Guest Editorial

Prasun Dewan, Purdue University

Collaborative software, also called groupware and multiuser software, allows multiple users, possibly geographically dispersed, to use the computer to solve problems together. The motivation for such software is straightforward and compelling: people spend a large fraction of their time doing group work. Collaborative software can allow them to hold computer-supported meetings without travelling great distances or losing access to the resources available at their workplaces. With networking becoming pervasive, there has been great interest in developing software that combines communication and computing.

Several computing concepts must be rethought before this idea can become practical. Transaction models must allow users to flexibly share information, input/output primitives must support multiuser interaction, distributed systems must allow efficient communication and synchronization of speech and images of the collaborators, undo/redo models must allow collaborators to recover from mistakes and explore alternatives, and so on. Today, researchers from several disciplines are engaged in identifying the issues raised by collaborative software, and extending and integrating the fundamental concepts in their areas. Conferences and workshops in user-interfaces, database systems, computer networking, distributed systems, and software engineering regularly hold special tracks devoted to collaboration. Moreover, new conferences and workshops have emerged to bring the researchers from different communities together to help identify the design space of collaborative applications and techniques and tools for implementing them.

The four papers in this issue describe the results of some of the research in this area. The first paper, "An Architecture for Multi-User

Software Development Environments,'' by Israel Ben-Shaul, Gail Kaiser, and George Heineman, focuses on the issue of synchronization in distributed software development environments. The traditional serializable transaction model often unnecessarily limits the concurrency in collaborative software development and several alternative models have recently been proposed. This paper describes a flexible architecture for supporting a variety of synchronization models in both traditional and process-centered software development environments.

The three remaining papers address systems and applications that offer real-time sharing among collaborators. The paper, "A UNIX Toolkit for Distributed Synchronous Collaborative Applications," by Dorab Patel and Scott Kalter, describes a server and library that make it easy to transform existing single-user applications to multiuser applications. An interesting feature of their system is that it does not require changes to the existing user interfaces of single-user applications. It also does not require the use of a particular window system, user-interface toolkit, or some other user-interface tool to develop the multiuser application.

The paper, "Issues in the Design of a Toolkit for Supporting Group Editors," by Knister and Prakash, focuses on system support for implementing multiuser text editors. The toolkit offers a distributed architecture that allows a multiuser text editor to be formed out of different existing single-user text editors. A novel issue addressed by the toolkit is collaborative undo.

Finally, the paper, "GroupDesign: Shared Editing in a Heterogeneous Environment," by Alain Karsenty, Christophe Tronche, and Michael Beaudouin-Lafon, describes a multiuser drawing application. The application offers novel audio-based mechanisms to make a user aware of the activities of other users. It offers a distributed architecture that uses semantic information to ensure causal broadcast.

All of these papers describe working tools and experience with them and represent the state-of-the-art in practical research in the emerging area of computer supported cooperative work. The techniques and experiences presented in these papers will be of value to future research in this area.

Gene Spafford conceived the idea for this special issue, advertised the call for papers, and helped with the selection of the reviewers and

papers. Many thanks also to the following reviewers who quickly and thoroughly refereed the papers: Nathaniel Borenstein, Rob Chandok, Steve Chapin, Keith Edwards, Saul Greenberg, Kevin Jeffay, Ralph Hill, Scott Kalter, Srinivas Kankanahalli, Scott McFaddin, Lola McGuffin, Richard Newman-Wolfe, Gary Olson, Dorab Patel, Atul Prakash, John Riedl, Tom Rodden, Sukumar Rathnam, and Honghai Shen.