The twelve papers accepted in this year’s Methods and Tools for IS Development minitrack provide an excellent overview of the current research being done in this field. For this year’s minitrack, we solicited and received research papers in many areas:

- Software requirements definition and validation
- Domain-specific modeling and architectures
- Software reuse strategies and techniques
- CASE tool integration and life cycle support
- Cleanroom software engineering methods and tools
- Software process modeling and assessment
- Empirical studies of tool and method effectiveness
- Formal methods for systems development
- Intelligent software agents
- Object-oriented development methods

Each of the submitted papers was refereed by three external reviewers and both minitrack chairs. The twelve accepted papers are organized into three sessions:

In the first session, we group papers dealing with the human element in software development. P.J. Guinan and Samer Faraj start off by presenting an empirical study of 57 software development teams that studies communication patterns and control strategies employed by the teams to reduce role and task ambiguity. Results clearly demonstrate that increased performance is associated with decreased levels of ambiguity. This line of research is continued in the second paper by Hakan Enquist and Nickolas Makrygiannis. They perform an empirical study on the frequency and impact of misunderstandings in software development projects. Using data from several European projects, they find that misunderstandings are a frequent cause of disturbances in complex software projects. Mike McGrath follows with an analysis of behavioral issues in software engineering process modeling. He argues that multiple views are useful for user understanding and effective participation in the software development process. The paper by James Collofello, Dwight Smith-Daniels, Ioana Rus, Anamika Chauhan, Dan Houston, and Douglas Sycamore completes the first session. The authors apply a systems dynamics simulator to study staffing decisions in software projects. Their results show no clear best staffing policy. Tradeoffs involving budget and completion time must be considered.

The second session brings together papers on Cleanroom software development and component-based software development. Marc Frappier and Richard St.-Denis present a new and innovative technique for specifying black boxes. Entity structure diagrams are used to streamline the specification of valid input sequences and corresponding outputs. The specification technique can be used to provide Cleanroom support for object-oriented development. The use and benefits of Cleanroom are demonstrated in the next paper by Frank Coyle and Robert Oshana. They discuss the customization of Cleanroom methods to a complex project. Vu Tran, Sophie Tsui, Dar-Biau Liu, Brad Hummel, and Jake Doan present excellent insights for managing a component-based software project. Using the IRIDIUM project as a case study, the authors discuss the cyclical dependency between system requirements and the selection of reusable components. They conclude with a set of risk mitigation strategies. In a different application domain, Marc Braun and S. Mohle share their experiences with component-based development in MRP systems. They present practical advice on configuring and interfacing off-the-shelf components to build effective software systems.

The third session focuses on software system requirements specification methods. Arco Dalebout, Jos van Hillegersberg, and Berend Wierenga begin by proposing the use of an object-oriented modeling framework in a semi-structured application domain. The domain framework is tailored and more fully structured to the point where it can be used effectively for complete object-oriented analysis and design. An example involving a sales promotion system is provided as a demonstration of the technique. Tereza Kirner and Rogeria Gratao present a tool for the specification of multiple views of system requirements. They use a requirements metamodel to translate between different requirement models. The use of the tool is demonstrated on several examples. The final two papers are by Raymond Buhr, M. Elammari, Tom Gray, Serge Mankovski, and D. Pinard. They present a theory and set of methods for specifying multi-agent software systems. Use case maps (UCMs) provide a high-level representation of whole system behavior patterns. The two papers present the new theory and methods as the authors apply UCMs to several interesting applications in telephony and multimedia systems.