Panel 1

Large Scale Software Transformation: Lessons Learned from the Year 2000 Problem

The Year 2000 problem has posed significant challenges that are being addressed using interesting automated software engineering (ASE) tools, techniques, and methodologies. This panel will attempt to relate and extrapolate the tools, techniques, and methodologies to those for other large-scale software transformation problems.

The focus will be on comparing and contrasting the Year 2000 problem with other interesting classes of problems and utilizing the lessons learned from the Year 2000 problem for motivating further development in ASE techniques applicable for the various phases of software transformation lifecycles. The panel will attempt to leverage the unprecedented scale required for solving the Year 2000 problem towards the goal of scalable automated techniques with well-defined characteristics.

The preliminary focus of the panel will be on tools and techniques. Process and methodology issues may be considered in the context of tools and techniques or separately if time permits and the audience demonstrates a preference for them. The list of questions to be addressed includes but is not limited to:

1. Problem Assessment and Estimation
   A. What are the other interesting classes of problems, and their similarities and differences, such as: currency changes, field (e.g. part number) expansions, and legacy system transformation.
   B. What are the characteristics of the impact assessment techniques used in solving the Year 2000 problem? What are the tradeoffs involved?
   C. How could the techniques used for partitioning and clustering “work units” of a transformation project be applied to other transformation projects?

2. Detailed Analysis and Transformation
   A. What are the roles of different analysis approaches? What are the tradeoffs involved in applying syntactic, semantic, and slicing techniques?
   B. What are the issues involved in separating analysis from transformation, and in comparing user-assisted versus automated transformation?

3. Testing and Verification
   A. What are the key tools and techniques required for planning and designing large scale tests? How could we quantify and evaluate the tradeoff between testing intensity and correctness?
   B. Are there any interesting techniques for automated test-bed building that can be generalized?

4. Common Techniques
   A. What are the key features of repositories that assist in various parts of the lifecycle? What are the consequences of store versus recompute decisions on scale, consistency and usability.
   B. Are there any canonical representations for software components that are critical for one or more phases?
   C. What are the key lessons learned from integrating analysis tools, repositories, code transformation tools and testing tools?