The Second Annual Computer Software and Applications Conference, held November 13-16 in Chicago, drew 715 attendees to consider aspects of software design, quality assurance, and project management.

Ware Myers
Computer Staff

Marks challenges software community to solve its problems

"The Department of Defense needs your help and understanding in dealing with its software problems," Hans Marks, undersecretary of the Air Force, told the opening session of COMPSAC 78 November 14 at the Palmer House in Chicago's loop. "There is one area in particular where we have serious conceptual problems—managers knowing what they are doing."

Marks drew upon information provided by his colleague, Dr. Ruth Davis, an assistant secretary, to put the problem in perspective. Technological developments, especially in the past 15 years, have dramatically reduced the cost of computer circuitry and increased hardware capability. A microprocessor costing about $20 today has the computing power of a large computer costing approximately $1 million 20 years ago. However, the cost of software has shown the opposite trend. The estimated cost of software development, testing, and maintenance for the entire federal government in 1978 is about $8 billion. Defense spends about $4 billion in this area. In the Air Force, Marks pointed out, about 80 to 90 percent of computer systems acquisition cost goes for software. The government owns something like $25 billion of currently employed software.

Software costs greatly exceed equipment costs over the life of the service system. Software is the component of a computer system with the most risk in development. Overruns on the order of 100 percent of both cost and time to develop software have not been unusual.

In fact, there have been cases of total failure to develop a system as planned—one of the unpleasant surprises that Marks has encountered during his short time in Washington. (He was formerly director of NASA Ames Research Center in California.)

Even after a software system has been developed, however, proving its correctness "remains a most elusive
goal." People always show surprise when told that there is no theoretical or mathematically rigorous way to prove a program correct, except for trivial ones containing, say, less than 100 statements. Moreover, this limitation will be hard to overcome because of the mathematical difficulty of constructing the necessary inductive assertions and the cost of the computer time to generate the proofs. Exhau stive testing is impractical—it would take the fastest machine available more than 30,000 years to try all the inputs to a simple multiplication program, for example.

From the Air Force's point of view, the potential software buyer is unguided and vulnerable in this uncertain and complex marketplace. First of all, he has trouble locating the type of software he needs. Once located, he has few criteria for measuring its performance or efficiency. He cannot evaluate the software product's features because of the lack of documentation standards. He finds it difficult to tell if it will be correct and reliable. He isn't sure if it will be easily maintainable and transferable, and finally he doesn't know exactly how much it will cost.

Defense finds itself in this rather unfortunate situation because it must make large investments in control equipment, with most of the money going for software. Yet it is software-controlled equipment that enables the United States to maintain its military capability without even more enormous investments in hardware and frequent hardware changes. This investment strategy makes sense, Marks said, "only if we can be sure the products work."

He isn't sure of that now. And that was the nature of the challenge with which COMPSAC participants entered three days of papers, panels, and discussion.

Architecture to support programming concepts

To combat the trend of increasing software costs, technical keynoter Per Brinch Hansen of USC suggested the use of abstract programming languages. Standing in the way, however, is the unfortunate fact that present computer architectures do not support abstract language as efficiently as they do machine language. The solution is to build architectures that support these programming concepts directly, as stack computers do the block and procedure concepts of Algol 60.

Microprocessor technology now makes it possible to create an architecture that will support directly the process and monitor concepts of concurrent programming. In fact, Brinch Hansen has proposed a 10-microprocessor system, each processor having a local store dedicated to a single process. The 10 processors share a common store that contains the monitors. There are no interrupts and the processors are not multiplexed among several processes.

He expects an increasing number of computer architectures to support concurrent programming languages for real-time applications. In applications of wide interest, it will be economical to write a concurrent program in an abstract language that hides machine details, test the program on an existing machine, and finally, working backward, derive the most straightforward specialized architecture from the program itself.

The development of architectures for concurrent programming is likely to be the work of a decade, partly because theoretical understanding of concurrency is in its infancy. This understanding depends, in part, on the development of notation.

Notation helps discover meaning. With good notation it becomes possible to further refine a concept and get a more formal understanding of its properties, Brinch Hansen observed. He quoted Suzanne Langer: "There is something uncanny about the power of a happily chosen ideographic language; for it often allows one to express relations which have no names in natural language and therefore have never been noticed by anyone. Symbolism, then, becomes an organ of discovery rather than mere notation."

A. S. Hoagland of IBM, AFIPS president and a past Computer Society president, discussed mass storage price/performance.

A. P. Ershov, of the USSR Academy of Sciences, explained how he uses notation to clarify software system relationships.

Technical keynoter Per Brinch Hansen of USC suggested that abstract programming languages can combat rising software costs.
A taste for the use of notation to clarify software system relationships was provided by keynoter A. P. Ershov of the Academy of Sciences, Novosibirsk, USSR. Ershov is trying to develop a system of notation and relationships enabling him to freeze all but one aspect of a large software design, so he can study that aspect. Then he can unfreeze it and freeze another for study, and so on. His idea seems to be that one can examine a part in detail, but not in complete isolation, because the other parts are still there, though frozen.

Ershov, calling his approach "the golden key of mixed computation," showed only its basic principles since it is not ready for practical use. His goal is to use his method to develop a compiler. It might also prove useful, however, in the design of distributed software or any large software system beyond the grasp of one mind at one time.

Networks next challenge. Summing up the challenges and responses of the last 20 years, Brinch Hansen said the result was reliable computer systems in which concurrent processes share storage. The microcomputer implies a network in which processors communicate with each other only over their input/output channels; they have no common storage. Message passing might seem simple but, according to Brinch Hansen, when all its aspects are taken into consideration it is not easy. Several proposals have been made, but their practicality remains to be established.

In fact, this new hardware challenge—networked micros—may lead to a software crisis, repeating the history of the past 20 years. A search for the concepts, languages, and theory needed to resolve such a crisis will start again. Brinch Hansen expects this process to take a long time. "I would expect distributed computing to be reasonably well understood by the year 2000."

Computers vs. communications

Another keynote speaker, Robert W. Lucky of Bell Labs, Murray Hill, New Jersey, addressed the network idea from the communications utility point of view. His group recently bought a Tandem system consisting of a number of microprocessors connected by a packet communications system. He knows this system is a computer because it says so on the nameplate!

Early this year AT&T petitioned the FCC for approval of its Advanced Communications System. Lucky said it consists of a packet communications system and processing nodes. He knows it is a communications system, because that is what it is called! The Advanced Communications System offers capabilities such as conversion from code to code, speed matching, protocol translation, data formatting, and storage. AT&T believes these are communications tasks. It maintains the real test is whether the "substantive content" of the message is altered. The proposed system will "effectuate the movement of messages pursuant to the customer's direction without altering the substantive content of the message." Thus it is a communications system.

That, however, may not be the last word—Washington will have the final say. Looking at the question as a technologist, Lucky thinks it is very difficult to decide what is communications and what is processing. Perhaps the important question for the technologist is whether to do these things at the network's center or periphery.

Lucky had a lot of fun imagining fiber optics blanketing the country and reducing the actual transmission cost of messages essentially to zero. Unfortunately, nobody sees a way to make local access to the telephone net really cheap. Besides that, a lot of wires are already in place, and the Bell system still has about a million employees doing all sorts of necessary things. Lucky concluded that the cost of a data communications channel will probably not dramatically decrease in the near future; more services will instead be offered per unit of voice channel.
At plenary session, outgoing President Merlin Smith (top left) presents Honor Roll Award to incoming President Tseyun Feng, for outstanding service as chairman of the society's Computer Standards Committee. At top right Gordon E. Moore, co-founder of Intel, receives the McDowell Award for his contributions to MOS technology. At center Smith presents Stephen S. Yau with a Special Award for his efforts in establishing COMPSAC. COMPSAC Standing Committee Chairman Yau also headed the 1978 conference's Tutorial Program Committee. Albert L. Hopkins, Jr., (right center), accepted an Honor Roll Award for his service to society operations and governance. At bottom left Smith presents Special Award to Jack Grimes for serving as Computer's technical editor. At bottom right David H. Jacobsohn receives an Honor Roll Award for chairing society operations. A Certificate of Appreciation went to Gerrie Katz, not shown, of the society's Administrative Office, for her support of conference activities.
COMPSAC 78 featured three pre-conference tutorials, drawing 289 attendees. At top, C. V. Ramamoorthy (left), of the University of California at Berkeley, and Raymond T. Yeh, of the University of Texas at Austin, assessed the current status of software development methodology. At center David W. Shipman (left) and James B. Rothnie, Jr., Computer Corporation of America, and Philip A. Bernstein (bottom left), of CCA and Harvard University, offered a survey of distributed data-base management. Im-song Lee (bottom center) and Roger Doering (bottom right) of Digital Electronics Corporation examined microcomputer applications and support software. Their tutorial featured an evening "hands on" laboratory session. Texts of these tutorials as well as the COMPSAC 78 Proceedings are available from The Bookshelf, p. 107.
Price/performance impacts applications

The decreasing cost of mass storage allows us to implement applications that were not previously cost-effective. That, according to IBM's A. S. Hoagland, president of the American Federation of Information Processing Societies, is the impact of mass memory technology on software applications. Hoagland was one of the four technical keynotes at COMPSAC.

Still, there will have to be many more applications if memory developers are to make enough profit to continue to improve price/performance. With a larger market manufacturers achieve economies of scale that feed back to further decrease the cost per bit. But part of the increasing income is fed back into more research and development, leading to higher technology and still greater storage density. Meantime other companies recognize that they must follow suit. It is a cyclic system which feeds on itself.

Users have an interest in the decreasing cost per bit, but they would also like a competitive marketplace in which there are many suppliers and many offerings. Technically they would like a store which can be written and read and is nonvolatile, according to Hoagland.

Manufacturers have a different perception of the situation. They like products that can grow into major businesses and that lend themselves to established production processes. Realizing that technology tends to advance, manufacturers would like to see the market advance faster than the technology. Unfortunately, bit density is currently increasing by a factor of two every other year, while system capacity in increasing only by a factor of 40 percent per year compounded. That does not offer a favorable path to the manufacturer or the the group considering entering the business, Hoagland believes.

Engineers and researchers are motivated to work on a technology that still has a long way to go. At the same time they would like to have the phenomena of the field fairly well understood, so they can get products out on schedule. They would like a process which is reversible and stable.

Magnetic recording fits bill. Hoagland believes that magnetic recording—disk and tape—meets these criteria. Because there is so much talk of developments in semiconductor, CCD, magnetic bubble, and even video disk memories, he put his comparisons in the context of semiconductor technology ("they") vs. magnetic recording technology ("we"): (1) Where they use three-inch wafers and are moving toward four-inch wafers, we have been using 14-inch wafers for a number of years. (2) Where they cut up their wafers into little chips and hope to get a reasonable yield from them, we just take the whole wafer at one time—disk capacity is about 10,000 chips. (3) Where they talk about different kinds of lithography, are pushing to get 3-micron bubbles, and hope to get down to 2- or 3-micron line width some day, we have bit lengths of 2.5 microns and are moving toward 1 micron. Spacing, the key dimension in recording, is on the order of a tenth of a micron.

Hoagland admitted one disadvantage in this rather idyllic picture. With anything involving mechanical motion, the access time will be relatively long. Sometimes mechanical motion is condemned out of hand by the supporters of semiconductor approaches to storage. But the real issues are the tradeoff between cost per bit and access time, and the reliability of the various storage methods.

For the 1980's Hoagland predicted several thousand million bytes on a spindle and densities approaching $10^8$ bits per square inch. In the past 20 years the density of disk files has gone up by a factor of 1500 or more. Hoagland feels that this rate of advance has not been out of line with that in semiconductors.

The way technology is going, the computer itself gets smaller and smaller. Our children, Hoagland said, are going to think of a computer center as a nest of tape drives and disks. The chips making up the computer will probably be somewhere in the drive cabinets.

Panel defines 20 software problems, cites solutions

In a COMPSAC panel on software engineering project management, held November 15, Air Force Colonel Richard H. Thayer defined a set of 20 major software problems or issues (see Table 1 on page 68). Thayer derived his list from a search of the literature since 1974 and then confirmed it with a questionnaire to project managers. The number one issue—with 100 percent agreement—is requirements specification. Second is delivery time estimation, followed by planning and resource estimation. If fewer than 30 percent of the managers felt an item was important, Thayer dropped it. As part of his study, he found that 47 percent of projects are delivered late—not just a little late, but sometimes as much as 100 to 150 percent late.

Note in Table 1 that Thayer does not say there are no software engineering or project management methods. Rather he is more subtle—there are no rules for selecting pertinent methods, or use of them is "poor," or quantitative answers are lacking.

People insecure in early stages. Analyzing Thayer's 20 problems in terms of the votes for each, V. R. Basili of the University of Maryland found that they broke out in five categories: requirements definition, management estimating and planning, product assessment (testing, reliability, maintainability, etc.), people assessment (manager selection, productivity, etc.), and process techniques. He drew the conclusion that people are more insecure in the early stages of the life-cycle process—in requirements definition and estimating—than they are in later stages—the actual code development. In between are their concerns about assessing the final product.

Are tools available? There are a whole series of techniques for requirements generation and they are very effective, according to Basili. One such method, SREM—software requirements engineering methodology—developed by TRW for the Ballistic Missile Defense Advanced Technology Center, has now spread to nine other organizations. And yet Thayer's respondents rank requirements specification the number one problem!

Similarly, Basili said there are at least 10 models for cost and time estimating, some of which are being tested at the Software Engineering Laboratory of the University of Maryland (sponsored by NASA Goddard Space Flight Center). The Putnam model, based on the Rayleigh curve (described in the December 1978 Computer), came within seven percent of actual Goddard data on relatively small projects, Basili reported.
Several reliability models are showing good results in product assessment. Walston and Felix have published results on productivity measures. Various methods which quantitatively assess whether a software product satisfies certain criteria are described in the literature. Without going into detail, Basili made the point that work has in fact been done on most of the problems set forth by Thayer. Sixty-three of the 147 COMPSAC 78 papers had something to do with this list of problems, as shown in Table 2.

"We must decide what we can have and what we can’t have," Basili went on. "Are we talking about perfect software or software that can satisfy some MTTF requirement? Because I use a requirements analyzer, am I guaranteed correct results?"

In the Thayer survey some respondents thought they had a particular problem and others didn’t. Logically, one possibility is that the ones who didn’t had solved the problem in their particular environment, although the second possibility is that they didn’t know they had the problem. In other cases a particular solution is seen as too expensive for the environment. Sometimes a person may know how to solve a problem in theory, but not in his particular environment.

This last seems to be the case for several members of the audience who commented from the floor. One claimed that program managers themselves do not understand the processes of program development and, therefore, they are not willing to put their resources at the stage of the process where they are needed. Rather than put money in the early stages of requirements definition and design, they keep expecting finished code. Another respondent, a project manager, said that higher management does not understand the software process—that in fact his program manager is not yet know that software is "engineerable." Colonel Thayer, as chairman, closed the loop by observing that "the programmer too, perhaps, doesn't know how to use these methods.”

All these circumstances are summed up in the term "technology transfer." Basili pointed out that transfer of expertise takes a lot of effort. Training, retraining, and the documentation of training are expensive, and so is the data collection and measurement needed to find out whether the transfer is successful.

**Effecting technology transfer.**

Teaching recent technological developments is no easy matter. The panel member A. B. Pyster of the University of California at Santa Barbara declared. A faculty member can do it in his own field, he noted, but he usually has to teach more areas than just his own specialty. In these areas he has to depend on someone else to bring the scattered papers together and organize a good teaching sequence.

Often that means waiting for a textbook that may be three or four years behind developments anyway. Pyster suggested that one solution may be to bring in experts from industry for short courses or as visiting professors.

There is also a "jurisdictional" problem in software engineering education. It is clear, Pyster said, that the technical aspects of the subject belong in the computer science department.
But where do the primarily managerial problems on Thayer’s list belong? In the school of business? That sounds like university budget makers have to be educated in the software issues too!

“Technology development is fruitless without the mechanism for its transfer to others working in the field,” M.W. Alford told a COMPSAC session in a paper celebrating the second anniversary of the introduction of SREM and REV5—requirements engineering and validation system—to the software engineering community. Three techniques have been used to transfer this technology:

1. documentation of procedures, language, and software capabilities (used by another TRW location and the University of California at Berkeley);
2. short courses in requirements engineering, taught by the developers and given to Hughes Aircraft, McDonnell Douglas Astronautics, RCA, Teledyne Brown Engineering, TRW units, and Army Computer Systems Command;
3. on-the-job training, given by the developers to the Applied Physics Laboratory.

“Although continual on-the-job training is the ideal, the training course approach was found to be an effective and cost-effective mechanism for technology transfer,” Alford concluded.

Money talks. “If the Defense Department had a $3 billion hardware problem, the people responsible would be ridiculed if they put any less than $50 to $100 million into trying to solve it,” said E. C. Nelson of TRW, the fourth member of the Thayer panel. In software, on the other hand, he felt that the Air Force regards even $5 million for advanced software development as excessive, since they recently reduced funds for this purpose to $3.5 million. This reduction will hinder the development of necessary tools and techniques, Nelson contended.

Similarly, the ability to maintain software later in its life is often neglected during development. One of the reasons for this neglect is funding, Nelson said. “People say all the good words, but where they put their money tells the software engineer what they actually mean.”

In addition, Nelson observed that few managers have felt able to carry
out software reliability measurement on an actual project "because it costs a little bit."

Software managers criticized. In Thayer's survey software quality was far down the list. Nelson interpreted this response to mean that managers had so much difficulty just getting their projects completed that they paid little attention to the idea of quality.

And then there is the tendency of managers to be too conservative in their use of tools. But conservatism isn't always the way to minimize risk, as Nelson pointed out. Continuing to use an established tool that gives a lot of trouble isn't really the conservative way to manage a project, he noted.

However, Nelson was optimistic about the long run. Tools and techniques do exist for each of Thayer's problem areas. Of course, they are not yet generally used, but the leading software companies which do use them are getting good results. He cited TRW's adoption of structured programming techniques in 1972 as an example of what good management can accomplish. Although today that decision would be noncontroversial, there was a lot of in-house opposition at the time. But the vice-president went ahead anyway and demanded that it be done. The results, of course, have proved him to have been right.

Nelson concluded with a challenge to managers. "If you're going to be a software manager, you have the responsibility to understand the tools and techniques of your trade. You have to find out which tools are good and how to use them." ■

Acknowledgments

I am grateful to Wallace A. Depp, general chairman, C. V. Ramamoorthy, program chairman, and Jack Scanlon, member of the program committee, for their insights on the conference topics.

References


---

**JOB OPPORTUNITIES**

Sandia Laboratories is seeking MS and PhD level applied computer scientists and computer related electrical engineers to assist in implementing a major change in architecture to its central computing facilities. Specific areas of interest include:

- High Speed Computer Communications
- Operating System Design
- Computer Security
- Computer Systems Simulation
- File Access and Control

Preference will be given to those individuals with experience on CDC 6000 and 7000 series computers. Please send resume and salary history to:

John Wheeler
Division 3531
Sandia Laboratories
Box 5800
Albuquerque, N.M. 87185

Sandia Laboratories is located in the heart of the sunny Southwest and offers a liberal fringe benefit package containing 24 days annual vacation. Recreation opportunities in and around Albuquerque abound.

---

**196: Proceedings, COMPSAC 78**

November 13-16, 1978 (800 pp.)

These proceedings include over 150 papers on software generation methodology and applications presented at the IEEE Computer Society's Second International Computer Software and Applications Conference. Among the topics covered are software design, testing, reliability, and standards; data-base design; and security and privacy.

Non-members — $35.00
Members — $22.50

Use order form, p. 106.