How software affects the world. How the world affects software.

Rich Santalesa

TALIGENT READIES A NEW DEVELOPMENT PARADIGM

WHAT'S PINK AND BLUE AND READ BY anything? The new operating system and development environments from Taligent, the joint venture of Apple, IBM, and Hewlett-Packard.

Although this is the year Taligent will finally ship a suite of products, it's been a long and winding road. The Taligent technology began life as Apple's secret Pink operating system, originally based on an Opus 68K-specific microkernel. In March 1992, Apple spun Taligent out into a joint venture with IBM and the technology expanded, and its position in the Apple and IBM operating-system hierarchy became a bit blurred.

Once Taligent moved out from Apple's control, the underlying microkernel was switched to Mach 3.0, which is both more powerful and portable than the previous 68K Opus kernel and has the advantage of being the same kernel beneath IBM's Workplace operating system. Hewlett Packard joined Taligent as the third major partner in January 1994.

THE GOAL IS TO REVOLUTIONIZE APPLICATION DEVELOPMENT. THE VEHICLE IS THE C++ OBJECT.

This operating-system history is primarily important in understanding the troika of components that Taligent will offer in 1995 and their potential effect on developers and applications.

The Taligent product line is completely built around objects. From the development tools, to the operating system, to the user interface and application design — every feature stems from the extensive use of consistent object technology. As the scion of Apple, IBM, and HP, Taligent inherits access to an astounding range of object and development technology, especially IBM's System Object Model and Distributed System Object Model and HP's Distributed Object Management Facility. Taligent has already stated that its technology will be bundled with the operating systems of IBM, HP, and Apple.

THREE PRODUCTS. The three prongs of Taligent's attack, in order of release, are

- CommonPoint 1.0, formerly known as TalAE (Taligent Application Environment). CommonPoint is an object-oriented framework that rests on the Mach kernel. Now in beta testing at 100 sites, it's more than your average framework. It is scheduled to ship in the first half of this year.
- TalDE (Taligent Developer Environment) consists of an advanced incremental compiler and linker, object-oriented project database, hyperlink tools, debugger, and additional tools, such as versioning control and an interface builder. It is scheduled for release in the second half of 1995.
- TalOS (Taligent Object Services) is essentially a complete object-oriented operating system. It is scheduled to ship in 1996.

AMBITIOUS GOAL. Taligent's ambitious goal is to revolutionize portable application development and how users interact with their applications, their work, and their computers. Taligent wants to be the industry's standard development method. Regardless of whether Taligent is ultimately successful, its approach and technology point the way to both the future of software development and computing — a future in which overall operating system functionality is dramatically enhanced, the underlying operating-system core itself is smaller, and applications interact to offer a document-centered computing environment.

The first examples of Taligent-forged applications saw light at last fall's Comdex, where a handful of independent software vendors demonstrated several programs and prototypes. Among them were Virtus Corp., known for the highly acclaimed Virtus Walkthrough modeling program, which displayed Virtus Navigator, a 3D Internet tool. Nisus Software showed Info Bank, a sophisticated document-management system. Abacus Concepts showed a 3D data-visualization tool that revealed data associations and patterns using a customized database-access framework. Taligent found the Abacus approach so appealing that it will include it as part of the standard CommonPoint environment.

COMMONPOINT FOUNDATION. According to Joe Beyers, Taligent's director of product marketing and planning, CommonPoint contains 100 frameworks that offer functionality ranging from high-level application frameworks, to complete text and graphics editing, compound documents,

Editor:
Angela Burgess
IEEE Software
10662 Los Vaqueros Cir.
PO Box 3014
Los Alamitos, CA 90720
aburgess@computer.org

international text support, 2D and 3D graphics, and lower-level system services.

CommonPoint is fundamentally different from conventional frameworks because it will extend down to the operating system level via TalOS. Conventional frameworks, such as those in Microsoft and Borland C++ environments, barely scratch the surface of the underlying operating system. CommonPoint digs deeply into the Taligent operating system or runtime environment and provides a wealth of methods and classes. Within CommonPoint's 100 frameworks are 1,730 public object classes, as many nonpublic classes, and 53,000 methods. This compares with roughly 4,000 calls on the Macintosh application programming interface and the 1500-plus calls in Windows. Taligent's technology is so replete with objects that its been called "a whole OS of nothing but hooks."

PORTABILITY PROMISES. This richness is intended to translate into a high level of true reusability and operating system functionality that compliments applications by enforcing programming discipline and maintaining clear communication among applications and between the operating system and the applications.

Despite the hype surrounding OO programming, reusability and object theory, swift and clean portability is the holy grail of developer tools. Many aspire; few deliver. Although the CommonPoint beta runs only on AIX RS/6000 systems, the Taligent APIs within CommonPoint, TalDE and TalOS will eventually run across several CPU architectures.

With today's porting tools, the major disadvantage is that you must port to the lowest common denominator, which means that the special aspects of certain operating systems are either handled poorly or ignored completely because they are not offered in other operating systems. This effect has led, to example, to the debacle that is Microsoft's MS Word 6.0 for the Macintosh: Mac Word 6.0's interface is a virtual duplicate of the Windows version — all Mac conventions are flouted.

Taligent's porting ace in the hole is the fact that Taligent is designed to coexist with other operating systems via a runtime environment that provides access to both the legacy applications and the full Taligent system, interface, and application functionality. As a result, taking an application from one Taligent environment to another preserves all the functionality offered by the Taligent frameworks. Nothing can be left behind because there is no lowest common denominator.

However, running the full TalOS precludes, at least for now, other non-Taligent applications. In other words, the TalOS completely replaces the host's original operating system. It remains to be seen how popular, or widespread, this will be.

BETA DEVELOPERS REPORT A STEEP LEARNING CURVE, THEN FASTER AND CLEANER DEVELOPMENT.

And the runtime environment, which maintains the underlying operating-system functionality and applications has a memory overhead of approximately 4M bytes and a disk footprint of 20M bytes. This is definitely not something you'll run on a four-year old 80386-based system. The ultimate size, memory requirements and performance issues are as yet unclear.

TALKING TO DEVELOPERS. What is perfectly clear is that Taligent is fully a C++ system. And it has a stiff learning curve, even for experienced C++ programmers, according to Jerzy Lewak, president and chief executive officer of Nisus Software. Taligent itself acknowledges that developing your first Taligent application will take longer than with conventional tools due to the learning process, but that subsequent applications are rapidly executed. Several Taligent developers confirmed this scenario.

"Taligent's frameworks are all coordinated much better than others I've seen. They're designed to work together with the underlying kernel, in a fashion similar to the Mac's ROM Toolbox calls, but on a supremely more advanced level," added Lewak. "Nextstep is the closest thing to Taligent but it's already old and not nearly as advanced — despite the fact that until now it's been *the* fastest development platform, bar none. We have spoken with people who have used Nextstep and we considered it, but it's clear to us that CommonPoint is the next Nextstep, if you will."

Likewise, Virtus Corp's director of technologies, David Easter, said that Taligent's developers "really understand the core processes in developing applications. For instance, there's a standardized way to make object changes, and make objects communicate cleanly. Basing apps on CommonPoint results in programs that are more consistent internally, cleaner, and allows the framework to do significant grunt work in cooperation with the Taligent environment."

For example, the CommonPoint framework supports multilevel undo, implements a basic menu structure, and offers pervasive drag-and-drop and other global tools that work throughout the environment, not merely within one application. With CommonPoint and the Taligent kernel you can use a highlight tool that works the same way in any document or application, and indeed the entire Taligent system. Taligent calls this interface approach "People, Places, and Things."

The PPT interface is based on a docu-centric model, in which a user creates documents that perform tasks, instead of using an application as such. It incorporates the notion of "people" you can send items to and access information about and the concept of "places" in which you have certain rights and are allowed certain functionality. Taligent believes this makes for a much more natural working environment and consistently more natural application behavior.

DEVELOPMENT ISSUES. To create Nisus' Info Bank program, Lewak sent two engineers to Taligent for training and education. Despite the fact that both engineers already knew C++, they spent three months working with Taligent to learn the environment and frameworks.

"Once you learn CommonPoint and Taligent's system you will be an expert C++ programmer, whether you want to be or not. It took three months to develop Info Bank once my engineers were up to speed. So there's a big payback in development speed, even with the rough state of Taligent tools at this point, once you grasp the concepts and embrace the discipline."

Info Bank lets users store and categorize any type of information from any application. Every window has an Info Bank icon in the upper left corner — again highlighting Taligent's task-document approach. Info Bank functionality appears ubiquitously by design throughout the environment, extending itself beyond windows and onto the desktop. "Any alias can be dragged from the desktop onto Info Bank to add data, while at any point you can access Info Bank and drill down with our guided-information access method," Lewak said.

And Lewak seconds Taligent's claims that porting is easy. "The current development platform is an IBM RS/6000 running AIX, which is what we demoed at Comdex on, but the porting issues are very, very fast. It's basically a recompile to move over."

Beyond development and porting mechanics, Taligent's very nature could change the contour of the application landscape. "In a pure Taligent environment, you really wouldn't create a monolithic program, like Nisus Writer. Rather the same functionality would be spread across several smaller apps that work tightly together — an offshoot of the stringently defined framework and object methods."

This is a radically different model from today's ever larger, ever later, ever buggier monster-sized applications. It envisages a terrain populated by small applications working in concert. Theoretically, since the applications are smaller, the ability to debug and optimize is suddenly magnitudes easier.

The unanswered question is who, on a system running a mix-and-match menu of applications from different vendors, answers the help line. Nevertheless, it's clear that Taligent is sitting on, using, and refining what is ostensibly the world's best developed, comprehensive, object-oriented development and system environment. If the future scares you, ignore Taligent. If not being prepared for the future scares you even more, it would be wise to investigate what's taking place in Cupertino. ◆

COMPANION PIECE

The March *Computer* carries "Taligent's CommonPoint: The Promise of Objects," a companion piece that focuses on the object-oriented technology that underlies the Taligent offerings.

FOR MORE INFORMATION

Taligent provides a very rich source of information, documents, and resources on its Internet home page, http://www.taligent.com. Or write to Taligent, 10201 N. De Anza Blvd., Cupertino, CA 95014-2233; (408) 255-2525 or (800) 288-5545; fax (408) 777-5181.

White papers available from Taligent:

- *A Study of America's Top Corporate Innovators*, 1992
- *Lessons Learned from Early Adopters of Object Technology*, 1993
- *Driving Innovation with Technology: Intelligent Use of Objects*, 1993

Other reading on Taligent frameworks and object technology include:

- *Taligent's Guide to Designing Programs* (Addison-Wesley);
- *The C++ Programming Language, Second Edition* (B. Stroustrup, Addison-Wesley)
- *Developing Object-Oriented Software for the Macintosh* (Goldstein and Alger, Addison-Wesley)
- *Object-Oriented Design with Applications* (G. Booch, Benjamin/Cummings)
- *Object-Oriented Technology: A Manager's Guide* (Taylor, Addison-Wesley)

ISO PUBLISHES ADA 95 STANDARD

David Sims

PROPONENTS OF ADA HOPE THE approval of a new version will boost the language's popularity in the commercial marketplace. In February, Ada 95 (formerly called Ada 9*x*) became a published standard from the International Organization for Standardization. Publication of standards from American National Standards Institute and Federal Information Processing Standards is pending.

The Ada trade group has renamed itself the Ada Resource Association and is reaching out for a broader membership. Meanwhile, the Ada Joint Program Office is sponsoring grants to computer science departments for Ada projects, and GNAT (GNU Ada Translator), an Ada 9*x* front end for the Free Software Foundation's GCC compiler, is designed to let users try Ada for free.

All this activity aims to reintroduce Ada to skeptics who think of it as "that DoD language." Ada's advocates say the new version incorporates the safeguards and software-engineering discipline of Ada 83 while adding full support for object-oriented programming and better interfaces to other languages. But vendors and others trying to sell users on Ada also must overcome a more tangible obstacle: a history of inadequate compilers and a lack of tool support.

WILL INDUSTRY ACCEPT IT? Ada's roots reach back at least to 1974 when the Defense Department began a project to develop a common language that could handle real-time embedded systems in mission-critical environments. After several years of proposals, reviews, and refinements, Honeywell-Bull won the competition to design the final language, which was released as Ada 83. (The language is named for Ada Byron, Lord Byron's daughter, who is often considered the world's first programmer

in light of her work with Charles Babbage in the 19th century.)

DoD wanted to develop a single, reusable software system to serve as a standard throughout the armed forces. In addition to simplifying training, Ada also aimed to reduce costs with efficient debugging and encouragement for reuse. The DoD also wanted a language that industry would support: A bigger market means a greater choice of compilers and tools. But part of the language's difficulties stemmed from the DoD's failure to promote it to industry, and thus ensure there were enough compilers.

"I think there was this assumption that it would stand on its own, and that they really didn't have to do anything," said Tucker Taft, chief scientist at Intermetrics' development systems department. "We know now — and people knew then — there's more to selling something than just announcing it and dropping it on their heads."

In 1988, the DoD undertook a revision and in 1990 Intermetrics won the bid to develop Ada 9*x*. The review committee and distinguished reviewers wrangled over how much to change the language, but a few areas emerged as crucial, including full support for object-oriented programming and the ability to interface with other languages. "We had to recognize it's a multilingual world," Taft, Ada 95's chief designer, said. "You're never really talking about building a whole system at one time. You're generally building subsystems that have to integrate with other systems written in [other languages]. ... We spent a lot of energy on interfacing."

TROUBLE AT HOME. Resistance to Ada was not limited to the commercial marketplace; it also had troubles within the defense community. Ralph Crafts, who led the Ada Strategic Alliance (predecessor of the Ada Resource Association) from its inception in 1989 until his resignation in 1994, said the Ada community has received mixed messages from the DoD, which mandates Ada's use but ignores violations.

Defense contractors who want to use another language must secure a waiver from the DoD. DoD officials said they issue few waivers, but Crafts said this is misleading. "Nobody who wants to avoid using Ada even bothers going through the waiver process," he said. "There's just no enforcement of it at all."

The Ada Joint Program office said it is difficult to assess the mandate's enforcement because waiver-granting authority is decentralized among military departments, and because the mandate allows the use of other languages in legacy systems and in cases where using Ada would not be "cost-effective" — a potentially large loophole.

REASONS FOR RESISTANCE. Why have some people resisted Ada? In part, Crafts blames the military for not explaining Ada's superiority to contractors and project managers. But he also blames the "guru" ethos of the programmer community — in which creativity is highly valued and standardization resisted — an attitude he said is obsolete in a world where software reliability is often vital.

Programmers may have resisted Ada because of a suspicion that it grew up protected, unable to compete with other languages. That characterization is unfair, according to Dave McAllister, who manages Ada and C++ products at Silicon Graphics. "[Ada] could always stand on its own, but there was this perception, like a kid inside of a plastic bubble, who would never get exposed to all the things that would make him grow up to be very strong."

Add to this the natural resistance to the difficulty of learning a new language and the cost of migrating legacy systems. "People say it costs a lot to learn Ada," said Christine Anderson, project manager for the DoD's Ada 9*x* Project. "I really don't think that's true. Certainly if you're a software engineer, learning one language or another doesn't take that long. What really takes some time and energy and cost is to train people in software engineering. I think that's what they're really complaining about."

However, she believes that industry is where the military was 15 years ago: beginning to develop complex systems that will require maintenance over decades. "I think some of these companies are going to get burned on their software practices. You see a lot of releases of software products, release 10, 15, whatever, because they probably didn't apply the right discipline to their software-development process." Ada's rigid discipline, its advocates maintain, actually saves time on debugging and maintenance of complex systems.

A more important reason may be that there weren't enough mature tools to support Ada back in the mid 1980s when many programmers took a look at it. "The compilers simply weren't capable of doing the level of work [programmers] wanted done," said SGI's McAllister. His customers now must be convinced of Ada's usefulness, he said. He positions the language as one of many solutions capable of working within a system. "Our view of Ada 95 is, 'Hey guys, it's just a compiler.'"

GIVING IT AWAY. One way to get people to try something new is to give it away. That's the idea behind the GNU Ada Translator, developed at New York University with funds from the Ada Joint Program Office.

FOR MORE INFORMATION

- Ada Joint Program Office, Defense Information Systems Agency, Code TXC, 5600 Columbia Pike, Arlington, VA 22204-2199; (703) 681-2463.
- Ada Information Clearinghouse, PO Box 1866, Falls Church, VA 22041; (800) 232-4211 or (703) 681-2466; fax (703) 685-2869
- The final standards of Ada 83 and Ada 95, lists of validated compilers and commercial Ada endeavors, and results of a US Air Force comparison of Ada and C++ are available via anonymous ftp at sw-eng.falls-church.va.us; World Wide Web site http://sw-eng.falls-church.va.us/.
- Ada Resource Association, 4719 Reed Rd., Ste. 305, Columbus, OH 43220; (614) 442-9232; fax (614) 442-0055; 73313.2671@compuserve.com.
- SIGAda, Association for Computing Machinery, 1515 Broadway, New York, NY 10036; (212) 869-7440.
- For information on GNAT, send e-mail to gnat-request@cs.nyu.edu.
- The Public Ada Library is at wuarchive.wustl.edu in the languages/ada directory.

The compiler is an Ada 95 front end for the GCC compiler, which is distributed freely over the Internet. GNAT makes Ada 9*x* available to most 32-bit workstations. It has three main components: a front end written in Ada 83 that parses and analyzes Ada 95 text to generate a tree form that can be mapped to the C semantics in the backend; modifications to the backend to support some Ada semantics such as variant record types and exception handling; and run-time features like a tasking module to support Ada 95's real-time capabilities.

Like other GNU software, GNAT is covered by the GNU Public License, which means you can use it freely to experiment, but are obligated to make any modified source code freely available. However, you can use GNAT to compile your own Ada code without inheriting GNU's so-called "copyleft" restrictions.

Another part of the effort to promote Ada 95 is the ARA. ARA's executive director Bob Mathis said the group is trying to build its membership beyond big compiler vendors, such as Rational, Thomson (which bought Alsys and Telesoft), Tartan, Intermetrics, and others. Mathis said the new organization is reaching out to include consultants, trainers, and developers of Ada support tools. ARA plans a series of seminars to promote the language this spring and summer. "For most people, [Ada's] an unknown," he said. "We've got to get out there and market it."

Crafts agreed that Ada's biggest problem at this point is lack of visibility. He cited an awareness survey conducted by Response Analysis Corporation asking commercial software users what criteria in a language were most important to them. The top four responses were reliability, performance, ability to handle large applications, and standardization — areas where Ada backers say the language excels. However, not a single respondent said they were considering Ada.

Ada backers are aware of what they call this "ignorance," but remain optimistic. "I think, if you look at the history of programming usage, every five years or so there's a new dominant language," said Intermetrics' Taft. "And I think, clearly, the upcoming dominant language is C++. But I don't believe it's going to remain dominant. ... If you talk to C++ programmers, you find many of them have plenty of gripes and wouldn't mind trying something new." ◆

IN BRIEF

♦ **Trade war.** The United States imposed trade sanctions on China on February 4, after the two countries failed to reach agreement on halting copyright piracy, including illegal software copying and sales. The Business Software Alliance estimates that only 6 percent of software used in the People's Republic of China is legal. The US developers' share of the rest could be worth $322 million.

China has copyright laws, but US industry observers say it fails to enforce them. BSA wants "explicit enforcement initiatives," including parity of software with other copyrighted goods, enforcement of China's anti-piracy laws, as well as raids and audits. BSA spokeswoman Kim Willard said that several raids last year by Chinese officials were "P.R. efforts." China has said that the US demands go beyond the scope of intellectual property laws.

♦ **Software patents surge.** The US Patent Office issued 4,569 patents in 1994 — nearly one-third of the 15,000-plus software patents issued since 1970. Image-processing patents were most numerous (623), followed by network and communications (532), operating systems (448), process and numerical control (374), and graphics (337). Fifty-six patents were awarded for algorithms, 28 for parallel programming, and 26 for virtual-reality applications.

IBM led the field with 396 patents, followed by Hitachi with 189, and then, with 107 software patents each, DEC, Xerox and Fuji Xerox, and Toshiba.

— Internet Patent News Service

♦ **IP spoofing.** The Computer Emergency Response Team's Coordination Center at Carnegie Mellon University reported intruder attacks on systems that use IP addresses for security authentication. Intruders create packets with spoofed (hoaxed) source IP addresses to gain user or root access to a system. Once in, intruders can dynamically modify the Unix kernel, allowing them to hijack existing terminal and login connections.

One way to detect the modification is by commands appearing on a user's terminal or the system not responding to typed commands. Vulnerable configurations include routers to external networks that support multiple internal networks, routers with two interfaces that support subnetting on the internal network, and proxy firewalls where the proxy applications use the source IP address for authentication.

The best prevention, the advisory said, is to filter on the inbound side of your external interface. CERT recommends installing a filtering router that restricts input to the interface by disallowing packets with source addresses from inside the network.

You can access the full CERT advisory on IP spoofing via ftp at info.cert.org in the pub/cert_advisories directory, under the file name CA-95:01.IP.spoofing.attacks.and.hijacked.terminal.connections. It includes references to vendors that provide filter protection and to academic papers that detailed the weakness as long ago as 1989.

♦ **GIF royalties.** Unisys Corporation said it owns rights to an algorithm widely used in on-line graphics and will seek royalty payments from developers who use it. CompuServe introduced the graphic-interchange format in 1987, incorporating Lempel-Zev-Welch compression, which it believed was in the public domain. In 1993, Unisys learned the LZW algorithm was used in GIF, and the two companies reached an agreement in June 1994 under which CompuServe paid an undisclosed sum.

To recoup that cost, CompuServe in December imposed a one-time $1 fee plus a royalty of 1.5 percent (or 15 cents per program, whichever is more) on developers who use GIF. However, the deal has caused some concern for shareware and freeware developers. Their products are distributed widely via on-line services and they say they have no way of contacting all the users, many of whom never register the software. Yet developers may be liable for royalties to Unisys on every copy of their software in use.

— NewsBytes News Network

Interview with Tom DeMarco

BLUEPRINT FOR SUCCESS: INNOVATE AND INVEST IN PEOPLE

AS AN ENGINEERING CONSULtant and trainer, Tom DeMarco works with software companies to manage change. A keen observer and accomplished writer, DeMarco is in a perfect position to comment on what's wrong and what's right with the software industry. He spoke with Managing Editor Angela Burgess after giving a keynote address at the Applications of Software Measurement Conference in which he criticized the use of measurement to mechanize software production.

Q: *Is the software industry ready for mechanized measurement?*

A: I feel that we, the software industry, have lost track of what it is we do. On a global scale, in countries everywhere, we're trying to squeeze software development into a production niche, whereas software is much more appropriately a research-and-development activity.

We try to squeeze it into this niche by focusing on process, by focusing on repeatability. The whole Capability Maturity Model is totally focused on software as a production activity. And software is not properly a production activity ... if you find yourself doing the same thing over and over again, something is terribly wrong. Rather than learn to do it better, that's the thing we ought to learn not to do at all.

Q: *But what about a Microsoft, say, which is big enough to have a research-and-development division, but also puts out products that are virtually replications of what they produced before with a few enhancements and a few new features. On the replication side of their business, why isn't measurement to modify behavior appropriate?*

A: I don't want to be too much of an extremist, but anytime you find yourself treating software as a production activity, I suggest that ought to be a danger sign. Maybe when you're designing a product for a new platform you ought to go back and rethink your cross-platform architecture instead of simply replicating what you've done before.

This subject is very strongly tied to our concept of risk management. With a production mentality, you minimize risk. With a research-and-development mentality, you realize that anything that has low risk is your enemy. Only the thing that has high risk is really worth doing. The Denver International Airport is a very compelling example. People have taken potshots at that over the Internet; it's a constant source of low comedy. But the fact is, that was a very high-risk project that rushed into a domain where angels fear to tread — robotics. Maybe they didn't take account of their risks — you can criticize the project in some sense. But the point is that it is a project we ought to be celebrating, because those people took on a challenge, an enormous challenge, and gave it a good shot.

Q: *Do you think there's a payoff for them down the road?*

A: There will be an enormous payoff for someone. The domain of software-controlled robotics is a rich and promising one. But the owners of the airport aren't the ones who stand to gain. They are just another example of how ignoring risk can hurt you.

WE'RE TRYING TO SQUEEZE SOFTWARE INTO A PRODUCTION NICHE, WHEN IT'S REALLY MORE OF AN R&D ACTIVITY.

Q: *How do you see this production attitude affecting the kinds of people who are being attracted to and who are getting hired into the software industry?*

A: Well, I think there will be fewer, better jobs in software as time goes on. I think one of the things that caught us unaware during the 1980s was that the established software organizations were becoming white elephants. Organizations that once built software that had enormous benefit built up a large staff during their development heyday. Then they became creatures of software replication. And the incentive to build a software system ceased to come from the user and came instead from the software people.

Very often now people will ask me, "How do you manage a project when the user doesn't want the product?" Twenty years ago that question would have been a joke, but today I hear it a lot. "How do we deal with this customer who really doesn't want the system — has never wanted it?" I see a lot of these projects. They're all fiascoes. But how do they come about? Well, they come about because there's this white elephant to feed — an enormous establishment built up with all these people that have jobs.

Q: *So what happens? Who services this store of software that's already out there and that continues to grow?*

A: Well, I think we're going to be generating a lot less software. I think this whole idea of the software factory in which we learn to be tremendously effective at generating hundreds of millions of lines of software is part of the problem — the production mentality carried to its extreme.

Q: *So you think that the export business being set up in much of the Third World is for naught?*

A: In the short run it will take business away from companies stuck in the implementation mode. But in the long run there is not going to be that much replicable software work. In the long run we'll find ourselves going back to doing software as a development activity without any production mentality at all and doing high-risk projects. System invention will be more relevant.

Q: *How do you see R&D work evolving? What kinds of people are hired? What skills do they have? What's the management structure like? How do they work?*

A: Companies that do software without the production mentality are, in my experience, mostly engineering companies. They hire people who have reasonable engineering skills and invest heavily in them. At Hewlett-Packard, a manager in medical imaging told me they invest two-and-a-half years in an employee before he or she begin to carry their own weight.

Q: *Talk about high risk!*

A: Well, it's only high risk if you don't intend to build a culture that tends to keep people. In American industry, we don't capitalize any investment in people. When Marie leaves and George takes her place, we ignore the fact that Marie did complicated jobs successfully, whereas George is worse than no one because not only can't he do these jobs, but he uses up Harry and Fred's time to try to get up to speed. At the end of two-and-a-half years you've got hundreds of thousands of dollars tied up in getting George up to speed.

Now in American industry we expense that cost. I'm not suggesting we should do it differently, but I am suggesting we manage people as if they were a capital asset. If we don't do that, we won't manage people in a way that makes very effective use of that capital. One of the most compelling signs in the 1990s is all these layoffs, many times by companies that are making money. Take IBM. Its cash reserves amount to tens of billions of dollars and its earnings are positive, yet it laid off many thousands of people. Well, you can only do that if you pretend you have no investment in those people. If you wrote down the $300,000 you had invested in each one, the stock market would just go crazy.

But by not keeping track of the investment, not managing human capital as though it *were* capital, we manage it rather foolishly. Development organizations have much more investment in human capital and so they are much more aware of this.

Q: *What about day-to-day management, what is that like?*

A: I think the best managers you encounter in a development organization are those who are charmed by the idiosyncratic nature of people. And I think that's a characteristic of a natural manager.

I have worked with seven great managers in my life. And I look back at these people as, first of all, people who understood the importance of community. They understood that the human creature has this great need for community. A great need to be part of a 19th-Century town, where you know everybody. Only, those towns are gone. Today that need is filled by our companies, or at least the best ones fill that need.

A NATURAL MANAGER IS ONE WHO IS CHARMED BY THE IDIOSYNCRATIC NATURE OF PEOPLE.

Q: *What about this new form of development organization springing up in the multimedia world in which all the work is outsourced from a publisher to a software house? The software house does the work, then moves on to someplace else. This is a much more nomadic model than the one in which there is heavy emphasis on training and long-term employment.*

A: I don't think the studio model is a real trend. It's amusing but irrelevant, because it's not happening that much. I do see a trend away from general-purpose programmers. Because now software is very evolved and further fragmented. It's increasingly "nichified," if I may coin that term. Pockets of developers in different organizations are working in entirely different ways. But I don't see them becoming nomadic.

Q: *Not even multimedia?*

A: Well, multimedia is just the thing de jour, and of course when you have a new component that adds itself to a development process, you have a small number of people who have that skill. They float for a while and then find their place, their niche.

Q: *What are the things that last, that seem to have persisted and are strong and important as ever?*

A: Well, I go back to the concepts of treating people as human capital and building community. People have a firmware need to work for a community. One that accounts for people as a capital component.

Let me make a point about the sociological character of our business that strikes me as essential. It has to do with "coaching." Software developers have a near-religious sense about coaching, because it is through effective coaching that we achieve meaningful growth. But while coaching is wonderful, it is not a management role. The model of the master craftsman has ceased to be very applicable. The only reasonable coaching that goes on today is at the peer-to-peer level. Companies that think a lot about coaching are trying to make across-the-organization peer coaching work. Of course, it's hard to be coached by your peers, because it calls attention to the fact that they know something you don't know. It is thoughtful, caring management that makes this okay. That is the key to making this coaching happen. You have to feel safe to be coached by your peers. ◆