Preface

This issue contains nine papers and a technical note on a variety of subjects, including communications technologies, compilers, program understanding, and expert systems for dump analysis.

The first paper, by Baade, discusses the use of the Network Design and Analysis (NETDA) tool to construct optimal computer network traffic routes for use with complex Systems Network Architecture (SNA) networks. The author has taken advantage of features in NETDA that allow for the use of actual network traffic data in the design of the routings. This approach yields significant reductions in network utilization without reconfiguration of the underlying hardware.

The Open Systems Interconnection (OSI) protocols have been made part of the Systems Application Architecture® (SAA™) and are expected to be implemented across all SAA operating systems. Goldberg and Mouton present the architecture of the OSI/Communications Subsystem Base, which provides a portable base for OSI implementations across diverse operating systems with dissimilar architectures. The Base in turn supports the OSI/Communications Subsystem and is capable of supporting similar layered communications protocols, especially if portability is of concern.

Multimedia instruction provides an opportunity for industry and academia to move toward an educational environment in which individualized instruction is the rule, rather than the exception. Multimedia efforts at California State University at Fullerton and at IBM are anticipating future educational programs that are distributed, modular, multisensory, portable, interruptible, nonlinear, transferable, responsive, engrossing, and enjoyable. Reisman and Carr describe these efforts, trace their history within the context of individualized instruction, provide a look at recent developments, and project a future educational milieu.

Sahulka, Plachy, L. Scarborough, R. Scarborough, and White describe the capabilities of a new level of FORTRAN—IBM Clustered FORTRAN—that makes it possible to share all the resources of two IBM ES/3090 Model 600 systems (up to 12 processors and Vector Facilities) executing one FORTRAN program. These capabilities include dynamic allocation of real processors at execution time, to make best use of available actual resources and to make the compiled program independent of resource availability. The authors discuss the associated language extensions, compiler, library, and high-speed hardware connection between ES/3090 systems.

Turning to another programming language, the next paper presents the foundations for partial compilation of the Restructured Extended Executor (REXX) language, which has in turn led to the new REXX compiler and could affect compilers for other interpretive languages. Pinter, Vortman, and Weiss describe the most important features of partial compilation for REXX. These include definition of an abstract machine and an intermediate language to support execution, management of bound variables through a unique symbol table organization called lazy hiding, additional symbol table support for compound variables, saving of multiple representations of variable values to avoid conversions, and the use of other delayed execution techniques. Many of these features are intended to improve execution efficiency in the compiled REXX environment.

The dramatic increase in peripheral equipment for personal computers has led to a need for efficient means to program their diverse device drivers in Operating System/2® (OS/2®). The traditional approach utilized assembler languages, but the opportunity exists to use higher-level languages in support of device driver code. The author, Feriozi, constructs a model for device drivers built on the C programming language. Efficiency is always a concern with device drivers and with higher-level languages. The author addresses these concerns in several ways, including the elimination of the C startup routines and the C run-time library.

Lenz and Saelens show how expert systems and knowledge-based technology can be beneficially applied to the classic area of reading dumps from the Multiple Virtual Storage (MVS) operating system. Their approach also demonstrates how the techniques learned in knowledge-based systems for medical diagnostics, which are fairly advanced, can be reused to develop such systems for another area. The data collected on use of this expert system—the MVS Dump Analyzer—show that many problems can be solved through the system without having a human consult the dump at all, and that the average saving in dump analysis is about one half hour. The authors also present their experiences in building the expert base and in educating people in the use of such new technologies.

The IBM Rochester Programming Laboratory has recently been in the public view because of its high-quality environment and results, as evidenced by the 1990 Malcolm Baldrige National Quality Award. Kan provides a look inside the processes used to achieve the necessary quality results by writing about the quality models used to estimate software reliability and to manage software development quality. There are five models used, two that span the software development life cycle and three that are for more specialized purposes. The author points out that the largest obstacle to successful use of quality models is not the models themselves, but the availability of sufficient, relevant data to support the use of the model. The history of Rochester's quality achievements can be seen in a simple and understandable form through these models.

Brown shows how the two fundamental aspects of software—documentation about the software and analysis of the source code structure—can be brought together through tools to aid software developers and maintainers during the life of a software product. His approach utilizes and integrates tools for program understanding and hypertext documentation. The ability to move between these two sources of information and to readily form the needed correspondences can be of considerable help to those who must analyze software problems and understand existing software that is new to them.

In a technical note, Appel, Cuomo, Overly, Walicki, and Yozzo present the results of their efforts to add images, and particularly those of people, to on-line systems that provide information about people, such as on-line directories. In particular, they describe the WATINFO system used at the IBM Thomas J. Watson Research Center and its use of face images to enhance the appeal of directories and similar systems.

The next issue of the *Journal* will be a special issue devoted to papers on APL (A Programming Language) in celebration of its 25th anniversary.

Gene F. Hoffnagle
Editor