Software engineering has an abundance of techniques, tools, and methods designed to improve both the software development process and the final product. Technical improvements continue to emerge and yield encouraging results. However, software isn’t simply the product of appropriate technical solutions applied in appropriate technical ways. Software is developed by people, used by people, and supports interaction among people. As such, human characteristics, behavior, and cooperation are central to practical software development.

Authors Fred Brooks and Gerald Weinberg recognized the human aspects of software development, once called “people factors,” as an important influence on software productivity over 30 years ago.1,2 Others have reiterated this thesis regularly over the intervening decades.3–5 Given the growing size and complexity of current software systems, the cooperation and coordination required to build software effectively has become as important as the human aspects of software construction. It isn’t surprising then to see a special issue focusing on cooperative and human aspects of software engineering (CHASE). Indeed, this issue comes at a time when software practitioners are paying more attention to such research. Recent changes in the software domain such as the Internet revolution, distributed and cross-cultural...
tories, team composition, the effects of managerial, workload, career choices, the importance of informal communication, expertise identification, understanding and awareness of colleagues’ actions, and the practicalities of involving users in development. In this issue, articles and regular columns focus on a representative subset of these topic areas: knowledge sharing, trust, distributed development, individual abilities, communication, self-management, and individual perceptions.

Researchers in this area draw on methods from different disciplines, including controlled experiments, surveys, and ethnographic studies, as well as code analysis and performance measurement; in fact, it’s recognized as good practice to use a group of complementary techniques to ensure triangulation. A range of analysis techniques and frameworks can also be applied including theoretical frameworks from social psychology, cognitive science, sociology, organizational behavior, and complexity theory. All these approaches have been used to study different phases of the software life cycle—from requirements elicitation to programming to maintenance tasks.

CHASE research has investigated software development activities ranging from traditional corporate software to open and free software (FLOSS), including commodity software that requires customization—for example, enterprise resource planning (ERP) systems.

It’s impossible for one special issue to include examples of all these different CHASE dimensions, so we chose articles that address different aspects of CHASE. For instance, Nils Brede Moe, Torgeir Dingsøyr, and Tore Dybå present an ethnographic study of three agile software teams that report on barriers to self-management and, more importantly, suggest how to overcome them. In contrast, Martin P. Robillard shares the results of a survey with Microsoft employees about learning to use APIs and presents general guidelines for practitioners dealing with them. Ban Al-Ani and David Redmiles, on the other hand, briefly describe a field study and discuss how their tools help promote trust among distributed developers. Two articles look at knowledge sharing and ERP systems from different perspectives. Yvonne Dittrich, Sebastien Vaucouleur, and Stephen Giff conducted an empirical study of ERP customization using a range of data collection and analysis techniques to suggest challenges and implications for tool support development, while Harris Wu and Lan

---

**About the Authors**

**Helen Sharp** is a professor of software engineering at The Open University, UK. Her main research interest focuses on understanding the social nature of software development and she has conducted qualitative studies of software practitioners for over a decade, most recently focusing on agile software development. Sharp has a PhD in computer science from University of London. Contact her at h.c.sharp@open.ac.uk.

**Li-Te Cheng** is a researcher with the Collaborative User Experience group at the IBM Center for Social Software. His research interests include exploring new tools inspired by social media and enhancing collaboration among software engineering teams. His early work informed the Rational Jazz platform’s development. Cheng has a PhD in electrical engineering from Memorial University of Newfoundland. Contact him at li-te_cheng@us.ibm.com.

**Janice Singer** works with small- and medium-sized companies in fostering innovation with the Industrial Research Assistance Program of the National Research Council of Canada. Previous to that, she was a researcher in the Software Engineering Group. Her research interests range from collaboration to individual cognition in the software engineering domain. Contact her at janice.singer@nrc-cnrc.gc.ca.

**Gina Venolia** is a senior researcher with Microsoft Research in the Human Interactions in Programming group. Her research focuses on understanding knowledge flow in software development teams and building systems to make it flow more freely. Venolia has a PhD in cognitive psychology from the University of Pittsburgh. Contact her at gina.venolia@microsoft.com.
Cao present a prototype Web 2.0 tool to support knowledge reuse in ERP implementation.

We’re pleased to include a Point/Counterpoint discussion between Dave Parnas and Bill Curtis where they debate the role of experimental and exploratory studies in software engineering—a topic frequently visited at conferences, workshops, and other forums. While Parnas argues, “It’s time to stop ‘exploring’ and start experimenting,” Curtis recognizes the valuable contributions exploratory studies can make.

We end this introduction with a question for you: when was the last time you considered the impact of human issues on you, your work colleagues, and your product? Human and cooperative aspects impact the business of software construction at many levels and from many perspectives. We urge you to pay close attention to these issues in your own work environment and share your results at future workshops and conferences, or simply within your own organization. You’ll be surprised how much these factors influence the activity and wellbeing of your team, your customer, your organization, and your community. Sharing experiences is particularly important because one challenge we face is the education of software professionals—that is, how can we convey to future developers the importance of the cooperative and human aspects of software engineering? This is a crucial question that would benefit from academic and practitioner input.

References

For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/csdl.