneous changes to multiple programs (e.g., tool suites like MS Office where individual features impact or cross-cut multiple tools) and changes to arbitrary program representations (e.g., makefiles, documents, code, etc.). Thus, by composing AHEAD refinements, consistent tool suites and their plethora of representations can be synthesized. AHEAD significantly broadens the capabilities of FOP and provides the basis for scaling system synthesis far beyond current technologies.

Attendees of this half-day tutorial will learn the basic concepts of FOP, models and tools for synthesizing a

consistent set of code and non-code artifacts by composing refinements (cross-cuts), automatic algorithms for validating refinement compositions, synthesis of product-lines for product-families (e.g., tool suites), and automatic algorithms for optimizing application designs (equations).

## **H2: An Overview of UML 2.0** **morning, Monday, May 5**

*Bran Selic, Rational Software, Canada*

Since its adoption as an industry standard in 1997, the Unified Modeling Language (UML) has been adopted widely by both industry and academia. This extensive experience has naturally led to demands for improvements and new capabilities. In September 2000, the Object Management Group-the industrial consortium that controls the UML standard-issued a request for proposal for the first major revision of UML, UML 2.0. This new version was conceived as the basis for the coming generation of model-based development methods, such as OMG's Model-Driven Architecture (MDA). The distinguishing characteristic of these methods is that their primary focus is on the definition and evolution of models rather than programs-with programs being automatically generated from such models. The combination of higher-level abstractions defined in UML and the use of automation provide the potential for a dramatic improvement in productivity and software reliability.

Attendees of this half-day tutorial will learn the salient aspects of UML 2.0-from the perspective of one of its primary authors. The following specific topics will be covered: a general and critical discussion of the effectiveness of modeling techniques in software engineering; a critical review of the original UML standard based on the lessons learned from its application as well as from theoretical studies; the formal and informal requirements that drove development of the new version of UML (this includes an introduction to model-driven development methods); the overall structure and design philosophy behind UML 2.0; the conceptual foundation, or "infrastructure" used to define the UML modeling concepts and their semantics; new modeling capabilities for rendering software structures such as those required for specifying software architectures; new modeling capabilities for describing complex behavior of individual objects and groups of collaborating objects; changes to existing UML concepts; and the application of UML 2.0 in model-driven software development methods based on executable models and the use of automatic code generation. Each of these concepts will be extensively illustrated with numerous practical examples.

## **H3: Best Practices for Implementing** **CMM-Based Software Process Improvement** **morning, Monday, May 5**

*Bill Curtis, TeraQuest, USA*

This half-day tutorial will provide an explanation of the principles underlying the CMM and CMMI, as well as

---

## Tutorials

guidance for choosing between them. The areas of focus in the tutorial include:

**The Process Maturity Framework** — The principles and theoretical foundation underlying all CMMs will be described. Participants will come to see CMMs as more than just process standards, and rather as a way to develop an organization's core competence in delivering software. The CMM for Software and CMMI will be described and guidance will be provided for selecting between them for use in guiding process improvement programs. Future directions for the process maturity framework will be described, since it is now guiding the creation of CMMs for business domains well beyond software and systems development.

**Software Process Improvement Benefits** — This section will describe the benefits companies have reported from conducting improvement programs. These benefits will be reported from both aerospace and commercial companies in a range of business domains. The benefits experienced differ by maturity level and the tutorial will provide expectations about what types of benefits can be observed at different levels and how much improvement might be expected.

**Software Process Improvement Programs** — This section of the tutorial will describe best practices for conducting improvement programs and lessons learned from 10 years of improvement programs. A project-focused improvement method will be described that has demonstrated good success and avoided some of the pitfalls that have been known to undermine improvement programs. Risk factors for improvement activities will be described and the critical differences in running improvement programs at different levels of maturity will be described.

**Tailoring Improvement Programs for Agile Environments** — This section will describe the tailoring of CMM-based software process improvement programs with software methods such as agile methods and the Rational Unified Process. There are fewer disagreements between the CMM and agile methods than many have characterized. However there are some differences that will be discussed and some guidelines for integrating CMM-based improvements in an environment attempting to implement agile methods will be presented.

Participants in this tutorial will learn of the benefits that have been achieved and how they differ by maturity level; lessons learned and best practices for implementing process improvement programs; and observations on integrating process improvement programs with agile methods.

### **H4: Designing Software Architectures for Usability afternoon, Monday, May 5**

*Jan Bosch, University of Groningen, The Netherlands  
Natalia Juristo, Technical University of Madrid, Spain*

Usability is increasingly recognized as a quality attribute that one has to design for. The conventional alternative is to measure usability on a finished system and improve it. The disadvantage of this approach is, obviously, that the cost associated with implementing usability improvements in a fully implemented system are typically very high and prohibit improvements with architectural impact.

In this half-day tutorial, we present the insights gained, techniques developed and lessons learned in the EU-IST project STATUS (Software Architectures That supports Usability). Attendees will learn about a forward-engineering perspective on usability, a technique for specifying usability requirements, a method for assessing software architectures for usability and, finally, techniques for improving software architectures for usability. The topics are extensively illustrated by examples and experiences from many industrial cases.

### **H6: Agile Unified Process afternoon, Monday, May 5**

*Craig Larman, Valtech, USA*

The Unified Process (UP) is popular-and with good reason, as it includes skillful practices such as iterative development, and early attention to risks. However, many organizations are adopting it in an unskilled manner, superimposing "waterfall" or "heavyweight" values onto it that need not apply. Applied well, the UP encourages an agile approach; furthermore, many XP and Scrum practices are either part of the UP, or specializations of more general UP guidelines.

In this half-day tutorial, you will learn why a combination of UP with other agile process practices is an excellent approach. Topics to be presented include: motivation and the business case for the UP; the key UP ideas to know and now to apply them; UP artifacts that are worth creating; UP anti-patterns (common worst-practices in adoption and use); how to adopt the UP within an organization; XP practices within the UP; challenges of pure UP; how to plan an iterative UP project; tips for good UP artifacts and models; UP models and the UML; how to define a UP development case; how to do architectural analysis and describe architectures in the UP; and how to fail with the UP (you know you didn't understand it when...)

Participants in this tutorial will learn the essential UP practices, the keys to successful introduction of the UP in an organization, and how to apply the UP in an adaptive and agile manner. And, how to screw it up.

---

## Panels

### Panel 3.4X: Automotive Software Engineering Tuesday, May 6

#### Panel Chair:

*Manfred Broy, Technical University Munich, Germany*

#### Panelists:

*Ulrich Weinmann, BMW Car IT GmbH, Germany  
Klaus Grimm, DaimlerChrysler AG, Germany  
Michael Reinfrank, Siemens VDO Automotive AG, Germany  
Erich Nickel, IBM Telematics Solution, Germany*

This panel gathers leading experts on automotive software development from carmakers, suppliers and research organizations in the US and Europe. These experts will discuss opportunities, trends and demands for software-related technologies, architectures, methods and tools to develop and integrate software-based car functions. Through prepared and audience-based questions and the ensuing discussion, we hope to raise awareness on Automotive Software Engineering in the software community and stimulate contributions in automotive software research and development.

### Panel 4.3: Empirical Evaluation: What, Why, When, Where Tuesday, May 6

#### Panel Chair:

*Rob Walker, University of Calgary, Canada*

#### Panelists:

*Lionel C. Briand, Carleton University, Canada  
David Notkin, University of Washington, USA  
Carolyn B. Seaman, University of Maryland, Baltimore  
County and Fraunhofer, USA  
Walter F. Tichy, Universität Karlsruhe, Germany*

Opinions as to the methodology to apply to validate SE research appear to be in disagreement. Without some consensus, an SE researcher is faced with a difficult task of convincing their peers that their selected methodology is appropriate, let alone the details of their validation. Topics to be considered in this panel session include: quantitative vs. qualitative validation; studying the world as it is vs. how it would be with a given technology introduced; control vs. analyzability; evidence justifying further research vs. evidence justifying industrial adoption. A lively debate will be encouraged.

### Panel 6.3E: Undergraduate SE Degree -- Pros and Cons Wednesday, May 7

#### Panel Chair:

*D. Perry, University of Texas at Austin, USA*

#### Panelists:

*Don Bagert, Rose-Hulman Institute of Technology, USA  
Rich LeBlanc, Georgia Institute of Technology, USA  
Hausi Muller, University of Victoria, Canada  
Wilhelm Schaefer, University of Paderborn, Germany  
Michal Young, University of Oregon, USA*

That there is a significant need for software engineering courses to better prepare students for the reality of building and evolving software systems is beyond any doubt. What is controversial is the context of software engineering education. Is it best served in the traditional context of existing computer science or computer engineering majors, or does it need a new context of a separate undergraduate software engineering major. The question is complicated by the fact that CS and CE are often in different colleges rather than aligned together. Depending on the view taken there are then subsidiary issues such as appropriate preparation as well as appropriate course structures, etc.

### Panel 12.3: Modularity in the New Millenium: A Discussion Thursday, May 8

#### Panel Chair:

*Prem Devanbu, University of California, Davis, USA*

#### Panelists:

*Bob Balzer, Tecknowledge Corp, USA  
Don Batory, University of Texas at Austin, USA  
Gregor Kiczales, University of British Columbia, Canada  
John Launchbury, Oregon Graduate Institute, USA  
David Parnas, University of Limerick, Ireland  
Peri Tarr, IBM T.J. Watson Research Center, USA*

Since Parnas' early work on separation of concerns in design, there has been a growing sentiment in many quarters that there are some concerns that stubbornly resist tidy confinement, when using traditional modularization mechanisms in programming languages. Several new approaches have emerged, in disparate quarters: aspect-oriented programming, multi-dimensional separation of concerns, mixins and mixin layers, and monads. While these approaches arose independently, they have each developed (to varying degrees) technical maturity, found applications in the real world, and attracted strong followers. We also are now beginning to see strong scholarly comparisons of the intellectual foundations and practical utility of these different approaches. The goal of this panel is to bring together leading experts in the different areas, to support and stimulate comparative analysis of these approaches.

---

## Demonstrations and Posters

In addition to the scheduled demonstration presentations, poster and demo presenters will be available to discuss their work at the conference reception.

### Demonstrations:

#### **AGENDA Tool Set for Testing Relational Database Applications**

*David Chay, Polytechnic University, USA*  
*Yuetang Deng, Polytechnic University, USA*

Database systems play an important role in nearly every modern organization, yet relatively little research effort has focused on how to test them. AGENDA, A (test) GENERator for Database Applications, is a research prototype tool set for testing DB application programs. In testing such applications, the states of the database before and after execution play an important role, along with the user's input and system output. AGENDA components populate the database, generate inputs, and check aspects of the correctness of output and new DB state.

#### **Architecture Level Risk Assessment Tool Based on UML Specifications**

*Hany H. Ammar, West Virginia University, USA*  
*Katerina Goseva-Popstojanova, West Virginia University, USA*  
*Ahmed Hassan, West Virginia University, USA*  
*Walid Abdelmoe, West Virginia University, USA*  
*Ajith Guedem, West Virginia University, USA*  
*Tianjian Wang, West Virginia University, USA*

Our demonstration presents a prototype tool called Architecture-Level Risk Assessment Tool (ARAT) based on the risk assessment methodology we developed. The ARAT provides risk assessment based on measures obtained from Unified Modeling Language (UML) artifacts. This tool can be used in the design phase of the software development process. It estimates dynamic metrics and automatically analyzes the quality of the architecture to produce architectural-level software risk assessment.

#### **CLIME: An Environment for Constrained Evolution**

*Steven. P. Reiss, Brown University, USA*  
*Christina M. Kennedy, Brown University, USA*  
*Tom Wooldridge, Brown University, USA*  
*Shriram Krishnamurthi, Brown University, USA*

We are building a software development environment that uses constraints to ensure the consistency of the different artifacts associated with software. This approach to software development makes the environment responsible for detecting most inconsistencies between software design, specifications, documentation, source code, and test cases. The environment provides facilities to ensure that these various dimensions remain consistent as the software is written and evolves. The environment works with the wide variety of artifacts typically associated with a large software system. It handles both the

static and dynamic aspects of software. Moreover, it works incrementally so that consistency information is readily available to the developer as the system changes. The demonstration will show this environment and its capabilities.

#### **DRT: A Tool for Design Recovery of Interactive Graphical Applications**

*Keith Chan, University of New South Wales, Australia*  
*Annie Chen, University of New South Wales, Australia*  
*Zhi Cong Leo Liang, University of New South Wales, Australia*  
*Amir Michail, University of New South Wales, Australia*  
*Minh Hoai Nguyen, University of New South Wales, Australia*  
*Nicholas Seow, University of New South Wales, Australia*

DRT is a design recovery tool for interactive graphical applications running under the X Window System. The tool automatically captures actions performed while using such an application. Functions particularly relevant to each action are identified. Moreover, the action itself is described visually from a fragment of the application display. One can search and browse these actions to learn about the design of an application.

#### **Eclipse Open Source Platform**

*Jim D'Anjou, IBM Corporation, USA*

Eclipse is an open source tool integration platform. In other words, it can work seamlessly with other application development tools you need to use to get your job done. Eclipse contains a Java programming development environment that not only offers a high degree of productivity and flexibility but that also, you can say, "Plays well with others." Through the programming interfaces in Eclipse, the door is open for the integration of multiple tools. Eclipse can run on a variety of different operating systems and is language neutral. It welcomes tools for building applications in Java, C, C++, HTML, and so on. As an open source project, you get all the source code, and best of all, it's free.

#### **An Effective Layout Adaptation Technique for a Graphical Modeling Tool**

*Christian Seybold, University of Zurich, Switzerland*  
*Martin Glinz, University of Zurich, Switzerland*  
*Silvio Meier, University of Zurich, Switzerland*  
*Nancy Merlo-Schett, University of Zurich, Switzerland*

Editing graphic models always entails layout problems. Inserting and deleting items requires tedious manual work for shifting and rearranging items in the diagram layout. Existing layout generation algorithms rearrange diagrams without preserving the overall visual appearance of the diagram.

---

## Demonstrations and Posters

We have developed a technique which automatically expands or contracts a diagram layout when items are inserted or removed while preserving its overall shape, i.e. the positions of the items relative to each other. Our technique has been implemented in a prototype tool. We are using it not just for simplifying editing, but primarily for implementing an aspect-oriented visualization concept.

### **Embedded Architect: A Tool for Early Performance Evaluation of Embedded Software**

*Jeffry T. Russell, University of Texas at Austin, USA  
Margarida F. Jacome, University of Texas at Austin, USA*

Embedded Architect is a design automation tool that embodies a static performance evaluation technique to support early, architecture-level design space exploration for component-based embedded systems. A static control flow characterization, called an evaluation scenario, is specified based on an incremental refinement of software source code, from which a pseudo-trace of operations is generated. In combination with architecture mapping and several component parameters, a software performance metric is estimated.

### **FEAT: A Tool for Locating, Describing, and Analyzing Concerns in Source Code**

*Martin P. Robillard, University of British Columbia, Canada  
Gail Murphy, University of British Columbia, Canada*

The FEAT tool supports locating, describing, and analyzing the code implementing a one or more concerns in a Java system. FEAT is developed as a plugin for the Eclipse Platform. With the FEAT plugin activated, programmers can use the integrated development environment as usual, to browse and modify code, perform searches, etc. However, with FEAT, a user can query the relationships between different elements of interest (such as fields or methods), and to capture elements and relations relevant to a task in a concern representation. The demonstration will consist in performing part of a change task on a Java program with the help of FEAT.

### **Java Program Analysis Projects in Osaka University: Aspect-Based Slicing System (ADAS) and Ranked-Component Search System (SPARS-J)**

*Reishi Yokomori, Osaka University, Japan  
Takashi Ishio, Osaka University, Japan  
Tetsuo Yamamoto, Japan Science and Technology Corporation, Japan  
Makoto Matsushita, Osaka University, Japan  
Shinji Kusumoto, Osaka University, Japan  
Katsuro Inoue, Osaka University, Japan*

In our research demonstration, we show two development support systems for Java programs. One is an Aspect-oriented Dynamic Analysis and Slice calculation system

named ADAS, and another is a Software Product archiving, Analyzing, and Retrieving System for Java named SPARS-J. ADAS supports debugging tasks for Java programs based on program slicing technique, and SPARS-J provides a retrieval result with the evaluation value based on actual use relations among components.

### **JIVE: Visualizing Java in Action**

*Steven P. Reiss, Brown University, USA*

Dynamic software visualization should provide a programmer with insights as to what the program is doing. Most current dynamic visualizations either use program traces to show information about prior runs, slow the program down substantially, show only minimal information, or force the programmer to indicate when to turn visualizations on or off. We have developed a dynamic Java visualizer that provides a view of a program in action with low enough overhead that it can be used almost all the time by programmers to understand what their program is doing while it is doing it.

### **A Software Process Scheduling Simulator**

*Frank Padberg, University of Karlsruhe, Germany*

We show how to use process simulation to support software project managers in scheduling. We present a discrete-time simulator which is tailored to the dynamics of software projects and explicitly takes a scheduling strategy as input. The simulator makes it easy for a manager to experiment with different scheduling strategies for his next software project. As an example, we use the simulator to study the performance of different list policies for a small sample project. The results show that the scheduling strategy has a strong impact on the average completion time of the sample project.

### **Source Viewer 3D (sv3D) - A Framework for Software Visualization**

*Jonathan Maletic, Kent State University, USA  
Andrian Marcus, Kent State University, USA  
Louis Feng, Kent State University, USA*

Source Viewer 3D is a software visualization application-framework that uses a 3D metaphor to represent software system and analysis data. The 3D metaphor is based on the SeeSoft pixel representation. It extends the original metaphor by rendering the visualization in a 3D space. Object-based manipulation methods and simultaneous alternative mappings are available to the user.

sv3D generates specific software visualization applications based on available analysis tools. The input format and mappings are defined in a XML file. In its current form, sv3D supports mapping and representation of 6 types of data in the 3D space.

---

## Demonstrations and Posters

### **Trustworthy and Sustainable Operations in Marine Environments**

*Martin Fredriksson, Blekinge Institute of Technology, Sweden  
Rune Gustavsson, Blekinge Institute of Technology, Sweden*

In order to address challenges and opportunities of engineering information systems for network-centric warfare, we have developed a prototype for trustworthy and sustainable operations in marine environments (TWOSOME). The system developed addressed qualities such as information fusion, target acquisition, and self-organization in open computational systems; comprised of distributed services. As such, the system prototype executes on a service-oriented layered architecture for communicating entities (SOLACE) and, furthermore, different perspectives of the prototype are visualized by means of a distributed interaction system for complex entity relation networks (DISCERN).

### **xChek: A Model Checker for Multi-Valued Reasoning**

*Steve Easterbrook, University of Toronto, Canada  
Marsha Chechik, University of Toronto, Canada  
Benet Devereux, University of Toronto, Canada  
Arie Gurfinkel, University of Toronto, Canada  
Albert Lai, University of Toronto, Canada  
Victor Petrovykh, University of Toronto, Canada  
Anya Tafliovich, University of Toronto, Canada  
Christopher Thompson-Walsh, University of Toronto, Canada*

XChek is a multi-valued symbolic model checker that works for any member of a large class of multi-valued logics. Multi-valued model checking generalizes classical model checking and is useful for analyzing models where there is uncertainty (e.g. missing information) or inconsistency (e.g. disagreement between different views). Explicit modeling of uncertainty and disagreement is supported by providing additional truth values in the logic. Our modeling language is based on a generalization of Kripke structures, where both propositions and state transitions may take any truth value of a given logic. Properties are expressed in XCTL, our multi-valued extension of CTL.

### **XVCL - XML-based Variant Configuration Language**

*Stan Jarzabek, National University of Singapore, China  
Hongyu Zhang, National University of Singapore, China  
Weishan Zhang, National University of Singapore, China  
Paul Bassett, Canadian Information Processing Society  
National, Canada  
Soe Myat Swe, Singapore Engineering Software Pte Ltd,  
China*

XVCL (XML-based Variant Configuration Language) is a meta-programming technique and tool that provides effective reuse mechanisms. XVCL blends with contemporary programming paradigms and complements other design techniques. XVCL uses "composition with adaptation" rules to

generate a specific program from generic, reusable meta-components. Despite its simplicity, XVCL can effectively manage a wide range of program variants from a compact base of meta-components, structured for effective reuse. Applications of XVCL include design of reusable program components, implementing product line architectures, design of compact, non-redundant, therefore, easy to maintain programs, and managing variants in multiple versions of software documents and models.

## **Posters:**

### **Flexible Static Semantic Checking Using First-Order Logic**

*Shimon Rura, Williams College, USA  
Barbara Staudt Lerner, Williams College, USA*

We present a static semantic checker for Little-JIL that is implemented by defining language semantics and coding guidelines as declarative assertions over the structure of a program. These assertions are written in first-order logic and input to xlinkit, a constraint checker, to produce an executable semantic checker. Our checker reports problems like linguistic restrictions, potential race conditions, lost data, and infinite recursion. The checker's architecture, which keeps rules separate from verification and reporting procedures, makes it easy to extend the checker with new anomaly detection rules and to adjust existing rules if language features change.

### **Experimental Software Schedulability Estimation For Varied Processor Frequencies**

*Sampsa Fabritius, Nokia Mobile Phones, Finland  
Raimondas Lencevicius, Nokia Research Center, USA  
Edu Metz, Nokia Research Center, USA  
Alexander Ran, Nokia Research Center, USA*

The poster describes a new approach to experimentally estimate the application schedulability for various processor frequencies. We use additional workload generated by an artificial high priority routine to simulate the frequency decrease of a processor. Then we estimate the schedulability of applications at different frequencies. The results of such estimation can be used to determine the frequencies and control algorithms of dynamic voltage scaling/dynamic frequency scaling (DVS/DFS) implementations. The poster presents a schedulability problem that is the basis for DVS/DFS applicability, the proposed schedulability estimation method, its analysis and evaluation.

### **Error Propagation Analysis using UML Architecture Specifications**

*D. Nassar, West Virginia University, USA  
W. AbdelMoess, West Virginia University, USA  
M. Shereshevsky, West Virginia University, USA  
N. Gradetsky, West Virginia University, USA  
H. Ammar, West Virginia University, USA*

---

## Demonstrations and Posters

Bo Yu, *New Jersey Institute of Technology, USA*  
S. Bogazzi, *New Jersey Institute of Technology, USA*  
A. Mili, *New Jersey Institute of Technology, USA*

In this work, we introduce an information theoretic approach for error propagation analysis using information about components and connectors of software systems that we can derive from their UML architecture specifications. We apply our analytical error propagation analysis on a medium-size real-time command and control case study to estimate the probabilities of error propagation between components of this system. We conduct a controlled fault injection experiment to obtain empirical estimates of the error propagation probabilities between components of that system. We perform a correlation study between analytical

and empirical error propagation data to validate our analytical approach.

### **The Role of Trust in Software Outsourcing**

Nilay Oza, *University of Hertfordshire, UK*  
Tracy Hall, *University of Hertfordshire, UK*  
Austen Rainer, *University of Hertfordshire, UK*

The poster will show the role of trust in software outsourcing arrangements and present a model to identify the significance of trust in software outsourcing. The main objective of this poster is to provoke discussion amongst researchers in this area. We hope this discussion will help in critically evaluating software engineering collaborations.

---

## Venue and Additional Information

### **City of Portland**

The "Rose City's" culture simmers in coffeehouses, Native American art galleries, ubiquitous bookstores, musical and theatrical events, and lively brewpubs. Classical and modern art of all kinds is currently displayed at the Portland Art Museum. Portland is a wonderful city for walkers. The city's streets, which feature statues, fountains and half-size city blocks, prompted Portland's selection in 1998 as one of America's best walking towns by *Walking Magazine*. For visitors covering a larger portion of the city, public transportation is both accessible and convenient. Light rail trains (MAX), the Central City Streetcar and an easy-to-navigate bus system all offer free service within downtown Portland. Crowning the city's skyline is Mount Hood, the tallest peak in Oregon's Cascade Mountain Range. Within the metro area are 37,000 acres of parks and green spaces that include fabulous rose gardens and the brand new Classical Chinese Garden. For hikers and mountain bikers, the city is proud to host the nation's largest urban wilderness, the nearly 5,000-acre Forest Park.

Adjacent to Portland itself are the outstanding recreational opportunities of the Pacific Northwest. Windswept beaches, verdant forests, and snow-capped peaks give way to sweeping rangelands, towering rock formations, and dramatic river valleys in this diverse land. ICSE 2003 has a number of local excursions and tours available, such as visiting Mt. Saint Helens, whale watching on the coast, a winery tour in the Oregon countryside, and a Native American cultural tour on an Indian reservation. To sign up for these opportunities, visit the conference registration area or follow the instructions on the web at:

<http://cs.oregonstate.edu/icse2003/travel/tours.html>.

### **Reception**

The reception is a time for attendees to mingle and relax together. Extra tickets are available for purchase at the conference registration area.

Funding for the reception has been graciously provided by Northrop Grumman.

### **WOW! The ICSE Newsletter**

Continuing a longstanding ICSE tradition, our onsite newsletter ("Window On the World" or WOW) will be published every morning of the conference. WOW will include a medley of previews of the day's events, reviews of yesterday's happenings, interviews with key people and award winners, pointers to local attractions and entertainment, interesting technical and non-technical news from beyond the borders of our sheltered hotel, including weather, humor, poems, contests, and maybe even some editorials.

If you're intrigued to join the WOW staff (2-3 hours nightly, for 2 or 3 or 4 days), options include production and copy editing in addition to producing content, send mail to Hal Hart, [Hal.Hart@ACM.ORG](mailto:Hal.Hart@ACM.ORG), to discuss openings and "benefits." Help make WOW a remembrance taken home by conferees, shared with others, perhaps even a conference highlight.

### **Student Volunteers**

The student volunteer program is an opportunity for students from around the world to associate with the top researchers and practitioners in software engineering. Student volunteers help with registration and assisted the organizers with running the conference smoothly. Student volunteers must be enrolled in a Ph.D., Masters, or full-time undergraduate program at the time of the conference.