The Ada effort is over ten years old. Ada has evolved through many stages: the initial studies on high-order languages for embedded computer systems; language requirements formulation/review; competitive language design; language evaluation; environment requirements formulation/review; compiler development/validations; and standardization. Ada technology is having a significant impact on industry, academia, and government. The original goal of a single, high-order language for embedded computer system applications has resulted in a powerful language for system and application programming as well. An extensive set of programming support environments are evolving with the language.

This special issue of *IEEE Software* is an outgrowth of the IEEE Computer Society Conference on Ada Applications and Environments held in St. Paul, Minnesota, October 15 to 18, 1984. The Ada topics in the conference program included methodologies, run-time models, environments, tools, distributed implementations, applications, use, validation, and education. The conference was initiated to serve as a forum for obtaining the current status of research and development related to the Ada language and environments.

The articles solicited for this special issue were among the key papers from the conference. Each illustrates the emerging emphasis on issues of reliability and efficiency in large and complex software systems. The articles cover a range of topics from static analysis tools to tasking implementations, and from methodologies for design of automated tools to issues of run-time efficiency. Instead of proposing yet another new editor, project management tool, or job control language, these articles argue for greater emphasis on analysis tools, highly integrated environments, and incremental development.

In their article, David C. Luckham and Fredrick W. von Henke describe a generalization of Ada to a specification language called Anna. The authors have been careful to provide the extension in a form that generalizes concepts and constructs already in Ada but that can be interpreted as comments within the existing Ada standard. By providing a rich set of annotations without requiring completeness, Anna opens the door to a
wide variety of analysis tools and allows the user to determine the thoroughness of specifications in any given application. The formal specifications of Anna suggest tools for checking consistency between the Ada program and the specifications, for translating the specifications into runtime checks, for automatically generating test, and for improving code optimization.

The article by David S. Rosenblum follows the Anna article by describing a practical design approach that can be applied to a broad class of Ada software tools. The method uses source-to-source transformations operating on Ada programs represented in a Diana form. He details the steps in implementing a transformation tool. He illustrates the steps using the tasking monitor described in the Helmhold and Luckham article, which also appears in this issue. The method depends on a common internal program representation, incorporates an incremental approach to tool implementation, and is suggested for use in program optimizers, readability enhancement tools, and debugging systems.

Ted P. Baker and Greg A. Riccardi survey the issues in efficiently implementing Ada tasking. They examine individual features supporting Ada tasking and propose an implementation for each. The article provides a clear explanation of Ada tasking complexities and a pragmatic approach to efficient implementation. It provides a thorough guide to Ada task implementation by concentrating on the issues unique to Ada and avoiding the details of well-known techniques.

David C. Helmhold and David C. Luckham discuss a real-time monitor that estimates the state of the tasks of a program. An analysis of this kind is important for the detection and elimination of a variety of dead locking errors that can arise in any multi-tasking application. Such errors often occur unpredictably in real-time systems, are influenced by random system events outside the program itself, and can go undetected in operational systems for extended periods. Ada provides a formal descriptive mechanism for describing multi-tasking programs. The reported monitor is a useful debugging tool for detecting and reporting potential deadlocks. The importance of this topic will grow as the number of Ada programs that are written and fielded increases.

Alexander L. Wolf, Lori A. Clarke, and Jack C. Wileden describe tools for precisely controlling the interfaces among modules in the design of very large systems. Although they focus on language features for interface control in the context of Ada, they argue in general for more rigorous analysis of programs throughout development and maintenance and for a more integrated environment approach for programming-in-the-large.

Joseph E. Urban is an associate professor of computer science, responsible for the software engineering program, at the University of Southwestern Louisiana. He is an editor of the *IEEE Transactions on Software Engineering* and a member of the IEEE Computer Society Governing Board. His research interests are software engineering, specification languages, test data generation, and computer languages.

Urban earned a BS in computer science from Florida Institute of Technology (1973), an MS in computer science from the University of Iowa (1975), and a PhD in computer science from the University of Southwestern Louisiana (1977). His PhD thesis was selected by the ACM Doctoral Forum as one of the four best computer science theses produced in 1977-78.

His address is Center for Advanced Computer Studies, University of Southwestern Louisiana, PO Box 44330, Lafayette, LA 70504.

David A. Fisher is president of Gensoft Corporation and Incremental Systems Corporation. His current interests are in software development and maintenance environments and in all aspects of Ada implementation and use. He is a senior member of the IEEE, a member of the ACM, 1985 Distinguished Visitor for the IEEE Computer Society, and a member of the Ada Board. He has been involved with the Ada effort since its inception in January 1975.

Fisher received a BS in mathematics from Carnegie Institute of Technology (1964), an MSE from Moore School, University of Pennsylvania (1967), and a PhD in computer science from Carnegie-Mellon University (1970).

His address is Incremental Systems Corporation, 319 South Craig Street, Pittsburgh, PA 15213.