SOFT NEWS

Contact: Galen Gruman
IEEE Software
10662 Los Vaqueros Cir.
Los Alamitos, CA 90720
Compmail+: soft.one
Sourcemail: cpa183

Soft News covers the social, political, and economic effects of software research, development, and application. How is the technology applied? How does it change people's lives? What legislation and other government actions affect the software professional? How? How do software professionals react? How much money is there for software research? Who is spending it? On what? The Soft News thrust is to show what the technology discussed elsewhere in the magazine is doing to the world at large. Your input — through letters, tips, and conference reports — is vital to Soft News' success.

Process programming: A seductive danger?

Galen Gruman, Assistant Editor

"We now have the second software crisis: maintenance," said Bob Balzer of the University of Southern California as he opened the March 31-April 2 Software Engineering Conference in Monterey, Calif. Software has undergone a major shift from a product in an assembly line to the recognition that software is a process, the conference cochairman argued. "We need to tackle maintenance not only in the maintenance phase but also in design," he said.

"We're moving from a mathematical approach to recognition that software engineering is a design process — an activity of many people over a long time," Balzer said. "The waterfall model is dead. What will replace it?" Balzer asked the 1000 people attending the opening session.

Process programming might replace it, and that proposition caused a strong debate between the conference's plenary speakers and evoked strong reaction from the audience. The opinions were strongly divided. Recognizing the divisions, the conference chairmen set the opening session up as a two-speaker presentation.

One speaker described the promise of process programming, while the other warned that the idea was too seductive for students to pursue because it might lead them astray. A member of the audience decried the focus on process programming, calling the approach dehumanizing.

Software design as software. "Software processes are software, too," argued Leon Osterweil of the University of Colorado at Boulder. The process approach seeks to automate and formalize the software process — treating the software design process as if it were a software program as yet unwritten, he said. The task is to "start with a process description and end up with a problem solver," Osterweil said.

"Software is an aggregate of information and objects tied together in complicated ways. The software product is a complex, messy thing. [But] we have little choice. So how do we do it?" Osterweil asked. "At best, this process is very informally specified," he said.

"Think of it as a data aggregate. You're trying to build a product that is the process of software development. The application domain is the domain of solving software problems," Osterweil said. "Why not program software development? And, dare I say it, with a programming language?" he asked.

"I do not believe you can replace software engineers. But we will change to program portions being executed by programs, and humans — managers — must decide who does what," Osterweil predicted. "We're creating a description that creates another description that goes off and solves the problem," he said.

To be successful, "we must make the process explicit. All too often the process can never be made tangible — it's not reusable. When that person [the designer] dies or is promoted, the effect is the same: The process is lost," Osterweil argued, so the description must be made rigorous.

His early process programming work was an attempt to model the software life cycle, Osterweil said. The waterfall-paradigm code developed had no control flow, no concurrency, and no parallelism, "but going through the effort of formalizing makes the technology apparent and the side effects more obvious," he explained.

But process programming requires more than rigorous descriptions, Osterweil warned. "You have to address not only the procedural aspects but also be very specific about the data. These are the stuff and things of the product," he said.

"Everyone in this room deals with these problems between their ears all the time — but not with a formal process description," Osterweil argued. "Maintenance is a process, too," he added. "A major point of maintenance is analyzing what may be reanalyzed. But usually it is all specified inside the head," Osterweil said.

To make process programming a reality, researchers must develop and use several tools:

- A programming language. "I suspect we'll come up with a specification for a language that doesn't exist," Osterweil said.
- Databases. "Software engineering can be greatly helped by the right database," he said.
- Artificial intelligence techniques. "We need to encapsulate what we do in modules. Perhaps we could use incremental sets of rules to handle problems and keep design as an activity bound to a human with rule-based help," Osterweil said.
- Interface management. "Binding to humans means it will be handled like debugging: tools to help users," he said.

"Software engineering is *not* the intersection of these disciplines, *not* a manifestation of these fields," Osterweil argued. "It is the software process itself. It is the study of problem-solving," he said.

Dangers seen. "The very act of pursuing the process influences the process. There is no such thing as a static program. The process is a function of itself," argued Meir "Manny" Lehman of the Imperial College of Science and Technology in London. "The interac-

tions are totally unpredictable. *People* have to decide what is correct — correctness is not an absolute concept even in mathematics," he said. It is a human interpretation that varies from person to person and situation to situation, he explained.

"The essence of a programming language is that underlying every verb is an interpretive mechanism that completely defines what the process is. Using verbs, descriptions of this type suggest a determinism that does not exist — that cannot exist," Lehman said. "We can never build machines that act like human minds," he said.

"The programs that we can make deal with the easy parts. The problems that arise are the ones that cannot be described," Lehman asserted.

Process programming is a good modeling technique, he conceded. If a step is algorithmic, it can be mechanized, Lehman said. Process programming "is one component," he said, an important part of formal specifications.

"[But] it represents a very real threat," Lehman warned. For example, "if you use natural language, you think you've defined the spec. But you really haven't since you cannot have a total definition," he said.

While process programming is "good for well-modeled, well-understood parts — the ones we can mechanize," it misses a significant point, Lehman argued: "Fundamental to programming is not knowledge, it is understanding." Process programming "is likely to convey a false sense of understanding," he said.

"It has a seductive value that will lead to a waste of efforts," Lehman warned. It requires self-control before use, he said. "Because it is so seductive, process programming is not appropriate for student research or PhD activity," Lehman argued.

Debate. "The truth is somewhere in between" Osterweil's and Lehman's views, Balzer said. "The essence of the technique now is to separate the straightforward pieces [of the programming process] and put it in a mechanical realm. This will focus human activity on the hard things," Osterweil responded to Lehman's critique.

Harlan Mills of IBM said from the floor that "I regard software engineering as an intellectual activity. I see here [in process programming] an anti-intellectual revolution." The audience clapped in support. Mills decried the absence of computer science giants like David Gries, Fred Brooks, Anthony Hoare, and Edsgar Dijkstra: "We're freezing out the intellectual forces out

NEWS **INTERVIEW**

Parallel processing's future dim, Unix's bright

While multiprocessing environments like Unix are necessary but controllable evils, parallel processors are something to avoid because they are uncontrollable and cannot be exploited well, said Unix creator Ken Thompson in an interview published in the winter 1987 special Unix issue of Computer Literacy Bookshops' *New Book Bulletin*. Thompson, an AT&T Bell Labs researcher, now works on C and System VIII Unix.

He also said he thinks that Unix will remain strong for years and that the proposed C standard is generally on the right track. Personal computers, however, are expensive and inefficient, he said. Excerpts from the interview are reprinted below.

Q: *You don't believe that parallel processing is the next big thing, the wave of the future?*
A: Multiprocessors are a necessary evil; you should try to keep things simple so it looks like a single thread. The pipes and windows in Unix are controllable. Parallelism is something to avoid; you can't control it, and you can't exploit it well. You keep getting closer and closer, but you never do get all the bugs out!

It's almost a fad: people running out and building non-von Neumann machines, then trying to control them later. Two processors on the same job with dynamic access to variables — that's a truly impossible debugging job! I've never seen a good multiprocessor language, or a good multiprocessor debugger.

There are some applications, like modeling the weather, where you may need this approach — cases where you want to divide the world into little boxes, then have all the boxes report to each other on boundary conditions. This will never be the mainstream of computing, though. It's not the future.

Q: *So you see the future as more of a traditional time-sharing approach on faster and faster hardware.*
A: Yes, I do.

Q: *Unix began in 1969 as something of a "hack" to use an available DEC PDP-7. It's over 15 years old now, a major force in the computing world. Looking back, would you have done anything different with the design?*
A: I did the best job I could at the time. There were five, six, maybe seven versions that no one ever saw — total rewrites. I maintained control of Unix through 1975; no one put any code into it without my OK.

Q: *Was it hard to give up that personal control?*
A: [Laughing] Not hard at all! I was glad to get out from under it. I took a year off and went back to Berkeley to teach for a year.

Q: *In a recent* Unix Review *interview, Sun's Eric Schmidt referred to Unix and C as "the Fortran of the 1980s," meaning that their large user bases and relative standardization would keep them outmoded but alive long past their prime. Is Unix becoming a dinosaur?*
A: It depends on what you mean by "Unix." If you're talking about the code running the kernel, it's crusty. I've been critical of that for years: It's gotten big and clumsy. Everyone's been dumping stuff into it for years. As code for single a CPU, it's too time-share-oriented to last. Crusty implementations won't survive, and they shouldn't.

If, on the other hand, you're talking about Unix as a system-level interface, a vehicle for portable programming, the design is clean and can last forever. Unix will adapt.

Q: *Do you like the emerging ANSI C standard?*
A: Generally, I applaud the committee's efforts, though I do have qualms about some particulars. To provide stronger typing, for example, they added C++ syntax for describing arguments. I believe that standards should stick to existing de facto standards. I think in some cases they went too far in designing a new language.

Personally, I'm going to sit back and see what happens.

of this conference. What is the alternative to making software engineering an intellectual activity?"

"I worry when it is too mathematical *and* too humanistic. You must balance them. That is the essence of our discipline," Osterweil responded. "We can get more insight on the intellectual questions by segregating out what is not intellectual," he argued.

"Some process programming is based on a very naive view of human processes. Twenty VAXs will return the same answer; 20 programmers with the same degree of experience will not," said MCC's Bill Curtis from the audience. "We need to give more than a process instantiation. We need an empirical approach," he argued.

"You can take unmasked input from the real world and . . . apply the process to that data and decide what to do with it." Osterweil responded.

Information policy partially rescinded

Galen Gruman, Assistant Editor

The federal government rescinded in March part of its policy on access to government databases by foreigners after the IEEE and Information Industry Association testified against the policy in Congress in February.

The government's intent was to prevent the information from being used by foreigners, especially Soviet-bloc citizens, according to several Defense Dept. reports. However, opponents argued that the policy's scope was so broad that it could abridge Americans' rights to freedom of speech and information.

Proposals included creating a new category of information for material that is unclassified but "sensitive," access-denial functions in federal databases to prevent foreigners' access, and "hav[ing] some automated tools for analyzing the information that is in both commercial information systems and also in federal government information systems," said Diane Fountaine, director of information systems for the Defense Dept., in a speech to the Information Industry Association last fall.

Databases have the same First Amendment rights as the printed press, and information should be either classified or unclassified, not in an uncertain category between the two, argued David Peyton, director of government relations for the Information Industry Association, a group of several hundred information-using or information-generating businesses.

An Oct. 29 memo from former National Security Adviser John Poindexter, who left the administration after

---

AT&T is probably going to adopt and use the new standard. Even so, for something to become a standard, it needs a user community and compiler manufacturers who will embrace it. Writing something on paper won't make it a standard.

Q: *What do you think of C++?*
A: I've played with it a fair amount, but I don't dare try anything serious with it: Bjarne Stroustrup [its developer] works on my system and he's constantly changing the compiler! I'd be afraid my code would bomb due to one of his changes.

Q: *How about some of the other newer languages? Any interest in Modula-2?*
A: I try to take a look at most of the newer languages, but they don't really offer me anything in my environment.

Q: *What is your current software development environment?*
A: Primarily System VIII Unix, Bell Labs' internal version of Unix, and C.

Q: *Have you had much experience with microcomputers?*
A: Overall, I think the software is really terrible, though some of the software written for the Macintosh is a little better. . . . Spreadsheets have been the only real software advance attributable to personal computers. They brought a whole new segment of people — managers — into computing.

I'm not happy about the current de facto standard for distributed computing; for example, workstations like the Sun in every office connected to remote file servers via Ethernet links. That's not a criticism of Sun per se but of the whole conceptual method.

I think this model has prevailed this long because, like PCs a few years ago, the cost of buying Suns is just about the amount of money a manager can work into the budget without making too many waves internally. There's no doubt that people like the machines; they're predictable and they're yours. Still, it's a very expensive and inefficient model of computing.

Q: *Specifically, what don't you like about them?*
A: In addition to hardware maintenance problems when a machine breaks down, there's a huge software maintenance problem; it's too hard to keep all the machines current with the latest version of whatever software they're using.

Ethernet is barely justifiable electrically — it barely works.

Also, I don't like the office environments they create: noise from cooling fans, excess heat, [floppy disks] everywhere, cable running through the office. The machines are big and, I think, poorly designed.

Q: *What's the alternative?*
A: I think the way to go is to have centrally owned and maintained clusters of multiprocessors, high-powered CPUs, with shared memory. It's cheaper, avoids the maintenance problems, and gives you the multiprocessor computing power whenever you need it.

Computer Literacy is the US's largest bookstore devoted to computer and electronics books. In addition to their retail business in California's Silicon Valley, Computer Literacy provides mail-order book services throughout the world. It also publishes New Book Bulletin *bimonthly, reviewing just-released computer and electrical engineering books. To receive free bulletins and get book information, contact Computer Literacy Bookshops, 2590 N. First St., San Jose, CA 95131; (408) 435-1118.*

the recent arms sales to Iran were disclosed, broadly defined what might be considered sensitive information. The policy in that memo was rescinded by new National Security Adviser Frank Carlucci after the Congressional testimony in March.

"National security interests are those that unclassified matters that relate to the national defense or the foreign relations of the US government. Other government interests are those related, but not limited to, the wide range of government or government-derived economic, human, financial, industrial, agricultural, technological, and law-enforcement information, as well as the privacy or confidentiality of personal or commercial proprietary information provided to the US government by its citizens," the memo said.

The Poindexter definition of sensitive information covers "everything," Peyton said. "It's so broad that it could cover every government information system," he said.

Association president Paul Zurkowski later called the rescission "a giant step in the right direction." "The most objectionable part has been removed," said John Richardson, who testified against the policy on behalf of the IEEE's Committee on Communications and Information Policy.

However, the status of the 1984 presidential decision, National Security Decision Directive 145, that set the groundwork for the database policy remains unclear. A Defense Dept. official said that the policy seemed to be effectively rescinded, at least until a review by the National Security Council of the directive. Council staffers confirmed that the directive was under review but would not comment on specifics.

Congress is now considering legislation to protect electronically stored private information. If passed, the Computer Security Act of 1987 (H.R. 145) would give the Commerce and State departments the responsibility for protecting information. The National Bureau of Standards would be responsible for setting the computer-security standards. The administration plans assigned the task to the National Security Agency and Defense Dept.

"We endorse the intent of H.R. 145 that a civilian agency rather than a defense agency should assist other civilian agencies and the private sector in the protection of computer systems," IEEE's Richardson testified Feb. 25.

Richardson criticized both the administration's and the proposed law's definitions of sensitive information as "too broad."

Developers expect DOS to diverge

Galen Gruman, Assistant Editor

"What will Microsoft and IBM do to DOS?" asked those who attended the West Coast Computer Faire in San Francisco March 26-29. IBM has since announced its next-generation operating system (see p. 86), but Microsoft remains quiet about its plans for MS-DOS, the most-used operating system in personal computers.

The new microprocessors and the recent availability of graphical operating environments promise (some say threaten) to split the IBM-compatible world between 8088- and 80286-based PC XT-like computers and 80386-based multitasking machines and between traditional command-driven interfaces and graphical environments.

The computer fair devoted two sessions to the question, but discussion continued on the exhibit floor and in sessions on other topics.

The question of what DOS to develop for repeatedly occurred in a session on RAM-resident programs. "The IBM PC and the clones were never developed to be what is becoming a multitasking environment," said Bill Higgs, director of software research services for Info Corp., a Cupertino, Calif., company. Despite Microsoft's assurances that its new MS-DOS (variously called ADOS, DOS 5.0 and New DOS) will handle memory conflicts between resident programs, software developers will find new ways to create problems, he said, and will have to wait a couple years before the new DOS is available, much less a standard.

While environments like Windows were offered as potential replacements for MS-DOS, "the world's going to be living in the current DOS world for quite some time," said Spenser Leyton, vice president of sales and business development at Borland International.

However, because Microsoft is pushing its Windows environment while also working on a new DOS to address the 80386's protected mode, break the 640K-byte addressable memory limit, and allow multitasking, panelists were divided about which operating system developers would have to write their applications for.

Because Microsoft is keeping quiet about its plans and because it is involved in the various potential divisions, speakers and attendees alike could only agree that the current MS-DOS environment would probably fracture into several subenvironments.