

▼

AT ICSE 13, DIVERSITY IS THE THEME

Galen Gruman, Editor

DIVIDED AMONG PAPERS DESCRIBING work in progress, panels on supporting technologies and overall issues, and advocacy-minded keynote addresses, the 13th International Conference on Software Engineering exemplified the diversity of the disciplines grouped as software engineering. The conference's set of multiple, narrow focuses drew complaints from some attendees during hallway conversations, but the sessions on overall issues seemed to fuel much positive discussion. ICSE 13 was held May 12-16 in Austin, Texas. About 650 people attended, down from previous years' averages of 900 to 1,000. It is cosponsored by the IEEE Computer Society and ACM.

MANAGEMENT ISSUES. Perhaps the liveliest session was a panel covering productivity and management issues. Bill Curtis of the SEI's process-assessment group ran through a litany of problems facing software firms: wide variance in developers' productivity, job in-

terviews that don't uncover applicants' skills and shortcomings, lack of career planning, inability for employees to practice what they learn in training classes, the skills lost when a project is finished and its team split up, and, most fundamental, the lack of preparation that managers get for management jobs.

And, he said, the deficiencies that software engineers have as managers begin in their student days: Software engineers are trained neither in management nor in working on team projects.

Among the solutions Curtis mentioned was active mentoring, whereby managers drawn from the engineering staff could be shown how to make the

transition to management and supported in their initial efforts.

Two other problems hinder good software management, said panelist Maurice Schlumbarger, manager of Cap Gemini Innovation: First, software managers have little training in the areas they manage because of the separate management and technical career paths that most companies have. Second, most managers don't like to be chal-

Perhaps the liveliest session was a panel covering productivity and management issues.

lenged and thus hate to build teams.

"Companies succeed to the extent that they allow communities to survive," said Panelist Tom DeMarco, a principal with the Atlantic Systems Guild. He criticized "dreadful uniformity" and urged companies to try different methods: "Each project is a pilot project in some sense." DeMarco laid the blame for uniformity and lack of communities on "fearful management."

What can developers do to encourage a feeling of community and motivate their managers to try new ideas? "Vote with your feet," DeMarco said, "That's what capitalism does best: It throws out those companies. You can do good by helping your company succeed. But you can also help by helping them fail."

DeMarco cautioned the audience not to look for magic answers elsewhere: "Don't bring in management from elsewhere. That's a disaster," he said, because other fields have their own management problems. Instead, build good managers from the software profession, he said.

PROCESS MATURITY IN JAPAN. To gauge the relative strengths of the US and Japanese software industries, a team from the Software Engineering Institute visited several Japanese firms last year to apply an abbreviated version of the SEI's process assessment to them. Despite the survey, it is hard to directly compare maturity levels in Japan and the US, warned David Kitson, an SEI researcher, because Japanese industry is different than that in the US. Japanese firms strongly focus on custom application programming and dominance by hardware manufacturers.

Another key difference is that the Japanese firms surveyed were largely MIS-oriented, while those surveyed in the US focus on embedded command and control systems. The Japanese firms surveyed also tended to write smaller programs than the US firms surveyed; most projects resulted in less than 200,000 lines of code.

Despite the differences, firms surveyed in both nations clustered around the lowest level in the five-level SEI model. But the US firms surveyed had 10 times as many at level 2.

"In both countries, the state of the prac-

tice is pretty poor," said Watts Humphrey, who developed the SEI's assessment program. From the limited data on US firms doing the same type of Cobol-oriented work as the Japanese firms surveyed, "they're in the same soup," he said.

In the long-term advantage, the Japanese people-oriented focus may result in higher competitiveness against tool- and system-oriented US firms, Humphrey said. But in the short term, US firms have an advantage because the Japanese are faced with a huge programmer shortfall, he said.

ETHICAL CONCERNS. Most software engineers will not deal with catastrophic ethical concerns, but they nonetheless face ethical issues each day, said Donald Gotterbarn, a computer and information science professor at Eastern Tennessee State University, in a panel on ethical issues. For software engineers, the problem is basic: "Implementation concerns override the users," he said.

He likened a software engineer's view of developing a system to that of someone doing a crossword puzzle: "What do you do when you finish a crossword puzzle? You throw it away." Software engineers forget that once they are done working out that puzzle, people will be using the results, he said. "Meet the technical specs *and* pay attention to the user," he urged.

Not everyone on the panel agreed that the engineer has such an ethical responsibility. Frank Marchelina, manager of the Patriot missile software at Raytheon, said three people share that responsibility: the

customer, the company, and the engineer, with the first two having the primary responsibility. "The company must provide the ethical standards" and audit compliance, he said.

"We need a shared vision of professional responsibility," said panelist Keith Miller, a computer-science professor at William and Mary College in Virginia. Although there are no magic answers for ethical questions, he said, software engineers can follow a basic process in addressing them: Try to evaluate all those who are affected, try to determine the effects on them, describe these effects as negative or positive, and weigh the results.

Ethics and good business go hand in hand, said panelist Carl Skoogland, ethics director at Texas Instruments. "Requirements are not just strictly documented specifications, they should also be fitness for use," he said. Such requirements may outweigh the customer's requirements, he said.

SOFTWARE-ENGINEERING EDUCATION. "Application research needs to be made academically respectable," said Barry Boehm, director of DARPA's Information Science and Technology Office, in a panel dedicated to trends and needs in software engineering. Boehm pointed out that the artificial-intelligence community "has managed to do this." He also urged the development of domain-specific architectures for software, rather than continuing to let hardware architectures determine software constraints and options.

"Computer-science and MIS graduates need to do more than know how to write compilers," said Bill Sasso of Andersen Consulting. "The key is making the programs more relevant," he said, by adding a problem-solving orientation, experience in working with teams, understanding of project and

risk management, and knowing how to transfer technology to users. For its part, "industry needs to reward, value, and expect these skills," he said.

"We start training people as programmers, then we wonder why they aren't software designers," admonished Herbert Weber, a computer-science professor at the University of Dortmund in Germany. "As educators, maybe we should stop teaching programming," he said.

USING FORMAL METHODS. Although its basis is implicit in early papers by John von Neumann and Alan Turing, formal methods became divorced from testing and thus did not mature as part of software engineering, said Susan Gerhart, who leads a formal-methods project at MCC. Only with government interest in the 1980s did formal methods become an accepted research area, she said.

Despite that acceptance, formal methods face skepticism in the commercial arena that "they are too hard to use," Gerhart said. Furthermore, the emphasis on their application to safety has caused many developers to think they are not applicable to other areas, she said.

Gerhart tried to counter these impressions by reporting on the strong work in Europe on formal methods and by presenting formal methods as a set of techniques of varying difficulty that can be tailored to commercial projects. "The basic math is at the freshman and sophomore levels, not the PhD," she said, requiring an understanding of discrete structures and logic coupled with induction skills and abstraction.

In a keynote address, Anthony Hall of Praxis Systems in Britain stressed the ability of working engineers to apply such methods on real projects by describing the use of formal methods at Praxis in the specification and design of an air-traffic control system.

The specifications did not tell the designers how to write their code, only what the system had to do and not do, Hall said. These specifications provided "clear definitions for implementers" and were a source for system tests. They also provided a basis for contracts with the customer.

New directions for formal-methods researchers include the development of a sociology of formal methods that lets developers negotiate on mathematics (for example, agreeing on the handling of floating points) and the application of formal methods to reverse engineering under a formal reuse framework, Gerhart said.