The Y2K issue generated an enormous flood of activity for organizations worldwide. While we survived Y2K with minor glitches, the Y2K “exercise” forced both developers and users of software to appreciate the need for software to function correctly in expected and unexpected situations. None of us is immune to the problems of the software industry. We must continue to drive our cars, fly on airplanes, use computer-controlled elevators, perform online transactions, and undergo medical treatment. If requirements are not well-defined and software is poorly developed, we all will suffer the consequences.

Integrating Technology

A combination of factors has made Internet-based computing an essential component of all types of businesses, and it is fast becoming a household necessity. Hardware technology has made sophisticated computers affordable by a large range of consumers. To keep up with hardware, many aspects of software technology have made significant advancements, including Web-based technology and general networking and communications.

But the field of software engineering has
made only incremental progress. Perhaps this is partly because the engineering of software is generally very difficult. After all, any uncertainty in a system is usually handled in software, since it is “easier” to change software. To make matters worse, as computing technology advances, greater demands are placed on software: an increasing number of critical systems rely on software, from avionics and air traffic control to banking and financial systems to medical life-support systems. Software is pervasive, and software errors have potential for costly disasters, both unintended and intended. The situation is much changed since 10 or 20 years ago.

Requirements Engineering and ICRE

A central factor in how well software is developed and operates is how well the requirements are understood during the development process. It is this realization by the development community that has stimulated a renewed interest in identifying more rigorous techniques for requirements engineering.

The papers for the last three International Conferences on Requirements Engineering3–5 reflect this shift in the research and industrial communities, both of which are investigating the use of rigorous techniques to handle requirements engineering. The success stories usually come from developers who have applied integration at many levels, including integrating research and industrial experience, integrating informal and formal techniques, and validating research results with practice. The papers that are part of the ICRE 2000 program6 were selected for their potential impact on the practicing requirements engineering community. Many of them involve industrial case studies that validate novel techniques. Often these techniques introduce automated means of checking consistency or completeness of requirements in ways that are difficult for humans to perform reliably. Such techniques are founded on formal models of requirements and strive to make the properties of the models easy to use by the typical software developer.

With the current interest in object-oriented development methods, techniques to integrate requirements models into existing OO notations and methods have also been appearing. As software engineering explores new approaches for improving productivity of software developers, such as software product-line engineering and domain engineering, new requirements engineering techniques suitable for these new approaches emerge.

The program committee selected three of these papers for inclusion in the conference and publication in IEEE Software because of their novel integrative approaches to different components of requirements engineering:

- Some of the articles describe how we can combine informal and formal approaches to requirements engineering, resulting in a more effective technique.
- Some show how experience and efforts between industrial and research organizations were integrated in order to develop and validate new techniques.
- One article demonstrates how integrating research and empirical investigations can produce a much more convincing demonstration of the usefulness of requirements engineering techniques than when only research or empirical data is available.

The first article, “Validating Voice Communication Requirements Using Lightweight Formal Methods” by Johann Hörl and Bernhard K. Aichernig, presents a “lightweight” approach to formal methods in order to perform requirements validation, where the formal specification is used to specify precisely an industrial voice com-
Systematic, rigorous engineering approaches to software development must start with systematic, rigorous approaches to requirements engineering.

The authors were able to detect ambiguities and inconsistencies when using a formal specification language to represent the system requirements; all of these problems can be detected without using heavy-duty theorem provers. In addition, the authors used the specification to help generate test cases automatically.

The second article, “Using Different Communication Media in Requirements Negotiation” by Daniela E. Herlea Damian, Armin Eberlein, Mildred L.G. Shaw, and Brian R. Gaines, offers an interesting study of a common problem faced by industry: How much impact do group dynamics and communication have on the requirements negotiation process? Particularly in the current age of widespread Internet use and more sophisticated meetings software, there is an increasing tendency to conduct group meetings through teleconferencing or video conferencing. This article empirically compares requirements conflict resolution in face-to-face and non–face-to-face meetings.

The third article, “A Reference Model for Requirements and Specifications” by Carl A. Gunter, Elsa L. Gunter, Michael Jackson, and Pamela Zave, presents a reference model for requirements and specification, thus enabling developers to perform informal and formal analysis of properties between different parts of the system, the environment, and between the system and environment. Developers can use the reference model informally to provide general guidance for the requirements engineering process, or they can incorporate proof obligations that let automated reasoning tools perform consistency checks.

Many of us in the software engineering field have long believed that systematic, rigorous engineering approaches to software development must start with systematic, rigorous approaches to requirements engineering. Finding ways to apply these approaches in practice on a wide scale has been the stumbling block. The ICRE 2000 contributions appearing here and in the conference demonstrate the continuing progress that is being made toward greater practical application. We now have the computing power, the knowledge of how to apply it, and enough experience in applying advanced requirements engineering techniques so that their wider use is obtainable with modest effort. The problem is to inform the development community of their availability.

References