Preface

Rapid development of business software for critical distributed applications is, and has long been, a key aspect of IBM's support for businesses and for independent software vendors (ISVs). IBM's San Francisco* project, which is the subject of most of the papers in this issue, provides that support and is an excellent example of joint IBM and ISV efforts. More than 130 ISVs have participated in the creation and development of San Francisco common and domain-specific frameworks. The frameworks allow each ISV to focus on the specific needs of its customers, resulting in products with a robust object-oriented architecture and for a networked environment.

This issue contains an introductory paper on the San Francisco architecture and frameworks, four papers by ISV developers on their use of the frameworks for application development, and a Technical Note describing a supporting tool. There are also three papers on other subjects: software requirements, reverse engineering, and code restructuring. We are indebted to V. M. Johnson of the AS/400 Division in Rochester, Minnesota, for his foresight in conceiving, and his considerable efforts in developing, the San Francisco theme for this issue.

Bohrer presents an overview of the San Francisco project and its results in a paper on the origins of the project, its objectives, the collaboration with numerous ISVs, the methodology used to carry out the project, and the development of the architecture and frameworks. The key objectives were ease of ISV entry into object-oriented techniques, capability for ISVs to develop significant business applications, and open frameworks allowing for ISV choices and trade-offs.

The ability to expand the power of accounting systems into the more important arena of business enterprise models is the focus of a paper by Inman. The San Francisco frameworks are shown to be an effective path toward this expansion, allowing for more flexible and capable financial systems, serving management needs and supporting business analysis. The paper discusses general ledgers as an example in which San Francisco frameworks succeed.

Some ISVs and some of their customers face significant challenges in moving their software skills and products to object-oriented technologies, such as those underlying the San Francisco architecture and frameworks. Henders shows how the use of such technologies was introduced into an existing software mix, how the challenges were overcome and the difficulties avoided, and how the San Francisco frameworks support an easier transition to the new object-oriented development methods and technologies.

Knowledge workers, knowledge applications, and the challenge of making those applications serve broader business purposes are explored in a paper by Retallick and Sánchez. Their claim is that managing knowledge data and organizing business activities are rarely amenable to real change, but that the San Francisco frameworks offer such an opportunity for significant progress toward rapidly developed and integrated knowledge-based systems.

Van der Salm discusses the issues faced by an ISV when it considers the prospect of moving from tried-and-true software methods and tools to new ones that offer the promise of faster and more effective software development and product delivery. It is particularly important for small ISVs to find a path to newer and better technologies for software development, despite the risks and costs often assumed with change in small companies. The experiences the author relates illustrate the potential of San Francisco frameworks for ISVs moving to object technologies.

The last contribution on the San Francisco frameworks appears as a Technical Note. The author, Polan, describes how to combine the power of the VisualAge* for Java** integrated software develop-

ment environment, with the structure and support of the San Francisco object-oriented, Java-based frameworks. This Note is intended to provide technical details for using these two systems together.

The development of software requirements is a challenging and complex task, which is both essential to the value of the system that results and fundamentally human in its process and progress. Crowston and Kammerer report on a study of how requirements analysts can manage their efforts to overcome the challenges and produce better results. The authors used coordination theory to understand the management of dependencies and collective mind theory to understand how individuals affect the work of a team.

Aiken builds on the general understanding of reverse engineering of programs to describe the newer area of reverse engineering of data, which combines structured data analysis and data management principles. An activity model and a data model are developed to reach the goal of integration of enterprise-wide data. The author presents four brief scenarios in which data reverse engineering was used to solve or resolve enterprise data issues, and proposes the use of these techniques for the Year 2000 transition.

Dynamic profiling tools have been used to support compiler-based optimization of applications by exposing their behavior during execution and particularly their uses of execution paths and data. Schmidt et al. thought that these same techniques could be applied to code reordering of operating systems software with similar beneficial effects on execution behavior. Their results were significant and argue well for reordering operating system code using profile data.

The next issue of the *Journal* will be a special issue on the Java language, application development in Java, and IBM's support for Java.

Gene F. Hoffnagle
Editor

*Trademark or registered trademark of International Business Machines Corporation.

**Trademark or registered trademark of Sun Microsystems, Inc.