First International Workshop on Software Factories

At OOPSLA 2005

Theme

Over the past few years, significant progress has been made in a variety of disciplines that build on object orientation, such as component based and model driven development, software architecture, aspect oriented software development, generative programming, requirements engineering, process engineering, and product line engineering. While progress in each discipline has occurred relatively independently of progress in others, several initiatives focusing on various synergies between disciplines suggest that understanding how they interrelate is one of the keys to further progress. Despite this progress, however, many open issues remain.

Software Factories is a new paradigm for automating software development, integrating advances in multiple disciplines to increase agility, productivity, and predictability across the software life cycle. It differs from other model driven methods through its reliance on domain specific languages and software product line practices, and its emphasis on integrating modeling with patterns, frameworks, testing, refactoring, and other agile, code focused development practices. It differs from other software product line engineering methods through its use of model driven development as a basis for automation, and its emphasis on making software product lines practices more accessible.

A Software Factory defines a tailored methodology for a specific system family, defining its life cycle processes, architecture, implementation, and deployment topology, using a graph of viewpoints. The factory associates reusable assets with each viewpoint, and delivers them in the context of the viewpoint, eliminating the need to search for applicable assets, and supporting manual and automatic guidance enactment and validation.

The graph of viewpoints, called a schema, relates work done at one level of abstraction, in one part of the system, or in one phase of the life cycle, to work done at other levels, or in other parts and phases. It may be used to fully or partially generate artifacts from others, to keep artifacts synchronized during development, to validate hand developed artifacts, to determine the impact of defects or changes in system requirements, to progressively map requirements onto implementation, to organize and apply patterns and other best practices, to capture metadata during development to facilitate operation and maintenance, and to provide other forms of guidance and governance.

Software Factories also provide a vehicle for automating the packaging and delivery of assets, including models and model based tools, other types of tools, such as wizards and utilities, tailored development processes, implementation components, such as class libraries, frameworks and services, and other types of assets, such as patterns, templates, style sheets, help files, configuration files, and documentation. Because factories are themselves defined using models, their definitions can be manipulated using tools, for example to compose larger factories from smaller ones, or to customize generic factories to create specialized ones.

More information on Software Factories can be found at http://www.softwarefactories.org/.

Topics Of Interest

Topics of interest for the workshop include the following. Position papers should shed light on one of these topics, or focus on synergies among topics from at least two different subheadings.

Requirements Modeling

- What types of models can be used to capture, analyze, and manage requirements?
- What are the relative strengths and weakness of different approaches to business process modeling?
- How do process and information models map onto software architectures?
- Should business processes be modeled independently of system processes and if so, how are the two types of models related?

Requirements Traceability

- How should architectures map requirements onto configurations of components that form systems?
- How can requirements be mapped effectively to architecture, implementation, deployment topology and development process for a family of systems?

Variability Modeling and Management

- How should variability in production assets be managed to facilitate their systematic customization and reuse in multiple contexts?
- How should variation points be defined so that they can be discovered and manipulated using tools?
- What are the most effective ways to express variability in models?
- How can the different variability mechanisms used at different levels of abstraction, and at different stages in the software life cycle, be integrated effectively?

Architectures, Components, and Services

- How do service oriented technologies and architectures facilitate and/or hinder assembly?
- How can models be used to provide more and better information about components?
- How should the semantics of features and feature interactions be defined, and how can this semantic information be used to support feature composition?
- How can models be used to automate and optimize asset configuration and composition, and to validate and reason about the results?

Modeling Techniques

- What are the most effective techniques for addressing cross-cutting concerns in models?
- What are the salient trade-offs between a single universal metamodel and a collection of many independently developed metamodels for the purposes of model driven development?
- What are the most effective techniques for composing multiple interrelated models?
- What are the salient trade-offs between general purpose and domain specific languages for the purposes of model driven development?
- How do modeling tool specifications relate to the specifications of the languages they manipulate?
- How should textual or graphical editors, compilers, debuggers, and other model based tools be specified, and how much of their implementations can be generated from the specifications?

Standardization Issues

- Should one or more metamodels or ontologies for software factories be standardized? What aspects of software factories, if any, should be addressed by such standards?
- What are the salient trade-offs between standardizing metamodels and standardizing information interchange formats?
- How effective are existing standardized metamodels for the purposes of model driven development (e.g, UML, MOF, SPEM, EDOC, CWM, OPEN, AS 4651)?

Automatic Configuration, Code Generation, and Model Transformation

- What are the best ways to capture configuration knowledge for a system family, and to use it to support the development of family members?
- What are the most appropriate approaches to model transformation for practical application in software engineering environments?
- How should the semantics of model transformation and integration be defined?
- How can model based metadata be used across the software life cycle, especially beyond code and artifact generation at development time?

Verification and Validation

- How can model analysis help assess architectural compliance with system requirements?
- How can the architecture of a system be described to ensure that invariants are maintained during the course of development, and to enable validation that functional and operational requirements will be satisfied?

Architectural Modeling

- How can models used for automation be integrated with descriptions of software architecture?
- How can models be used to describe architecturally significant constructs, such as components, services, systems and subsystems, and their integrations?
- What are the salient trade-offs between architecture description languages and architectural modeling languages?

Domain Analysis and Scoping

- How can frequently encountered system families be identified and classified, and what kind of information should be captured about them beyond architectural style?
- How could schemas for frequently encountered system families be used to organize the pattern literature, well known development methods, or other forms of practice guidance?
- How can technologies appropriate for developing a given type of system be identified, and their interactions in the resulting system described?

Schema Design

- How should multiple overlapping, interrelated and simultaneous concerns be separated and modularized to facilitate their simultaneous use and subsequent composition?
- What viewpoints occur in frequently encountered system families and how should they be organized?
- What kinds of relationships can be defined among viewpoints and what activities can they support across the software life cycle?
- What concerns are commonly encountered across the software life cycle by people in various roles, and what are the most appropriate viewpoints for each of them?

Process, Guidance and Governance

- What kinds of mechanisms are effective in supporting the delivery, enactment, and validation of architectural guidance?
- How should software architecture be described to facilitate the manual and automatic delivery, enactment, and validation of architectural guidance?
- How does the delivery, enactment and validation of guidance and governance improve the scalability of agile development practices?

Agility in Model-Driven Development

- How can models be integrated effectively into file oriented software engineering environments?
- How can model driven methods be integrated with agile, code focused development practices?
- How can model driven methods assist developers with implementation tasks, such as querying and navigating code bases, debugging, profiling, coverage analysis, pattern application and refactoring?

Product-Line Practices

- How can commonalities and variabilities, and reusable abstractions that represent them, be identified to support the derivation of Software Factories from existing products?
- How can the planning and implementation of multiple products be introduced into organizations focused on one off development?
- How can commercially available life cycle tools, such as requirements management, configuration management and defect tracking tools, and software engineering and testing environments, be used with families of systems?

Factory and System Evolution

- How should the evolution of family members be supported by a software factory?
- How can family members be maintained when the software factory used to build them evolves?
- How do factories evolve and what techniques can be used to facilitate their evolution?
- What does it mean to compose factories and what kinds of contracts should govern factory composition?

Organizational and Strategic Issues

- What are the key drivers for adopting software factories and the key challenges in doing so?
- What are most effective routes to adopting software factories, and what technical, business, or organizational competencies does each of them require?
- How would the widespread adoption of software factories affect the software industry?
- Are software factories equally suitable to both shrink-wrapped products and bespoke projects?
- What is the most appropriate scope for software factories in terms of organization, team, or project size? How far up or down do they scale on each of these axes?

Goals

This workshop aims to support the growing international community of interest in Software Factories by providing a forum for collaboration among researchers, practitioners, academics, and students. Its goals are to establish common vocabulary and shared perspective for Software Factories and its constituent technologies; to identify problems in developing and applying Software Factories, and to suggest strategies for solving them; to disseminate research findings and best practices, and to expose them to peer review; to collect scenarios that illustrate open issues; to develop consensus regarding research priorities; and to increase awareness and understanding of Software Factories in the industry at large.

Related Workshops

The following workshops were held on disciplines integrated by Software Factories within the last few years. This workshop will build on the results of these workshops, focusing ways of integrating findings and practices in the disciplines they address.

1. *Advanced Separation of Concerns*. ECOOP 2001 (Budapest, HU).
2. *Advanced Separation of Concerns in Software Engineering*. ICSE 2001 (Toronto, ON).
3. *Advances and Applications of Problem Frames*. ICSE 2004 (Edinburgh, UK).
4. *Agent-Oriented Software Engineering*. AAMAS 2003 (Melbourne, AU).
5. *Agent-Oriented Software Engineering*. AAMAS 2004 (New York, NY).
6. *Architecture Reconstruction and Product Lines*. SPLC 2000 (Denver, CO).
7. *Aspects and Dimensions of Concerns*. ECOOP 2000 (Cannes, FR).
8. *Aspect-Oriented Modeling with UML*. AOSD 2001 (Enschede, NL)
9. *Aspect-Oriented Modeling with UML*. AOSD 2003 (Boston, MA)
10. *Aspect-Oriented Modeling*. AOSD 2005 (Chicago, IL)
11. *Aspect-Oriented Modeling with UML*. UML 2003 (San Francisco, CA)
12. *Aspect-Oriented Modeling*. UML 2004 (Lisbon, PT)
13. *Automating Object-Oriented Software Development Methods*. ECOOP 2001 (Budapest, HU).
14. *Best Practices for Model-Driven Software Development*. OOPSLA 2004 (Vancouver, BC).
15. *Combining Architecture, Asset Management, and Culture to Successfully Develop Product Lines*. SPLC 2000 (Denver, CO).
16. *Component-Based Software Engineering: Benchmarks for Predictable Assembly*. ICSE 2002 (Orlando, FL).
17. *Component-Based Software Engineering: Automated Reasoning and Prediction*. ICSE 2003 (Portland, OR).
18. *Component Engineering Methodology*. GPCE 2003 (Erfurt, DE).
19. *Component-Oriented Programming*. ECOOP 2002 (Malaga, ES).
20. *Component-Oriented Programming*. ECOOP 2003 (Darmstadt, DE).
21. *Composition Languages*. ECOOP 2002 (Malaga, ES).
22. *Composition Languages*. ECOOP 2003 (Darmstadt, DE).
23. *Constructing Software Engineering Tools*. ICSE 2000 (Limerick, IE).
24. *Correctness of Model-based Software Composition*. ECOOP 2003 (Darmstadt, DE).
25. *Defining Precise Semantics for UML*. ECOOP 2000 (Cannes, FR).
26. *Describing Software Architecture with UML*. ICSE 2001 (Toronto, ON).
27. *Directions in Software Engineering Environments*. ICSE 2004 (Edinburgh, UK).
28. *Domain-Specific Modeling*. OOPSLA 2003 (Anaheim, CA).
29. *Domain-Specific Modeling*. OOPSLA 2004 (Vancouver, BC).
30. *Early Aspects: Aspect-Oriented Requirements Engineering and Architecture Design*. AOSD 2005 (Chicago, IL).
31. *Early Aspects: Aspect-Oriented Requirements Engineering and Architecture Design*. AOSD 2004 (Lancaster, UK)
32. *Early Aspects: Aspect-Oriented Requirements Engineering and Architecture Design*. OOPSLA 2004 (Vancouver, BC).
33. *Evolution and Reuse of Language Specifications for DSLs*. ECOOP 2004 (Oslo, NO).
34. *Feature Interaction in Composed Systems*. ECOOP 2001 (Budapest, HU).
35. *From Software Requirements to Architectures*. ICSE 2001 (Toronto, ON).
36. *Generative Programming*. ECOOP 2001 (Budapest, HU).
37. *Generative Programming*. ECOOP 2002 (Malaga, ES).
38. *Generative Programming*. OOPSLA 2001 (Tampa, FL).

39. *Generative Programming and Component Engineering*. Principles, Logics, and Implementations of High-Level Programming Languages, 2002 (Pittsburgh, PA).
40. *Generative Techniques for Product Lines*. ICSE 2001 (Toronto, ON).
41. *Generative Techniques for Product Lines*. SPLC 2000 (Denver, CO).
42. *Generative Techniques in the Context of MDA*. OOPSLA 2002 (Seattle, WA).
43. *Generative Techniques in the Context of MDA*. OOPSLA 2003 (Anaheim, CA).
44. *Graph Transformation and Visual Modeling Techniques*. OOPSLA 2004 (Vancouver, BC).
45. *Implementation of Software Product Lines and Reusable Components*. International Conference on Software Reuse 2004 (Madrid, ES).
46. *Integration and Transformation of UML Models*. ECOOP 2002 (Malaga, ES).
47. *Iterative, Adaptive, and Agile Processes*. ICSE 2002 (Orlando, FL).
48. *Managing Variabilities Consistently, Design and Code*. OOPSLA 2004 (Vancouver, BC).
49. *Managing Variability in Domain Engineering using OO Technology*. OOPSLA 2001 (Tampa, FL).
50. *Method Engineering for Object-Oriented and Component-Based Development*. OOPSLA 2003 (Anaheim, CA).
51. *Method Engineering for Object-Oriented and Component-Based Development*. OOPSLA 2004 (Vancouver, BC).
52. *Methods and Techniques for Software Architecture Review and Assessment*. ICSE 2002 (Orlando, FL).
53. *Model-Based Software Reuse*. ECOOP 2002 (Malaga, ES).
54. *Model Driven Architecture and Product Line Engineering*. SPLC 2002 (San Diego, CA).
55. *Model Engineering*. ECOOP 2000 (Cannes, FR).
56. *Model Transformation and Execution in the Context of MDA*. ECOOP 2004 (Oslo, NO
57. *Modeling Variability for Object-Oriented Product Lines*. ECOOP 2003 (Darmstadt, DE).
58. *Multi-dimensional Separation of Concerns in Software Engineering*. ICSE 2000 (Limerick, IE).
59. *Product Line Architecture*. SPLC 2000 (Denver, CO).
60. *Product Line Engineering, The Early Steps: Planning, Modeling, and Managing*. GPCE 2003 (Erfurt, DE).
61. *Requirements Reuse in System Family Engineering*. International Conference on Software Reuse 2004 (Madrid, ES).
62. *Reuse in Constrained Environments*. OOPSLA 2003 (Anaheim, CA).
63. *Semantics, Applications and Implementation of Program Generation*. Principles, Logics, and Implementations of High-Level Programming Languages, 2001 (Florence, IT).
64. *Software Architecture*. ICSE 2000 (Limerick, IE).
65. *Software Product Lines: Economics, Architectures, and Implications*. ICSE 2001 (Toronto, ON).
66. *Software Product Lines: Economics, Architectures, and Implications*. ICSE 2002 (Orlando, FL).
67. *Software Requirements to Architectures*. ICSE 2002 (Orlando, FL).
68. *Software Requirements to Architectures*. ICSE 2003 (Portland, OR).
69. *Software Transformation Systems*. OOPSLA 2004 (Vancouver, BC).
70. *Software Variability Management*. ICSE 2003 (Portland, OR).
71. *Software Variability Management for Product Derivation - Towards Tool Support*. SPLC 2004 (Boston, MA).
72. *Techniques for Exploiting Commonality through Variability Management*. SPLC 2002 (San Diego, CA).

Organizing Committee

The workshop organizing committee is representative of the various disciplines integrated by Software Factories, and we are interested in bringing together participants from similarly diverse backgrounds.

Jack Greenfield (chair and primary point of contact) is an Architect for Enterprise Frameworks and Tools at Microsoft. He was previously Chief Architect, Practitioner Desktop Group, at Rational Software Corporation, and Founder and CTO of InLine Software Corporation. At NeXT Computer, he developed the Enterprise Objects Framework, now part of Apple Web Objects. A well known speaker and writer, he is a co-author of the book "Software Factories: Assembling Applications from Patterns, Models, Frameworks and Tools", with Keith Short, Steve Cook and Stuart Kent. The book defines the Software Factories methodology in detail, and explains its motivation and timeliness within the industry. He has also contributed to UML, J2EE, and related OMG and JSP specifications. He holds a B.S. in Physics from George Mason University.

Steve Cook is a Software Architect in the Enterprise Frameworks and Tools group at Microsoft, which he joined at the beginning of 2003. Previously he was a Distinguished Engineer at IBM. He has worked in the IT industry for 30 years, as architect, programmer, author, consultant and teacher. He was one of the first people to introduce object-oriented programming into the UK, and has

concentrated on languages, methods and tools for modeling since the early 1990s. He is a member of the Editorial Board of the Software and Systems Modeling Journal, a Fellow of the British Computer Society, and holds an Honorary Doctor of Science degree from De Montford University.

Krzysztof Czarnecki is an Assistant Professor at the University of Waterloo, Canada. Before coming to Waterloo, he spent 8 years at DaimlerChrysler Research working on the practical applications of generative programming. He is co-author of the book "Generative Programming" (Addison-Wesley, 2000), which is regarded as founding work in the area and is used as a graduate text at universities around the world. He was General Chair of the 2003 International Conference on Generative Programming and Component Engineering (GPCE) and keynote speaker at UML 2004. His current work focuses on realizing the synergies between generative and model-driven software development.

Jeff Gray is an Assistant Professor in the Department of Computer and Information Sciences at the University of Alabama at Birmingham. He received the Ph.D. in May 2002 from the Electrical Engineering and Computer Science department at Vanderbilt University, where he also served as a research associate at Vanderbilt's Institute for Software Integrated Systems (ISIS). Several of his research interests intersect the ideas surrounding Software Factories: His funding from DARPA supported investigation into domain-specific modeling and aspect-oriented software development; his funding from IBM is driving the investigation into debugging and testing capabilities for domain-specific languages. Jeff was the co-founder of the annual OOPSLA conference on domain-specific modeling. For 2005, he is on the organizing and program committees of 16 events, including chair of the Doctoral Symposium of the MODELS conference (formerly the <<UML>> conference) and coordinator of workshops and tutorials at GPCE. Jeff currently serves as the chair of the Alabama IEEE Computer Society. More information about his research and publications can be found at http://www.gray-area.org.

Michael Stal is a Siemens Principal Engineer within Siemens Corporate Technology. Michael leads a team which focuses on middleware, architecture, integration in the context of distributed systems. In addition, he is responsible for a research programme that investigates architecture, processes and tools for the development of product families and platforms. Michael also supports Siemens business groups as a consultant and architect in product development. He has co-authored the books "Pattern-Oriented Software Architecture - A System of Patterns" and "Pattern-Oriented Software Architecture - Patterns for Concurrent and Networked Systems". He holds a diploma in computer science from the Munich University of Technology.

Gabor Karsai is Associate Professor of Electrical and Computer Engineering at Vanderbilt University and Senior Research Scientist at the Institute for Software-Integrated Systems at Vanderbilt. He got his BSc, MSc and Technical Doctorate degrees from the Technical University of Budapest, Hungary, in 1982, 1984, and 1988, and the PhD degree from Vanderbilt University, in 1988. He conducts research in model-integrated computing (MIC), in the field of development environments, automatic program synthesis and the application of MIC in various government and industrial projects. Since 1988 he has been involved in various projects on fault diagnostics modeling and algorithms, that have been applied and demonstrated in various embedded systems, including chemical manufacturing plants, the Space Station, and next-generation fighter aircraft. Recently, he was Principal Investigator on various DARPA-sponsored research projects on Fault-Adaptive Control Systems, on Autonomic Logistics, and on Software Synthesis for embedded systems. He is a member of the IEEE Computer Society and the TC on Computer-Based Systems.

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. Markus is the author of several magazine articles, patterns and books on middleware and model-driven software development. He is a regular speaker at conferences world wide. Markus can be reached at voelter@acm.org or via www.voelter.de.

Don Batory is a Professor in Computer Sciences at the University of Texas at Austin. He was an Associate Editor of IEEE Transactions on Software Engineering (1999-2002), Associate Editor of ACM Transactions on Database Systems (1986-1992), a member of the ACM Software Systems Award Committee (1989-1993; Committee Chairman in 1992), and Program Co-Chair for the 2002 Generative Programming and Component Engineering Conference. He is a leading researcher on feature-based and automated program development. He has given numerous tutorials on Feature Oriented Programming and Product-Lines and is an industry-consultant.

Brian Henderson-Sellers is Director of the Centre for Object Technology Applications and Research and Professor of Information Systems at University of Technology, Sydney (UTS). He is author of ten books on

object technology and is well known for his work in OO methodologies (MOSES, COMMA and OPEN) and in OO metrics. He was recently awarded a DSc degree by the University of London for his work in object-oriented methodology. He was co-chair of AOIS 2003 (Agent-Oriented Information Systems) in Chicago in October 2003, organiser of the OOPSLA 2002, 2003 and 2004 Workshops on Agent-Oriented Methodologies and organiser of the Second Workshop on Method Engineering for Object-Oriented and Component-Based Development at OOPSLA 2004.

Cesar Gonzalez-Perez has been working at UTS since April 2002, focusing on object-oriented and agent-oriented software development methodologies and metamodelling. Prior to this, he was the chief developer of the OPEN/Metis methodological framework. Cesar has been the co-organizer and chairperson of the Process Engineering for Object-Oriented and Component-Based Development workshop at OOPSLA 2003 and the Third International Workshop on Agent-Based Methodologies at OOPSLA 2004.

The workshop will be chaired by Jack Greenfield. All of the members of the organizing committee will review position papers and participate in the selection process. Every position paper will by reviewed by at least two committee members.

Important Dates

Initial position papers due: TBD
Notification of acceptance: TBD
Camera ready versions due: TBD
Workshop: TBD

Call for Participation

Researchers, practitioners, academics, and students focusing on Software Factories and its constituent disciplines are invited to participate in this workshop. Attendance is limited to 30 and will be by invitation on the basis of position papers that demonstrate insight regarding the challenges associated with integrating the broad range of disciplines relevant to Software Factories, and that describe relevant commercial or academic experience, and/or findings that suggest new solution strategies.

We are particularly interested in experience reports, case studies, practical applications of theory, results involving multiple disciplines, and evaluations or comparisons of methods, practices, or technologies. Papers without strong potential for practical application within the next three years will not even be considered.

Submissions will be evaluated and selected according to their pragmatism, originality, relevance to the workshop theme, goals and topics of interest, diversity of perspective, the amount of discussion they are likely to provoke at the workshop, and the significance of their contribution to the workshop outcome.

In order to promote coherent discussion, submitters should follow the terminology defined in the Software Factories book identified on the workshop web site. In the event that the submitters have issues with the terminology, they should explicitly identify and explain the points of disagreement and map their proposed alternative onto the terminology used in the book before using different terminology.

Accepted papers will be published in the workshop proceedings on the workshop web site. The workshop organizers may also elect to edit and publish the best papers in a book or journal, in which case their authors will be required to submit camera ready copy, and to sign the publisher's release documents.

Some of the accepted papers may be selected to be presented during the workshop. Presentations will be limited to 15 minutes each, including questions and answers. All of the accepted papers will be assumed to have been read by all of the participants, however, and will collectively provide a basis for discussion.

Papers will be selected to ensure a diverse set of interests at the workshop. The submission of a paper represents a commitment that at least one of its authors will attend the workshop if the paper is accepted. Parties intending to submit a position paper are asked to send email to jackgr@microsoft.com and stcook@microsoft.com carrying the subject line "Software Factories Workshop at OOPSLA 2005", indicating their intent to participate, and providing the title, authors, and a short abstract.

All papers must be submitted as PDF documents between 3 and 8 pages in length, and must adhere to IEEE CS Press Format. The work must be original, must not have been submitted or published in the same form in any other venue, and must include full contact information for at least one of its authors.

Accepted papers will be posted to the workshop web site at least three weeks before the workshop, along with a detailed workshop schedule. Participants will be expected to read all of the accepted papers, and to be prepared to engage in active debate and discussion regarding the positions taken.

Format

This is a full day workshop. Its format is designed to promote interaction among the participants, rather than presentation.

During the workshop, the participants will form breakout groups around topics drawn from the accepted papers. Each group will formulate and perform a collaborative exercise on their chosen topic, and will produce a deliverable to be presented in a plenary session at the midpoint of the workshop. Examples of the types of exercises desired include:

- Develop a prioritized list of research or engineering challenges associated with Software Factories.
- Describe some real world examples of Software Factories, or its constituent disciplines or technologies, and analyze the insights provided by each example.
- Develop a set of scenarios that involve two or more of the constituent disciplines or technologies.
- Develop a set of guidelines for practitioners based on real world experience with Software Factories or its constituent disciplines or technologies.

Each group will then be asked to challenge the position of one of the other groups. The challenges will be presented in a plenary session at the end of the workshop, and each group will defend its position against the challenges. Following the workshop, each group will submit a short paper representing its deliverable. These papers will be included in the proceedings.

The workshop organizers will compile a list of issues debated, points of consensus, and areas of disagreement. This information will also be included in the proceedings.

Schedule (tentative)

Format	Start	End	Description
Plenary	0900	0915	Welcome and introduction of participants
Plenary	0915	1000	Presentation of selected papers
Plenary	1000	1030	Identification of topics and organization of break out groups
	1030	1045	Coffee Break
Breakout Groups	1045	1215	Collaborative exercise and production of deliverables
	1215	1315	Lunch
Plenary	1315	1430	Presentation of recommendations
	1430	1445	Coffee Break
Breakout Groups	1445	1530	Formulation of challenges to deliverables
Plenary	1530	1645	Presentation of challenges and defense of deliverables
Plenary	1645	1700	Wrap up