Within reengineering, we will see a growing interest in formal methods in the future, both in semantics (quality) and in metrics (quantity). As software systems become more and more complex, their change management needs to evolve from manual and adhoc-methods to a rigorous discipline. To achieve this, both sides of software engineering, notations and processes, need a clear and integrated semantics on which solid tool support can be built. To satisfy this demand, research on formal methods should take the special requirements of software maintenance and reengineering more into account. The papers of the forthcoming session are good examples that this line of research leads to useful results.

In the past, formal semantics have mainly been applied in specifying and designing new software systems, that are developed from scratch. In this context, they are the starting point for a rigorous and correct development of the final programs. Research on applying formal methods to the restructuring of existing systems (besides statistics and metrics) is rather rare. In this scenario, they provide the basis for analysis tools and the definition of ‘semantics preserving’ operations. Since the demand for reengineering and maintenance support increases, we will see more formal methods being applied in this field in the future. One major problem for the application of formal methods to existing systems is the lack of precise semantics of the older programming languages. That this is not only a problem of traditional languages like FORTRAN and COBOL is demonstrated within the first paper of the session. The paper discusses a denotational semantics for CPP, the classical preprocessor for the C-language family. The semantics provides a solid basis for analysis tools and maintenance support. The major example in the paper is a program specialization technique for variant removal and partial readings.

The second paper introduces a new and interesting approach for the assessment of the failure-proneness of evolving software. It applies statistical methods. This research area has a longer tradition within reengineering, especially for the measurement of the quality and maintainability of software systems by appropriate metrics. Its application to the prediction of the failure-proneness based on testing results and general constraints restricting the evolution spectrum of software is rather new.

As the two papers indicate, there are some useful and applicable formal methods available. To show their value within software engineering and reengineering in the long run, all these techniques, qualitative and quantitative ones, should be integrated within a uniform framework for formal software evolution: a challenging task for the future.