INCIDENTAL PAPER

Seminar on Intelligence, Command, and Control

I3I: Information, Information, Information, and Information
Timothy G. Hoechst

Guest Presentations, Spring 2000
Charles E. Allen, Albert J. Edmonds, John J. Garstka, Timothy G. Hoechst, Hans Mark, Dale W. Meyerrose, Mark C. Montgomery, Scott A. Snook

July 2001

Program on Information Resources Policy

Center for Information Policy Research

Harvard University

The Program on Information Resources Policy is jointly sponsored by Harvard University and the Center for Information Policy Research.

E-mail: pirp@deas.harvard.edu URL: http://www.pirp.harvard.edu
ISBN 1-879716-74-7 **I-01-1**

July 2001

PROGRAM ON INFORMATION RESOURCES POLICY

Harvard University **Center for Information Policy Research**

Affiliates

Anonymous Startup
AT&T Corp.
Australian Telecommunications Users Group
BellSouth Corp.
The Boeing Company
Booz•Allen & Hamilton, Inc.
Center for Excellence in Education
CIRCIT at RMIT (Australia)
Commission of the European Communities
Critical Path
CyraCom International
DACOM (Korea)
ETRI (Korea)
Fujitsu Research Institute (Japan)
Hanaro Telecom Corp. (Korea)
Hearst Newspapers
High Acre Systems, Inc.
Hitachi Research Institute (Japan)
IBM Corp.
Korea Telecom
Lee Enterprises, Inc.
Lexis–Nexis
John and Mary R. Markle Foundation
Microsoft Corp.
MITRE Corp.
Motorola, Inc.
National Security Research, Inc.
NEC Corp. (Japan)
NEST–Boston
Nippon Telegraph & Telephone Corp (Japan)
NMC/Northwestern University
PDS Consulting
PetaData Holdings, Inc.
Research Institute of Telecommunications and Economics (Japan)
Samara Associates
Sonexis
Strategy Assistance Services
United States Government:
- Department of Commerce
  - National Telecommunications and Information Administration
- Department of Defense
  - National Defense University
- Department of Health and Human Services
  - National Library of Medicine
- Department of the Treasury
  - Office of the Comptroller of the Currency
- Federal Communications Commission
- National Security Agency
- United States Postal Service

Upoc
Verizon

I3I: Information, Information, Information, and Information

Timothy G. Hoechst

April 6, 2000

Timothy G. Hoechst is vice president of technology for Oracle Service Industries, where he works with Oracle's customers in government, education, health care, financial services, utilities and telecommunications. He and his staff of senior technologists design and build information management solutions, resolve technical issues, and help clients to leverage information to improve their services. He is also one of Oracle's chief spokespersons on technical issues. Mr. Hoechst joined Oracle as a consultant in 1988, and built database systems for many of Oracle's largest federal customers. In 1990 he co-authored Guide to Oracle *as part of McGraw-Hill's Database Experts Series. He then moved from consulting to technical roles in marketing and in sales, He now represents and supports consulting, sales, marketing, and development. Mr. Hoechst was graduated from Harvard College, where he earned a B.S. in computer science.*

Oettinger: Because Tim is a Harvard graduate, this is a kind of a homecoming for him. Also, he's alone among all of the speakers this semester to bring us a flavor of the role of technology in all of this. As Charlie[1] indicated, there's a cornucopia of opportunity, and I hope that he will enlighten us a bit more on that. In fact, he can do what ever he damn well pleases.

Hoechst: I didn't tell him to say that! As you know, I work at Oracle, a big software company. My understanding is that today we're the sixth largest company on the planet. Tomorrow it may be totally different.

Student: Probably broken up by the federal government.

Hoechst: When Tony invited me to speak here, I asked him what he thought I should talk about, and he said, "Whatever you like. The context of the course is talking a lot about command and control and intelligence, obviously." I don't know much about those things, except that I came to Oracle right from Harvard about twelve years ago and I've spent a lot of time helping service-oriented customers in federal and local government to implement systems with our technology. I bounce up against this a lot, but I'm clearly not from that world. That's why I enjoyed Charlie's talk so much, because I heard his perspective on this. My perspective on these sorts of issues is a little different.

When we think about command and control, or the whole operation of warfare or of running government, I give it a different vernacular, which is that this is all about information. As

---

[1]Charles Allen sat in on Mr. Hoechst's presentation, which immediately followed his own.

far as I'm concerned, there are no relevant computer problems that aren't related to the storage, retrieval, analysis, and dissemination of information in some form or another. The process of communication with a distributed group of people who need to perform a task or to reach some goal, whether that goal is tactical or strategic in the defense world, is all about the gathering, sharing, storage, analysis, and dissemination of information.

I'm going to focus my remarks on how the world is changing in the way it stores and retrieves and manages and analyzes and distributes information. I hope that throughout the discussion we'll dive a little bit into the context of what you're studying for our examples, but I'm going to go from the very high, philosophical end of this subject to the very specific. We'll come back and forth a little bit. So, feel free to direct me in any way you like.

For those of you who don't know, Oracle's primary product is a database. It's really only a small percentage of what we do, but everything revolves around that. Mostly we store stuff. We talk a lot in my business about the storage of lots of data, and about vast databases and so on, but none of that's really relevant. When we talk about information, it's really all about access. What's relevant is that you can get just the information that someone needs to the right person in a fast, simple, cheap, secure, and reliable way.

If you could show up tomorrow and say, "You know what? All the people in the organization have at their fingertips exactly what they need. It's fast. Trust me, it's secure. You won't see anything you're not allowed to see. You won't see anything you don't need to see, and it didn't cost us much to do this," you'd be the next president. That would be a wonderful goal for us to reach.

If you could do this, no one would care how you did it. They wouldn't care what sort of architecture you chose, or whether you bought big mainframes and put them in the middle of Missouri, or you put a personal computer [PC] interface on your desk. They wouldn't care whether you used wireless. They wouldn't care what encryption you used. Keep this in the back of your head when we talk about the goal. Whether you are at the Pentagon looking at an entire globe of conflict, or whether you are in a foxhole looking at your immediate conflict, it's all about having access to the information you need.

Everybody's going to have a different perspective on the information. The policymakers have a very different view of the same information from that of someone in the field. There are some tanks on the other side of this hill. Somebody sees that as a big picture in terms of the economics of how they were bought and who sold them and so on. Someone else sees them as an immediate threat. But it's the same information.

How do we create an environment where all of this information is gathered, it's all available, and we meet these goals? I'm going to talk a little bit about why we think technology is making this widely possible. I'm not going to suggest for a moment that this has not happened yet, but we're moving this way.

You've heard of this thing called the internet. This is a popular technology today. It is very widely used by a lot of people. We have a lot of customers who talk about starting to understand and adopt this bit of technology, and they view it as the current thing. We had the prior thing, and now we have this current thing in technology. I'd like to suggest, though (and maybe you'll permit me to wax philosophic for a minute), that the role the internet is playing in our lives is

much broader than just a current technology evolution. So let me actually take a historical perspective on this.

Gutenberg has been thrown around a lot as a milestone recently. We tend to look at computing since the 1960s. We say, “Gosh! We had big computers then. They filled rooms and they were really expensive and now that power is on my desktop. Isn’t it funny how far we’ve come since 1984 when the PC was introduced?” I think that narrow view of our perspective on the gathering and dissemination and use of information is naive.

If we look at Western civilization over the last 2000 years, it accomplished very little from about 500 to 1400. There were really only two innovations of note: the windmill and the waterwheel. Mostly during that time we were spending our money on the church. We were building monuments to God. The church leaders, the wealthy, and the elite were the only ones who knew anything because they were the ones who could afford the small number of books. Then the plague came and killed between a third to a half of the population—poor people and rich people, lay and clergy, equally—and people thought, “Maybe we’re not spending our money wisely here.” So they started to invest in art and science and more secular things.

What’s interesting is how much changed during the fifteenth century in the way we look at our world. The Renaissance began by looking at the world with perspective. We started to look at art in a new way. All of a sudden, we also looked at the heavens. We looked at the human body. We started to discover a lot more about how medicine works. We started studying things that had otherwise simply not been looked at for literally a thousand years.

Johann Gutenberg, in Mainz, Germany, invented the printing press around 1450. He jury-rigged a wine press and created the first movable type so he could reproduce multiple copies of a book. At the time, it was apparently not perceived to be all that relevant, because so few people could read. Why invent something for a market that doesn’t exist? The reality is that it changed everything. People now could learn to read because they could afford books. It’s a very simple idea.

What’s interesting is that this really marked the beginning of a new pursuit of innovation; that is, seeking more knowledge about the way our world works and sharing that knowledge with each other so that we can further the earlier innovation. If you look at the pace of invention and innovation since then, which is really only half of that idle time we’d had over the prior millennium, it is extraordinary that it doubles every ten years. This is magnified even more by our study of biology. We’re very close to finishing the mapping of the human genome. In ten years we’re going to look back on our current understanding of medicine and laugh, whereas forty years ago the concept didn’t even occur to us. That sort of innovation has been built on the shoulders of the innovation before.

It all started with Gutenberg. *Life* magazine published a millennium retrospective that talked about the most significant things that happened in the last thousand years and marked his invention as Number One. They called it the beginning of the Information Age, because, for the first time, information and knowledge became more widely available.

If you look at it in these broader terms, the internet is not a cool technology innovation based on a standard network protocol called IP [Internet Protocol] that allows us to share files. The internet is really the culmination of this philosophy. Within the next decade, there’s not going to be a person on the planet who doesn’t have access to the sum total of human knowledge, and

that is the culmination of what began just a few hundred years ago. The point is that this is a fundamental change in the way we're going to interact with one another as humans and in the way we do business.

I'm going to talk a little bit about how it has changed the way we build information systems, because we've seen that it has given us a new perspective on the way we store, share, and disseminate information. Keep this thought in the back of your head as we talk more tactically about how it has changed the day-to-day aspects of managing information. I just want to make sure everybody is clear on our perspective, which is that the internet is not the current architecture for sharing information: it is the last one.

What makes it unique in that sense is that we've had various architectures for the storage and sharing of information (I'll talk about them in a second), and this is the first one that's network centric. We have a long history of large networks, including the telephone, water, and electricity networks, and some of them have worked and some of them haven't. My favorite example of one that has worked well is the water network—the way we've solved the problem of storing and distributing water to the people who need to consume it. It is fairly simple. We gather it in big holes in the ground and we add stuff to it and we take stuff out of it and we filter it through ever-narrowing channels to a fairly simple user interface device. Sometimes, in the airport, you just put your hands there and water comes out.

This is a huge and complicated network. But where's the complexity? It's not in the device, it's in the network. This complex network is centralized. When the surgeon general says we've got to add fluoride to the water, we don't upgrade all our faucets and add fluoridators to them. We put fluoride in the network, and then it's easier to change and distribute. It's the same whether we disseminate a telephone signal or an electrical signal. The complexity of these large distributed networks is in the network, not in the access device. The internet has taught us a lesson about computer networks that we didn't know before.

I don't know if you've seen a funny PBS show called *The Triumph of the Nerds*. If it comes around again you should watch it, although it's really long—about four hours. It tells the story of the beginning of the PC. They interview Steve Jobs and Bill Gates and Steve Wozniak—a lot of these fellows—and they talk about how they invented the PC. I remember a couple of interesting things from it. First, Gates said, "Our goal was to put a mainframe on everybody's desk." They've done that. The computer I'm using here is more powerful than the mainframes they were replacing in the early 1980s. Windows 95 has more lines of code than MVS [multiple virtual storage; a mainframe operating system]. Microsoft has accomplished that goal.

What's more important is that Steve Jobs said, "I went to PARC—Xerox's Palo Alto Research Center—and I was looking for technologies. I saw three things. I saw the graphical user interface; I saw Ethernet; and I saw objects. Before, none of these had been seen outside the lab. My brain wasn't big enough to hold all three, so I took the graphical user interface and ran, and I created the Macintosh. I left Ethernet—networking—behind."

The PC, which was born out of a desire to put computing power in my hands, didn't consider networks. We thought of them later, when we needed to share printers. We wanted two PCs and one printer, so we had to wire them up. Then we said, "Maybe we could share a disk," and suddenly we had a file system. Everybody had a well, and later we thought, "Maybe we could

have a big centralized reservoir and distribute water." Networks grew up as an afterthought in computing.

The internet has shown us the power of taking the complexity off the desktop and putting it back into the network. This is important, because when we talk about the way we now move, or store, or disseminate, or secure information, the fundamental difference is huge. When you go to Amazon.com and buy a book, it's a huge, really complicated information system. You query their library books. They authorize your credit card. They have a profile on you. They say, "There's a new Martha Grimes book you might like, because you bought one last time," and all this sort of stuff. Where's that complexity? At Amazon. It's not on your desktop. That's the way the World Wide Web works. What we've done is finally started to wake up to the idea that we should not only allow the internet and the simple examples like Amazon and Yahoo, but we should use this architecture when we build our own information infrastructures. That is what's beginning to happen.

Let me talk a little about that fundamental architecture, before we talk about the way it applies in specific examples inside implementations like command and control. Pardon me if I get too simple, but what I want to do is make sure that everybody understands why this architecture is fundamental. All the people who have ever clicked on a Web browser think they get it, but when we talk about how it relates to building systems only a small number of those people actually do.

Every information system really has three components. First, it's got a user interface. This is what interacts with you and determines what color the screen is, what happens when you click the mouse, and what the menu looks like. Second, it's got a data layer. This is for storing and retrieving of bits on disk, and backing data up and securing it and so on. But then its got the really important part in the middle. It is the soul of an information system. It is what decides what that information system does.

Suppose I need to order a part. Before I order it I'd better check if I have it in inventory. If I have it in inventory, I want to check the price times the discount and I want to figure out the cost. All that work that an information system does we call the application, or logic, layer of an information system.

It used to be, in the early days, that all of this was a single program running in a centralized way, typically on a mainframe, typically written in COBOL [Common Business Oriented Language] (**Figure 1**). One program interacted with the user, manipulated the data, and then read and wrote it to disk. This model worked beautifully. It is a very robust model for building information systems. The reason is that this big computer is managed by professionals. It's protected behind a really strong piece of glass. We have power, we have generators, we know how to back it up, and all that sort of stuff. We have one copy of this application. We know exactly how it works. As long as the guy who wrote it doesn't leave, we're okay. We know how to maintain it, and so on. These systems work so well that they are still running after thirty or forty years. That's why we had the Y2K problem: because the developers who wrote these took a shortcut when they were writing stuff to disk, but who can blame them? Who'd have thought these programs would be around for thirty years?

There are some downsides to this model, however. These computers are very expensive, and they didn't get faster and cheaper quickly enough, especially compared with the

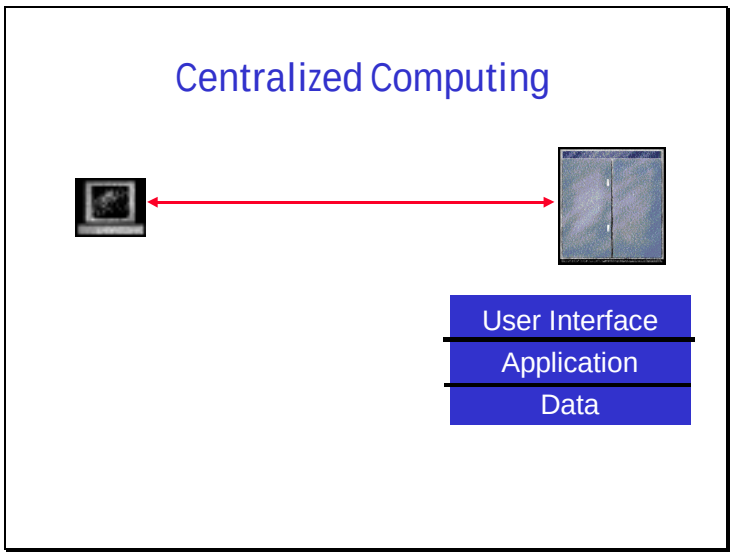

Figure 1

minicomputers and personal computers that came along. So we needed a new model, and our buddies Gates and Wozniak came out with personal computing (**Figure 2**). They said, "We're going to free you from the bonds of tyranny that IBM has imposed upon you with the mainframe and allow you to do all of your computing on the desktop." Apple's 1984 ad actually said that, not in so many words: "Now you can do all of your computing here locally. You can have your application, you can have your graphical user interface, you can store your data and manipulate it, and so on."

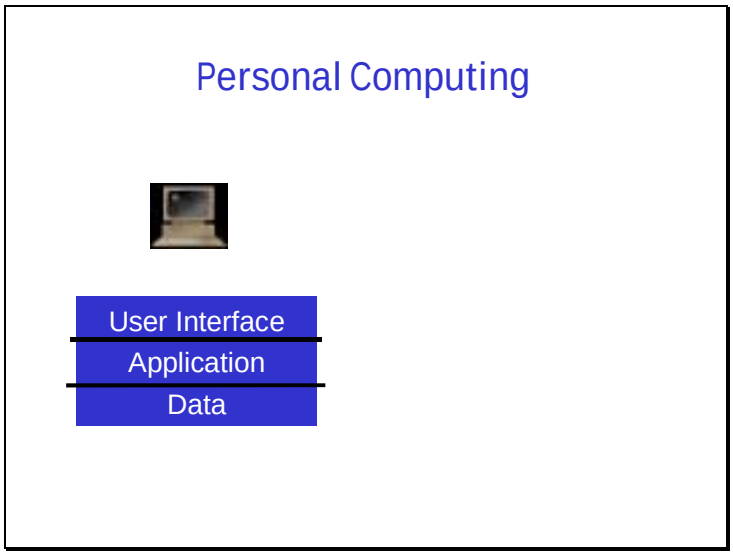

Figure 2

Personal computing was actually a very short-lived period in information systems, because everybody quickly realized it is only relevant if we can share the data. It's only good if you create it, I modify it, and then you analyze it. We quickly moved to a new model that we call client/

server computing (**Figure 3**), which we lived by for a decade and a half or two decades. What it said was, “We’re going to take advantage of the PC, and have lots of sophisticated software there at your desktop for accessing and manipulating the data, but we’re going to put the data up on those big computers where we can share it.”

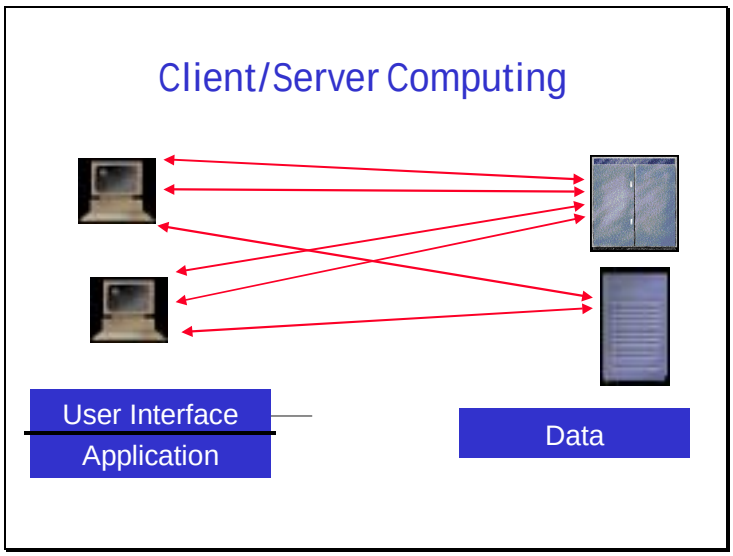

Figure 3

This is where companies like Oracle grew up: by writing software for storing and sharing data. Then we also started getting into the business of building tools so that you could write sophisticated applications for anything you wanted, whether it was a payroll system or a command and control system. It didn’t matter. You had the tools to build it here and the database to do it. What did we connect them with? A network, typically, a private network. This is the genesis of all networking, before the mainstream use of the internet.

We loved client/server computing, but it doesn’t work. It works in a lab with no dust, but it doesn’t really work in the real world on a large scale, and the reason is pretty simple. Remember where I said the complexity is in a system? It’s in the network. That is where people spend all of their money writing software. That is where people spend all their money maintaining software. That is where people spend all their money updating software.

What did we do with client/server? We took the complexity and we spread it over all of the computers throughout the enterprise: computers on ships, computers in tents in the desert, computers in the Pentagon, computers at our partners’ locations, computers all over the place. We put computing power on your desk, but we didn’t put an administrator there to manage this complexity. What’s happened in the real world is that people have discovered it’s too expensive to do computing this way, because I can’t manage this sort of environment for application development and deployment.

What we discovered with our military customers is that, because it’s so expensive to deploy software in their environment, we really have to make sure we get it right the first time. This is why I spend two or three years thinking about and planning and building these huge systems that I’m going to put out there, throw the switch, hold my breath, and hope they work. If they don’t

work, everybody gets fired; and if they do work, then they will probably only work for a month or two, until our requirements change. The sort of huge, monolithic development methodology for information systems that grew up around this simple need—that we'd better get it right the first time—is what has really paralyzed a lot of our information systems in large organizations like the government.

What did we do? Along came the internet (**Figure 4**) and taught us a new model that says, "Let's put that complexity in the network. Let's not put it on the desktop. Let's put it at Amazon." That's a great example. What software do you need to install on your computer to go to Amazon and buy a book? None. You have a Web browser, right? If Amazon had to install software on your machine, they wouldn't exist, because I use a Mac, you use Windows 95, he uses Windows 98, she uses Windows NT, and someone else uses Linux. You have *this* board, *this* driver, *this* thing, *that* screen saver. It's too complex. There's no way they could write one piece of software that will allow you to deploy it inside an organization, let alone to the public, as Amazon has done. It just won't work.

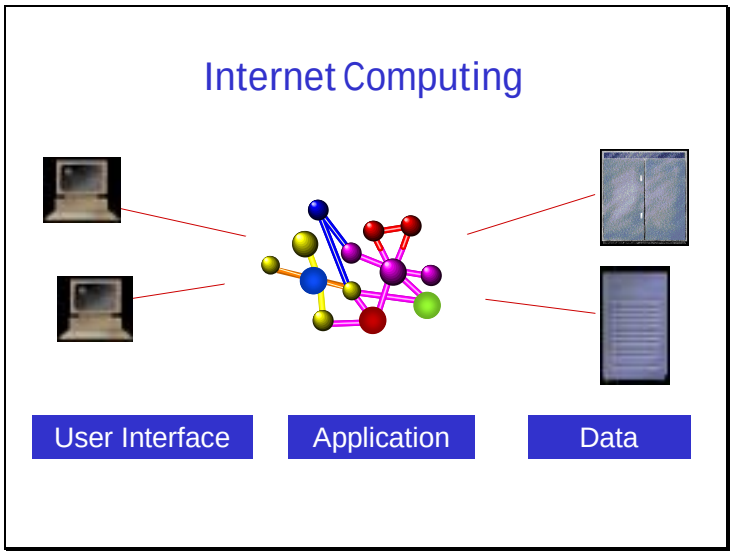

Figure 4

Student: Doesn't Java give you a way you can do this?

Hoechst: Java means two things. Java is a language and Java is a platform. As a language, it's just a language, like C. Java is hip. It's cool. You can easily buy books or hire people who know it. As a platform, which means a program running on my computer, it does start to do this because the software is deployed here. Who cares where it executes? But where do I maintain it? On the net.

When you buy Datek, you fire it up, pull down your Java stock-quote ticker, and you're watching it. Great. Tomorrow it's better, it's faster, or it's different. They changed colors. Did they tell you or ask you what version of Datek you were running in order to do that? No. It's still in Java. Do you have the Java platform? Great. That's all you need. As a platform, Java begins to do this by allowing them to own the complexity and allowing you to come by and execute.

Let me tell a personal story about Oracle and how we learned this lesson. We invented client/server. A company called Sybase, a competitor of ours, coined the term, but we were doing it before they even existed. I'd like to make sure everybody knows that.

At Oracle we have 40,000-plus employees around the world, in about 190 countries. Most are here in the United States. We have all sorts of information systems, just like everybody else. We have a payroll system, a human resources [HR] database, and a financial system with a ledger accounts receivable, all those things. We have a bug database. We keep track of all the bugs in our products. We have a support system. Every time somebody calls support, we record it. We have a corporate repository, where we keep collateral and presentations that we want to share. It's in our knowledge management base. We have, we think, about 2,800 Web sites inside the company.

We had the idea that we were going to give our employees the ability to access our information sources. How do you do it? We had what we called "sales force automation." We gave them all a notebook computer. For one summer, almost three months, we actually bought all of the Dell Latitudes that Dell could manufacture; their entire output went to us. You could pick up any two notebooks and their serial numbers were just a few digits apart. We were buying the latest and greatest hardware. We installed the application for our HR system, for our payroll system, and for our corporate repository. We had something we called a standard configuration, which every enterprise attempts to do with the Net. It's a myth. For us at the time it was Windows 95 and Microsoft Office. We shrink-wrapped it all into one box. We wanted to see the thing called "The Oracle Base Image."

We ran through the whole exercise. We handed everybody a notebook and we said, "Here's a $10,000 machine and you can use it, but there's only one rule. Don't touch it. Don't mess with it. Don't install anything on it. Don't take anything off it. There's one little area called 'C:Mydata.' You can put whatever you want in there, but don't touch anything else around it."

People took them home. If they were technical people, the first thing they did was format them. They put a key on them, or OS2. They had all sorts of different things. Even our nontechnical people installed a screen saver, or their kids put a little book in the compact disk drive and then put on some driver. Overnight every one of them was different. It was designed to be this way. That's the "P" in "PC." It's supposed to be unique to each user.

What did I do? I literally put it into my desk drawer and said I wasn't going to use it, because I'm a Mac user. I didn't want anything to do with it.

Monday rolled around and people started calling the help desk. "I can't get into HR. I can't get into the corporate repository." "Well, what did you do?" "I don't know. It doesn't work."

All of a sudden, this became too complex an environment for us to have distributed applications. How did we solve it? We threw money at the problem, so we had rooms full of people at every building and their job was simple. It was just to get your machine back to the base state. In Bethesda, where I worked, we had a room—I swear it had a "Mr. Ed" door where just the top opens—and there were half a dozen people in there. You'd walk up sheepishly, and ask, "Can you please try and save anything that ends in .ppt or .doc?" They said, "Well, we'll try. You made your bed, and you have to sleep in it, but we'll try." Do you know what a room full of people like that costs? The costs are huge. Every organization, especially the government, is spending an extraordinary amount of money maintaining this environment.

We had everything going for us. We had all the money we wanted for new hardware. We had a very homogeneous network. We had IP to just about every desk. We had a rather technical user community, and we had all the free Oracle software we wanted, yet we couldn't make this work.

It doesn't work in large enterprises, except in particular circumstances. If it's April 16, the Internal Revenue Service has rooms the size of football fields full of people, three shifts a day, eight hours a shift, pounding in tax returns. That's all they do. Those people are not even looking at the screen. They're not surfing the Web. They're not checking their 401(k) benefits. They're not logging into HR. They're not downloading PowerPoint. In that environment, this works well. A single-purpose machine doesn't need to do more than one thing. You have a nice big, fast database. These are very fast applications.

Who works that way? Very few people. Every enterprise has them. In our enterprise our HR clerks do that, and our corporate support people do that, but the rest of us—those who needed to get access to all of it—couldn't get it. Client/server doesn't work for access, and we said it's all about access. This new model says, "I've got some information, and more important, I have a way of accessing it. Come on by."

The internet taught us two lessons, and maybe if you only remember two things I say, these should be the ones. First, this idea of moving the complexity off the desktop and into the network (we call it internet computing or network computing), has two fundamental benefits. The first one—and this is one that everybody talks about—is that it's cheaper. It costs less to build, deploy, maintain, and buy computers that can do this. I don't care what computers you have. They probably run a Web browser. Great. That's the entire price of entry. It doesn't matter what you use. Last year's computers work just fine. It costs less, because we've centralized to manage the complexity. That's the hip and mainstream and cool thing about internet computing that people talk about.

More important, though, is the more subtle thing that it does. Do you guys have more than one information system and wish they could talk to each other? Probably. I don't know of an organization that doesn't, including Oracle.

Allen: When Bob Gates became director of central intelligence in 1992, he found he had four directorates within the same agency that did not communicate.

Hoechst: Let me tell a very basic story. Maybe I should be embarrassed to tell this story for my company's sake, but I'll tell it anyway.

Last summer we had an executive committee meeting, which is the senior management of the company, the top half dozen or so people. My boss is on this, so he was there and he tells this story. We have a fairly aggressive and flamboyant chief executive officer, a fellow named Larry Ellison, who founded the company. He's a rich guy who flies a MiG and things like that, but he's really smart. They were making some decision about compensation, and he turned to the chief financial officer [CFO], Jeff Henley, and asked, "How many people work at Oracle?" Jeff said, "I don't know; about 42,000." "How many exactly?" "I don't know." "Can you find out?" "Well, I guess I could." "How long would it take?" "Three weeks." Larry, as you can imagine, blew a head gasket and said, "You're telling me that I am the president of the largest information management company on the planet, and I can't tell how many paychecks I send out every week?

I don't know how many people work here?" Jeff had the unfortunate job of saying, "Yes, that's exactly what I'm telling you."

The reason is not that we didn't have enough of the latest and greatest technology, it's that we have, in 190 countries, 190 HR systems. A lot of these countries were using Excel as their HR system, some were using our big, high-end HR software, and others were using everything in between, and they don't talk to each other. The reason they don't talk to each other is not really a technology issue. It's related to technology, but, very simply, the issue is that the systems weren't built with each other in mind. You can give me any two systems and I can cram the data from one into the other, but if you want them to work in concert, you have to plan ahead. Every customer I have says, "I've got more than one system. I wish they could answer a common question. It's neither system's fault. *This* system was built to solve a problem and it solved it. *That* system was built to solve a different problem and it solved it. Not until we had both did we say, 'Oh gosh, wouldn't it be cool if we had some common answer?'"

What we used to try to do in this world was hook the databases together. Yesterday I sat in a meeting with the Department of Defense, literally. There was a representative from each branch of the Department of Defense, and they were going to have a discussion about a global data model. "You've got tanks. I've got tanks. Let's make sure we call them all by the same thing." "What do you call it?" "Well, we call it a "T-No." It's got a pound sign (#) at the beginning and we give it a number. What do you call it?" "Well, we start with a dash (-) and we have different names for the same things …." Try the politics of getting that resolved!

Assume, however, you could just come up with this global single definition of how everything works. The people still won't allow this to work. I talked with the chief information officer of the state of California, and he said, "We want people to renew their driver's licenses on-line. It's service to our citizens. You come in, put in some information, give us $15, and we'll renew your driver's license. But while we've got you, we're going to check and make sure your child support payments are up to date. We're not going to renew your driver's license if you have outstanding child support payments." It's probably a pretty good idea.

The problem is, here's the Department of Motor Vehicles [DMV], and here's Social Services. The DMV calls up Social Services and says, "If I give you a Social Security number, can you tell me if somebody has outstanding child support payments?" "Sure, it's what I do for a living. No problem." "So, what database do you use?" "We use Oracle." "Well, great, we use Oracle. Do you mind if we just go into your database and muddle around with your data?" It's not going to happen. Even assuming we could get around the security and all this sort of stuff, it's a social problem. There's no way all of these databases are going to come together. In many cases, especially in the intelligence world, we don't even want a wire to exist between them, for whatever reason.

Allen: Policy and security.

Oettinger: It could be something much more benign. I want to keep certain benefits detached, by law or by the decree of the body politic, from certain other benefits. My driver's license and my child support are two different worlds.

Hoechst: Let's say Social Services has the big picture, and DMV should not be interested in the big picture. But all I'm asking for is a Boolean true or false: Is this guy a deadbeat dad? That may be public. Social Services might even publish it on their Web site, yet I can't hook the two

databases together. It simply doesn't work. We spent a decade building technology that allowed these things to talk to each other and nobody used it, because the social problems overwhelm it. Nobody wants to do this.

This is lesson number two. The more fundamental thing is that the middle, or application, tier is where I control access to my data. That is where I decide how it is used, who uses it, and what they see it from. When that is in the network, I can connect the other tiers to one another. You can do a demo of this for yourself.

AltaVista is a search engine on the Web. You've probably all been there. If you go to AltaVista, you can look up people; it's the White Pages of the United States. If you put in "Tim Hoechst," in Virginia, you'll get two. I don't know the other one. Right next to my name is a little flower. When you click on it, *boom*, you're at 1-800-FLOWERS with an order form all filled out ready to send me flowers. (Yes, that's funny.) People say, "Big deal! I do that every day. It's the Web. *Click*, *click*, *click*, *click*. I jump around. That's the way it works." This is a huge deal!

You really went to a huge database at AltaVista. You queried the White Pages—the population of the United States that has published phone numbers. You went to one information system, and you used that information to populate an order to another company's information system. We've been trying to do this for decades, and here they do it with a little HTTP [Hyper Text Transfer Protocol]. It's possible that 1-800-FLOWERS doesn't even know AltaVista is doing this. All they did was link to it. They didn't say "What database are you using? Let's ship some data over to you." They took advantage of the fact that there's a standard way of accessing these information systems and that these standard interfaces to information systems can ask each other questions. "I don't care, 1-800-FLOWERS, what your inventory is. I don't care how many employees you have. All I care about is this one little function that allows somebody to order flowers, and I'd like to hand you an address to do it."

Once I build this sort of infrastructure, for the first time we have the opportunity not only to save money, but also to integrate information systems. This is the promise of incremental advances in the large information systems that exist inside our government. Today we have huge monolithic systems with huge policies and procedures around them. The functionality is locked in applications written by integrators who are no longer in business, instead of having an open environment where these can communicate with one another.

Student: You mentioned the state of the way we access. Right now, in the real world, we've got two Web browsers out there, Explorer and Netscape. How do we keep that standard way standard and at the same time allow it to improve and upgrade?

Hoechst: That's a good question. There are two things. One is that from the middle (application) tier down, everything is HTML [Hyper Text Markup Language], or, in a more complex way, Java. HTML is pretty standard. You can go from one browser to another and things pretty much work. Maybe they draw tables a little differently or whatever. Java is more complicated. A program that works on one browser might not work on another, because what it's trying to do is more sophisticated. As a platform, Java is still immature, but that will improve. I'm not really worried about that, because the advances will come: HTML, DHTML [Dynamic HTML], XML [eXtensible ML], all different sorts of ways of rendering the information.

What's important are the standards that happen in the network for interapplication communication. In my simple AltaVista example, HTTP is the standard. Basically, all that

happens is that AltaVista sends a request to a Web site that's already set up to do this. There are more sophisticated standards, such as distributed object standards based on things like CORBA [Common Object Request Broker Architecture] and COM [Microsoft's Component Object Model].

Student: All of this will work only as long as we can keep standards for communicating.

Hoechst: Why does the internet work?

Student: Because they all use the TCP/IP [Transmission Control Protocol/Internet Protocol] standards.

Hoechst: Everybody agreed. They said, "We're just going to agree to do it this way." They have an incredibly small number of standards. You don't need huge tomes of standards. You need a couple that everybody agrees on, and everybody plays on that field. Right now, the standard for networks is IP. The standards for Web dissemination to the browser are HTML, or DHTML, or SSTP [Simple Symmetric Transfer Protocol] for secure HTML. The standards between the different application (or logic) tiers are evolving. Maybe it's CORBA, maybe it's something simple like HTTP. XML is a popular one now for the logic tiers to share information with one another and so on.

What's important is that companies such as Oracle are just trying to make sure we've got them all. We're really promiscuous, needless to say, because we have the money to do that. We'll adopt them all and see which ones fail, and then we'll help drive the others. What's important is that people actually build their systems this way, because then they have the flexibility for tomorrow and the next day when the standards do change or evolve and so on. The way we build systems now is different from the way we used to do it. We don't say, "We're going to spend two months identifying the problem, two months identifying the team, two months designing the product, six months implementing it, four months testing it, and then turn it on." We say, "On Monday I have an idea, on Tuesday I build the system, on Wednesday I use it, and on Thursday I throw it away, or I decide maybe I'll want to use this again next week and I'll make it better." Systems now are organisms that we feed and fertilize and prune. They're not big monolithic things, and this architecture is what makes that possible.

Student: How does the architecture make it possible to develop systems that way? Can you expand on this?

Hoechst: It works because I didn't invest in the deployment of technology to you. It's only deployed in one spot, so I can change it very rapidly, on the fly, in a centralized way, and you inherit those deployments. Again, there are two things. One is in the deployment and one is in the intersystem communication. Because I have only one copy of the software, and because I can make minor changes to that copy at very little cost, I can improve it today, I can break it tomorrow, I can fix it the next day, I can always be tweaking it and making it better and advancing it and connecting it to other things. Once I go through the commitment of putting some software on your machine, I have to live with that for a long time. We have customers who are using eight- or ten-year-old versions of our software. The people who wrote the software are dead, but we still have to answer the customers' questions. It's very expensive for us, because we actually put something on their computers.

If you wanted to start your own on-line bookstore, you couldn't buy Amazon software. They don't have a piece of software that's on a compact disc that they installed. They have a thing that they've been feeding and nurturing and adding to and taking away from. That's why every day it's a little different: a little better or a little faster or a little slower. Sometimes you feed it something that makes it sick and then you try to heal it the next day. I'm going to tell a story about that in a second.

Student: Could you give us an example of a deployed application?

Hoechst: There are two things. There's Windows 2000 and there's Office 2000. Windows 2000 is an operating system that allows a computer to take advantage of its processor and memory and so on and run a program called a Web browser. Every computer out here needs an operating system. You can argue whether that operating system is good or bad or expensive or whatever, which I love to argue about. The important thing is that the access to the information system happens to be your Web browser.

Office 2000, however, is a piece of application software that's deployed to the desktop. Microsoft can support that because they have a lot of money and the customer support line. Next year they'll say, "Here's Office 2001, just overwrite it," and so on, but that's expensive. I don't have a problem with that for personal productivity, because building a spreadsheet is not necessarily a collaborative task. There are some collaborative tasks there, but most of them are not. For personal productivity, you use your own machine. I have a really expensive computer. It's got a lot of power and lots of software stored in it, mostly only what I use. That's great. But when I interact with my enterprise, I use a free piece of software from Netscape, and that's it. That's the main difference.

I've beaten that horse. The point is that this internet computing architecture has changed everything. Now let me talk a little bit more about how things have changed industry-wide. I'll try to avoid repeating myself.

Every information system has lots of pieces. It's got hardware, it's got different classes of software, its got people, it's got all sorts of stuff. I'm just going to point to a couple of trends that I see.

First, at the base of everything is hardware (**Figure 5**), and it is polarizing. At one end, our hardware is getting very lightweight and small. I can check a stock quote on my pager against a very large and sophisticated information system somewhere out in the ether. My own machine is smaller, more specialized, lighter weight. Does it have an operating system? You betcha. Do you care what it is? I'll bet not. It doesn't matter, because it's focused around a particular task. I'm never going to install Microsoft Word on my pager, because it doesn't support it.

Hardware is getting smaller at one end and it's getting bigger at the other end. People are now interested in large servers or, at least, collaborating collections of servers that appear big from the outside, so we're seeing a polarization. This is why Windows 2000—you heard it here first—is doomed. It's too big for the desktop and it's too small for the enterprise. It's caught in the middle. People don't want big operating systems on their desktop or small operating systems in their enterprises.

The operating system has changed a lot as well (**Figure 6**). It's really being "commoditized." This has an operating system, that has an operating system, this other one has an

operating system. Some of the watches in the room probably have operating systems. Do you know or care which one it is? No. I can get the same stock quote *here* that I can get *there*. It doesn't matter, because it's being commoditized on the server as well. What operating system does Amazon use? Quick, quick, quick! Who knows? Who cares? What operating system do you have? Who knows? Who cares? It doesn't matter as long as it can publish its information via these standards.

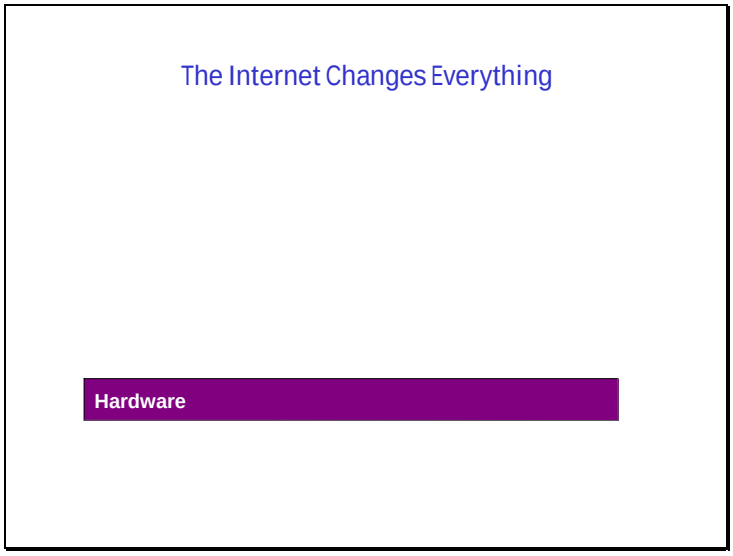

Figure 5

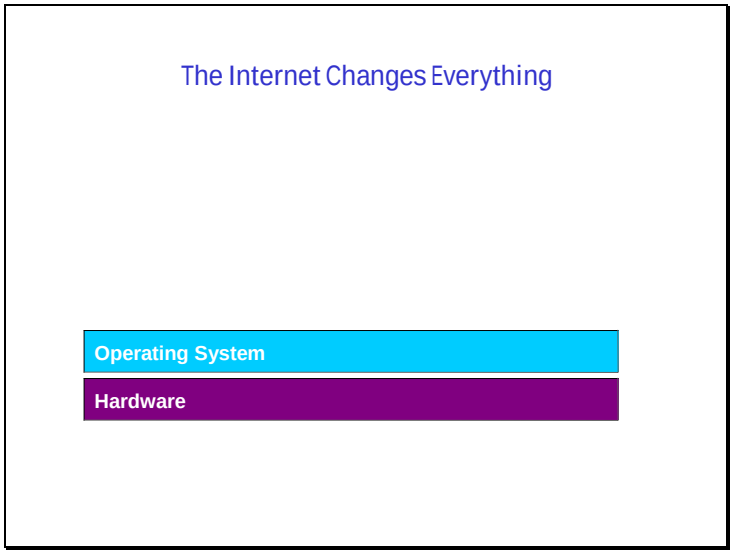

Figure 6

In the database world, in our space, we're starting to see a bit of a change (**Figure 7**). We were going down the path of commoditization as well. People said, "Who cares? It's a place to store stuff. Put your bits in there, put your bits in here. Pick the cheapest one." It's come back from that a little bit because people are now starting to put their application functionality in the database, so there's a whole new way of differentiating the way your application and your data interact as an information system. That's changed the whole process quite a bit.

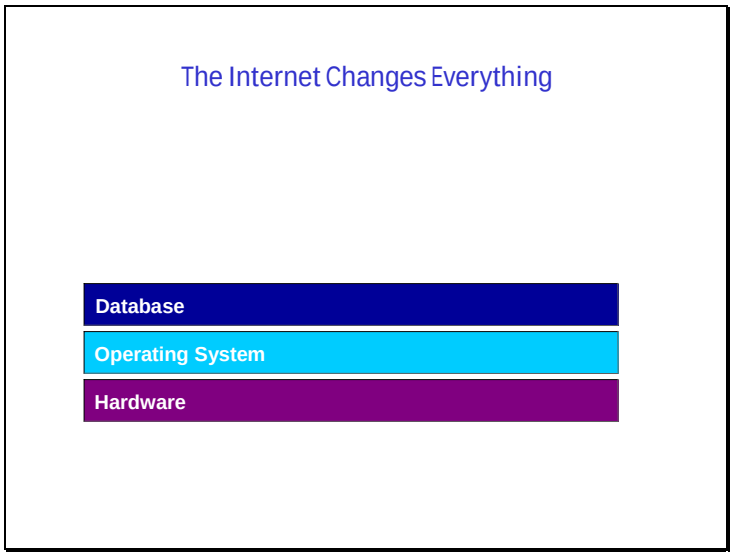

Figure 7

There's also a whole industry around building software for writing applications, whether it's a language like Java or C, or something fancier like Windows-based tools, or whatever else (**Figure 8**). Here, the big deal was how companies were able to make the transition to the Web,

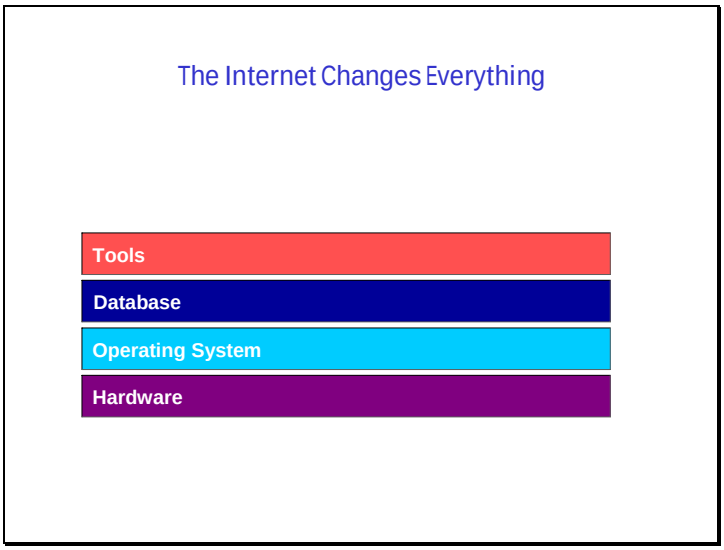

Figure 8

whether they were able to build software that could deploy to Java or whatever. It's funny: more than in any other space, the players in this are totally different from just two years ago. A whole bunch of the client/server world either hit the wall or totally changed direction. This whole industry has changed.

There's also a whole industry around selling COTS, commercial off-the-shelf software (**Figure 9**). Everybody gets a general ledger. Why? Because everybody is building general ledgers. You don't buy tools to build a general ledger, you buy a general ledger. In this applications space there are companies such as SAP, PeopleSoft, Bonn, Oracle, and so on. Most important here is to remember what I said about the architecture: that these were all built with tools. The architecture of the tool, whether it could go to the Web or not, had a big influence on whether the application could go to the Web or not, and that's changed the landscape a great deal.

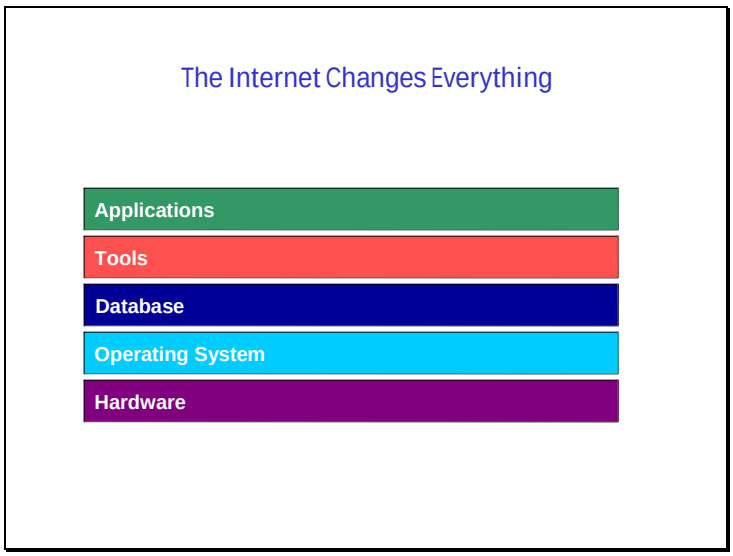

Figure 9

Their people have changed a lot. The expectations of our user community are very different (**Figure 10**). "You mean to tell me that with three clicks I can find Tim Hoechst's address and send him flowers, yet I can't determine how many people work in my organization?!" Because the internet has made so much accessible and easy to them, people don't understand why this huge information technology [IT] department we have can't do at least what all these Linux hackers are doing out on the Web.

Also, the users are already trained. How many of you went to "How to Buy a Book on Amazon" training? Training here is, "Click on the blue underlined things. Any other questions?" It's simple. They all work the same way. We have common access to stuff. I'll bet you've seen this as you started to deploy information to the Web. You don't teach somebody, just point at it and give them five minutes and they're finding everything they can. That's a huge difference from the great expense that we used to incur.

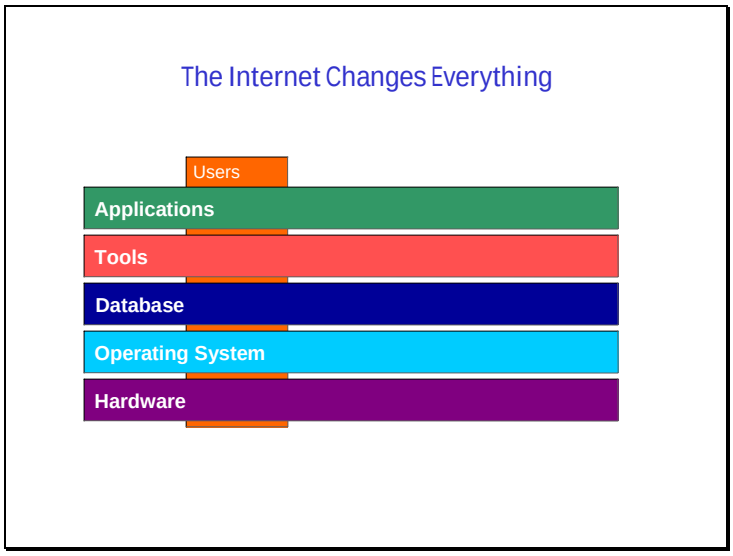

Figure 10

Consultants are a huge part of the way we build information systems, especially in the government (**Figure 11**). Maybe I should point a finger a little bit more at the integrator community. These are people who are paid for things to take a long time. These are people who have no interest, necessarily, in having short-time, high-value solutions, but they're having to reinvent themselves because the old models simply won't hold up anymore.

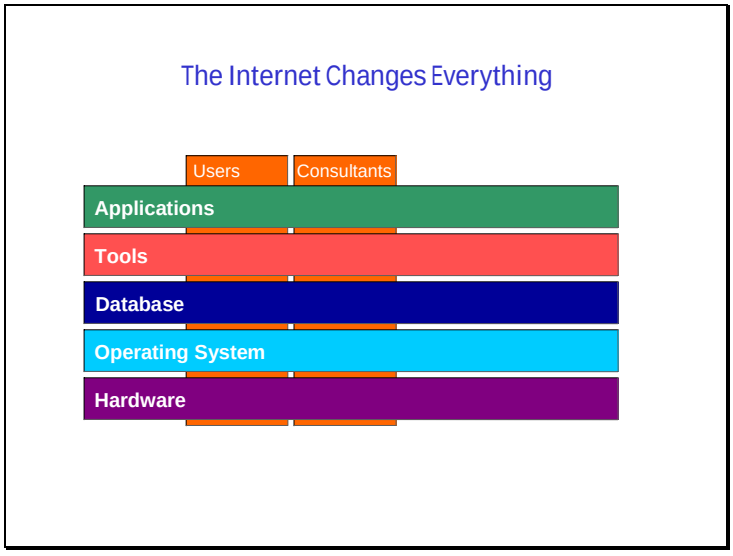

Figure 11

We're also seeing that developers are more commonly becoming implementers of off-the-shelf software rather than builders of new software (**Figure 12**). They have changed everything. Developers no longer write big pieces of software. Everything is incremental. Again, it's the idea that they're feeding an organism rather than creating something that exists at a point in time. If

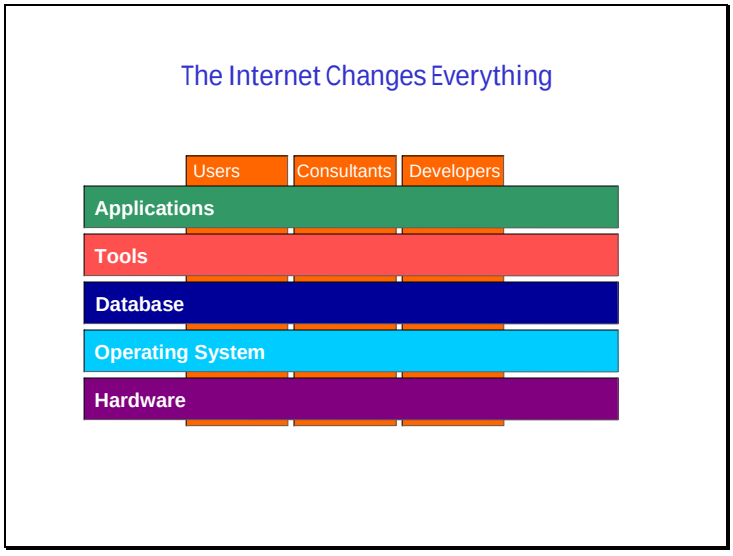

Figure 12

you go into a bookstore like the Coop, you can't swing a stick without hitting a Java book. There are more Java programmers than there are Windows programmers. Whether the language is any good or not, who cares? It's fine. There's a bunch of people who know it. That's a good thing. I say it happens to be a fine language, as a language. That has changed the development mode a lot.

We wanted to hire a Web developer who had a visual sense for building Web sites, because we're building a Web site for educating kids. We were searching the Web, thinking we'd find some cool sites and then contact the people who wrote them and see if we could hire them. We found one called phong.com. You should go to it. Just visually, it's really a cool Web site. It's got all sorts of lessons on how to build Web things. We were saying, "This is brilliant! This is great!" So we contacted the developer, and we said, "We want to hire you. We're a big company." We went through the whole courtship. He said "That's great. I'm only seventeen, so, in six years, when I finish college, maybe I would love to hear from you." The whole world has changed here in what people can do rapidly.

Finally, the internet has changed the role of the administrative setup for IT (**Figure 13**). These were folks who, through two decades, were saying, "I want to keep inside my clerk-less data center." It is cool again to have big, centralized computers managed by professionals.

We're also going to see a big shift in that area. Microsoft and Cisco, to a large extent, are creating a huge population of middle managers with technical knowledge. They're creating network and systems engineers for mid-tier systems. What we're starting to see is that large enterprises don't want ten or twenty or fifty of them. It's too expensive. You'd rather have one administrator whom you pay $1 million a year because he or she is extraordinary, than have nine or a hundred administrators whom you pay $10,000 or $50,000 each. Because of the centralized complexity, you want fewer people managing your systems, but they're more expensive people. This is critical, because there simply aren't enough people to go around. We have negative unemployment in our industry.

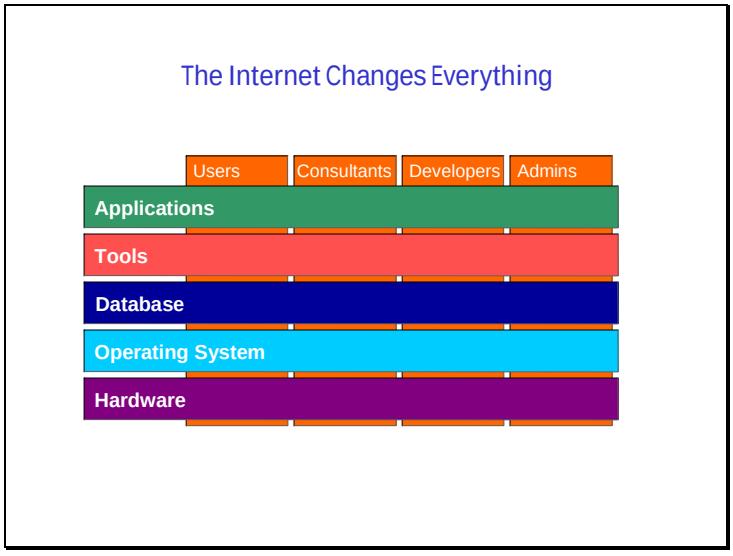

Figure 13

I'm now going to tell a story. If the whole Gutenberg humanities discussion were at one end of the spectrum, this example would be a very specific problem at the other, but I think it articulates a little bit about what I see happening with every one of our customers.

I personally had an IT problem, which was that we had a phone directory that we distributed throughout our company. Every six months you would get a book and it had everybody's name and phone number and location in it. The first thing you would do is grab it and check if your information was correct. If your number was wrong, you'd say, "Now it's going to be wrong for six months. I'm going to have to deal with this."

This was a bad idea in many other ways. First, you'd try to throw away the old one, and they'd say, "No, you can't throw it away. You've got to recycle it." You'd go out in the hall and you'd see these big bins and you'd know just how many trees we had killed to print this book. It was fairly inconvenient, because when I was at work it was at home and when I was at home it was in my briefcase in my car. I never had it with me. Finally, it was a huge security risk. Remember, I said our people are sought after. Recruiters could decimate us if they got hold of this book. This was just bad.

Meanwhile, my boss was giving me grief because he said I was no longer technical. He said, "You're just managing people. You can't do IT anymore." I said, "I'll show you that, yes, I can." So, I looked for a problem and this was my problem. I thought it was pretty simple. We have an HR database (I already referred to it). It's got everyone's phone number in it. Wouldn't it be cool if I could just look up people's phone numbers?

Allen: Put it in your Palm Pilot.

Hoechst: Put it in my Pilot. We'll get there.

I swear I was just trying to solve this problem for myself and to prove to my boss that I could still at least tinker with technology. I sought to figure it out, and I went to our HR data, which at the time looked like this screen (**Figure 14**). It was terminal based. This is the way our

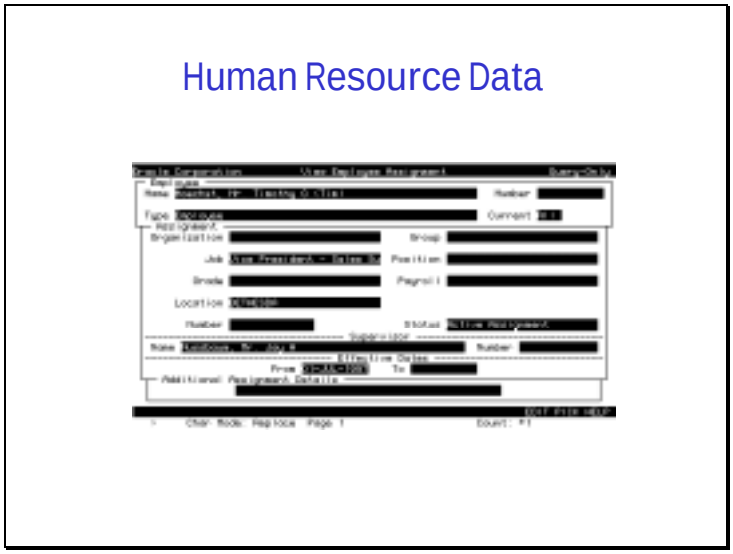

Figure 14

HR people saw my information in a system like this. As fancy as your interfaces to data are, today most people still see their data through systems like this.

I asked, “Can I get access to the data, because I want to be able to search and find people’s names and their phone numbers?” I built a system where, via an internal Oracle Web site that we call @ria, I could put in a little search string and, *boom*, it gave me the name and phone number (**Figure 15**). Done. My whole goal was speed, so this was very fast. By the time you lifted the return key, you had the phone number, even though we have 40,000 people. Problem solved. I showed it around. Everyone thought it was great.

Figure 15

This is when all my problems began. I didn't put this in public space, but it is in Oracle's public space, so people started calling me with the, "Wouldn't it be cool if…?" syndrome. "Wouldn't it be cool if we could show more detailed information than our phones?" I said, "Okay." I still had a little energy around the project. I didn't have to redeploy anything. I just changed it, and we started to show more detail. This was the first time I used data that were stored locally. Everything else I got out of HR. I thought, "Wouldn't it be cool if we let people add their pictures?" Great. Now I had more detailed stuff (**Figure 16**).

Figure 16

Then someone said, "I know you're a manager. Wouldn't it be cool if I could represent that?" Great. I worked on it a little bit more, and now it showed an organization chart based on HR data (**Figure 17**). You could navigate up and down. Everything was cool.

This is when HR first called me, and they asked, "How are you doing that?" I said, "What do you mean? I'm just taking your data and representing them here on a Web site." I should preface this by saying that these data are public inside Oracle. That means that we have this little @ria database, and anybody can extract the data and do what they want with it, but there's no private data on it like salaries or anything. It's not on the internet.

HR didn't even know I was doing this. They started coming to me, and I asked, "How did you find out about it?" HR said, "What do you mean? We use it all the time. This is how we look up people's information." I said, "What about your system?" "Oh, ours is slow, or ours doesn't show the picture, or ours certainly doesn't show the hierarchy." "What do you mean? Just how difficult is this?" "Well, our system isn't doing it."

We have a big global IT organization, and it had started a project that is defining a Web site for looking up people's information. They had forty people on it, and they had been working six, eight, or twelve months and didn't have anything yet. The HR people were waiting for that, and here this thing had popped up so people started using it.

Figure 17

Then a guy who works for me said, “I’ve had a phone call from Tony Oettinger and I don’t who he is. Wouldn’t it be cool if I could relate him to you organizationally, so I could tell whether I need to help him or not?” This blew HR’s mind (**Figure 18**). It’s all their data, but we just made them more available.

Figure 18

Then we went too far. I took some of the public data, and I started showing people’s job histories, the idea being that if you called me up asking for a job I could find out whom you worked for before and call that person (**Figure 19**). I activated this feature on Tuesday afternoon. On Wednesday morning, I was walking past our vice president of HR’s office. She was on the phone, and she whispered, “Tim, come here. Are we showing people’s job histories on the Web?”

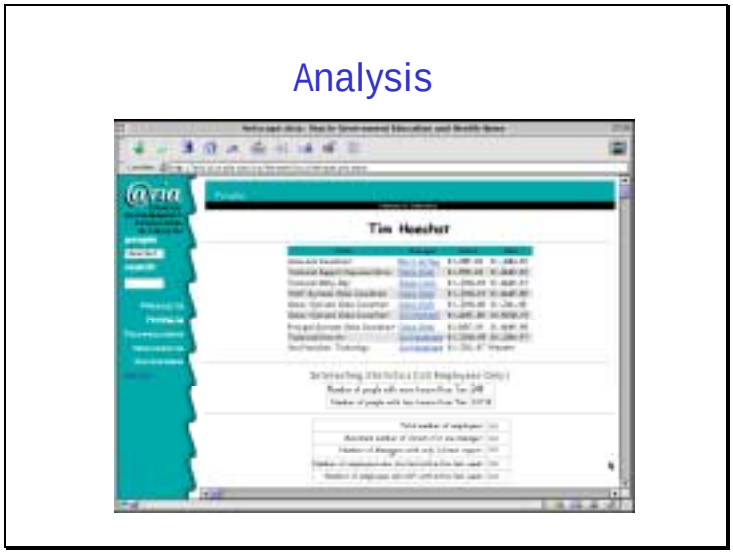

Figure 19

I said, “Yes, isn’t it cool?” “I have the CFO on the line. We can’t do that! It’s private information.” “It’s not private! It’s in this public database.” I had taken public data and turned them into information that wasn’t public. She asked, “Can we turn the system off?” I said, “Hold on.” I undeployed it instantly, and before she hung up the phone, I said, “It’s gone.” She said, “We’re not showing people’s private information on the Web.” We were able to undeploy something as quickly as we deployed it. We went a little too far. Too bad! We pulled back.

I must have gotten a thousand phone calls from people saying, “Why did you take this off?” This is when I realized how many people were using this system. It had been up for only twelve hours. I never advertised the new feature.

To make it more complex, we made a deal with Fidelity, which wanted to get into the HR outsourcing business. We said, “You use our software, we’ll be your customer,” and now they outsource to other people. A woman called Fidelity and said, “My 401k statements are going to the wrong address because it’s wrong in your system,” and Fidelity told her that they were getting their information from this site. They are the masters of the information, I get it way downstream, and I just show it. But so many people used the system because we made access to the information so available, so quick, so simple, so fast, so reliable—all the things I listed in the beginning—that all of a sudden, people thought that system was the source.

To make a long story short, this system gets about three million hits a week. I’m The Man for maintaining it. I’m the guy people call when their phone number is wrong. Every day I get issues. I get: “My phone number’s wrong. My address is wrong. The length of time I’ve been here is wrong.” Yadda, yadda, yadda, yadda. I’ve learned the problems of all these databases around the world because people say, “How come I’m not in there?” I say, “Well, you live in a country that has no HR system. What do you want me to do?” The complexities came about because all I tried to do was show people’s phone numbers.

It actually got a little bit more complex. We showed maps to people’s homes (**Figure 20**). That went really too far, but all I did was cut and paste into Yahoo’s maps. Anybody could do it. This is called security through obscurity. I can find out what any of my neighbors paid for their

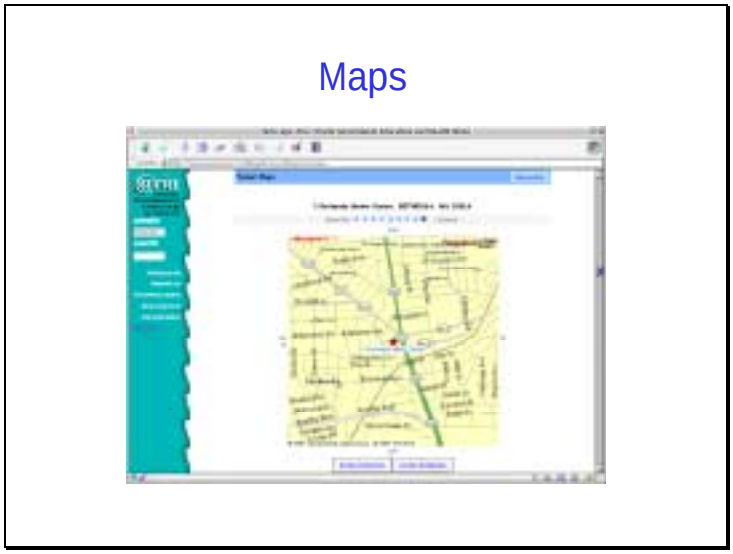

Figure 20

house if I'm willing to go dig through the county records, but I don't because it's a pain. If somebody publishes it on the Web, all of a sudden it's a little too easy. Is it secure? Maybe.

We also had all kinds of different systems that began relying on my system (**Figure 21**). Remember, I said that now the systems can tie together. So everybody and their sister, with the different systems we have, wanted to find out somebody's phone number or who their manager was so they could report to that person. This became a cornerstone in our information system. It was running on a tiny machine. We actually had to buy a bigger computer for this program because it's so active. Global IT is taking it over. When it's down, people get paged in the middle of the night.

This is how modern information systems are born. They are not born in an IT shop that says, "We know what the users need." They are born through innovation coming from the users. The successful organizations are the ones that can exploit a system and integrate it into their enterprises, because they've adopted standards. Anybody who has built a Web site in Oracle can call my people system for data if they want to.

I saw a demo of a system yesterday. It was a perfect example of the metaphorical "book" I've been discussing. One of my customers said, "We want you to see this cool new system," and they showed what a young fellow did with the data that he interacts with every day. The customer said, "Isn't this the greatest?" "Yes, but how do you make the system available to the other thousands of people who have the same problem?" "Well, we don't know. We have no infrastructure to share innovations." Whole enterprises grow up this way. This is a fundamental difference in the way information systems are built and the way they're maintained and the way they're deployed. It's wonderful, yet it's painful. This is the future.

When I talk to folks, I sometimes tell this story about my personnel information system and they say, "I can't look up people's phone numbers in my organization." Yet it's such a simple, trivial thing. People rally around the things that are popular. They don't go to the ones that are not. It's about that simple. It becomes the market economy for what works inside an enterprise.

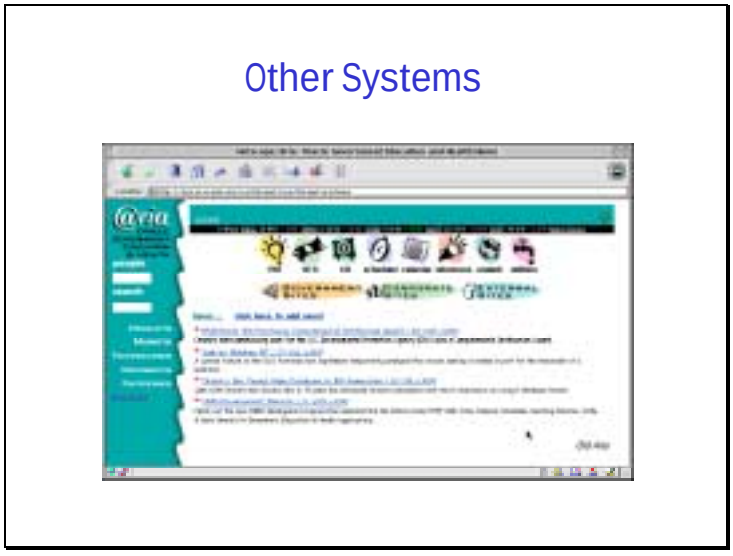

Figure 21

With that, let me summarize a little bit. I talked about a lot of things, but mostly this idea that the internet has not only changed the way we work and interact as people, but also has fundamentally changed the way we manage information. It has done so by turning upside down the model of how we store and disseminate information, by making it cheaper, and by making it possible for these information systems to work with one another.

I will leave you with a few bits of advice I give when we talk to people about how they ought to build information in an enterprise, whether it's a widely distributed enterprise or a focused one (**Figure 22**). First, scratch an itch. Modern information systems do not start by trying to solve all dermatological problems in one fell swoop. I couldn't look up phone numbers. I just scratched an itch, and things grew up around that.

Modern information systems are built in a network-centric way, rather than in a desktop-centric way. There are some exceptions, of course. Maybe you're on shipboard and you have no network, or your computer is in your backpack. In the military, we have discontinuous networks and security issues, and there are technologies to allow them to participate. But as a whole, the network-centric approach gives us huge benefits and huge flexibility (we talked about the flexibility design) to take on whatever comes next.

I know we have this problem, and I know you have this problem, which is that IT people try to over-engineer a problem. I call it the ten-yard-line effect. What happens is that the team is building a piece of software, and they're really interested by the problem. You have smart people, they're running down the field, they get to the ten-yard line, and now it's boring stuff. "I've got to write the installer. I've got to write the documentation. I've got to debug it down to the last little bug. I'm going to make this problem hard again. I'm going to figure out a way to expand the problem so that I'm still interested by it."

We have this problem at Oracle. We build a product and I see a demo of it. I say, "That's done. Ship it. Let's go. Let's get it out to the customers." Eighteen months later, it's still not there, and they've added huge amounts of stuff and complexity to it just because the people working on

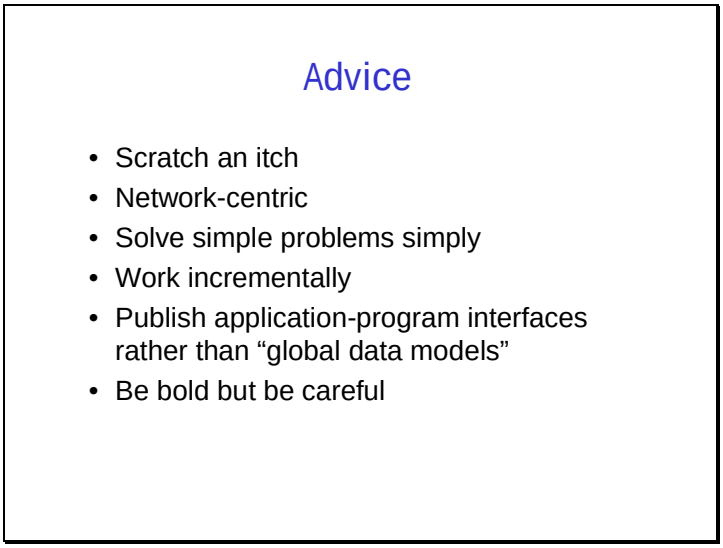

Figure 22

it need to keep it interesting, I think. When you solve simple problems simply, you can quickly get on to the next thing.

Working incrementally is very much like this. Systems will build up. In the government we have a large procurement process. I understand why it is that way—for competition and legislative purposes—but the reality is it paralyzes the process. People believe that because it’s so complicated, they’ve got to do it in one fell swoop. The reality is that you need a process that allows us to try stuff and say, “It works,” or “It didn’t work.” Lots of little failures for every success are great. Then we build on the things that worked well.

How do we make sure our systems work together? We make sure that every system we write has a way of communicating via a standard, whether it’s HTTP or something more complicated, rather than that we’ll get everybody talking on the same page. As information flows more freely, more problems will arise, just like my showing people’s job histories.

An application-program interface is a function that you would call, like, “Hey, Social Services, if I give you a Social Security number, can you return true or false to a question?” The programming interface just asks that question. So, that’s important. We’ve got to be bold and take these steps and try to do new things, but we’ve got to be careful and realize that sometimes we’ll go too far. It’s easy to come back from that. Does it make sense? This is really what we’re telling people. The reason we tell people that is because the internet has changed everything.

Student: It sounds as though a lot of the issues you highlighted are people-oriented issues, not technology ones. You said that technology really can solve most of the integrated tasks we need to perform for the future, but people are inhibiting the process. How do we get at that issue? Is it a leadership function? Is it more knowledge-based applications? Is it knowledge management? What needs to happen to get the people part of this to integrate with the technology that’s already there?

Hoechst: First, I’m happy that’s what you heard me say.

Student: That’s the message I got. Is that the message you sent out?

Hoechst: That's one of the things I wanted to say. Second, I have no idea. The challenge of getting people to change the way they do things is great indeed. It's not just changing the way they manage systems, which happens anyway because things get out of date and they sort of have to, but changing the way they run their enterprise, their business, or their government, or how they meet their mission. There are a lot of people who think the way we've been doing it is just dandy and will stay that way for a long time. A lot of the issues were not just technical; they were social issues with people realizing we now have a different kind of environment in which we work.

Student: Political and cultural issues have a lot of inertia in technology.

Hoechst: That's right.

Oettinger: Let me challenge you in this, because it sounds rather negative: that people are the problem. I want to remind you again of the first reading you did in this course about balancing acts and so on,[2] because that was my starting point for writing that piece: the notion that the problems that both Charlie and Tim have described are perennial. When you start seeing something that hasn't changed over a period of fifty years, does that mean that everybody is so damn stupid, or incompetent, or maladaptable, or inertial and so on, that it persists? When something persists over so many organizations for so long a time, I think there is more to it than just cussed people. To be fair, Tim hinted at that. There are other considerations.

In the presentation he was pushing the Oracle party line a little bit. That's okay. There are instances where you don't want something centralized. His prize example was that if it's something you're doing—you're calculating your own income tax, or you're writing poetry, and so on—you don't necessarily want to do that as a shared enterprise, so there is an argument for having it on your own machine. This has been hinted at in terms of the military: you want to have compartmentation to protect sources and methods. You want to have (he admitted it) a wall around the Oracle internal data, et cetera. Before you know it, when you start looking at the problem in the full-blown context of organization X, Y, or Z—whether it's a green field organization that's just inventing itself or an old one that's trying to change itself—it turns out it's more complicated than that and that there are sixteen different tradeoffs to be made.

Hoechst: That's true. Let me answer that. I don't think people are a problem, but collections of people somehow become a problem. While I do, to some extent, overstate the point to make the point, the reality is that organizational inertia exists everywhere. Any group, any organization that has more than two people, has inertia, and technology can't overcome that. However, it can help, or it can guide, or it can take old processes and streamline them and maybe give you the opportunity to recognize new processes. But the organizations that change are the ones that reject their inertia. In business, that's the new economy. It's like 3M saying, "We don't mine sand anymore, we're making sandpaper. We're not buying any gold, so we're going to make sandpaper." That kind of re-inventing yourself is rejecting the organizational inertia. I wish I knew why I can take two people into separate rooms and they'll both say the same thing, but if I put them together they don't. I don't know why. I'll figure that one out sometime.

---

[2]Anthony G. Oettinger, *Whence and Whither Intelligence, Command and Control? The Certainty of Uncertainty* (Cambridge, Mass.: Harvard University Program on Information Resources Policy, P-90-1, February 1990), [On-line]. URL: http: pirp.harvard.edu/pubs.html

Oettinger: I think it's more delicate than that because, again, there are the interactions with the customers. Think of some of the large, old economy companies that are trying to remake themselves. If they remake themselves too fast they lose money hand over fist, because their customers aren't in the new world yet. You see old-line phone companies still profitable twenty years after the AT&T breakup. Their managers are not stupid. They're looking out there and saying, "Yes, we understand about IP phone, et cetera, but some of our folks are still getting used to tone dialing. And, by the way, they're still willing to pay fifty cents a month for the privilege of tone dialing, even though they could do it themselves. They just bought a touch-tone phone and never told us, so they don't know the service is free." If the name of the game is to satisfy your stockholders, there is virtue in inertia. In other words, Tim characterizes inertia as a stark negative. In the real world, inertia can be an asset or can be a liability, like anything else.

Allen: Sometimes it becomes a real liability in government, where the technology moves rapidly. Programmatically and from the policy perspective, we have recently had extraordinarily sensitive computer network failures in our most critical agency. We had similar problems in the summer and fall of 1999 in another major agency, where the failure impaired computer network transformation technology. The organization and management of that failed, and we're still coping. I don't know what lessons the private sector can bring, but it seems to me that in the intelligence community, at least, we're having some extraordinary difficulties.

Hoechst: That's right. I can't bring myself to say that inertia is okay. I think you're bringing up examples where the fact that a customer of the company has more inertia than the company itself allows the company to survive. That's dandy, but all I know is that our enemies in his world are not waiting around for us to give them total dialup.

Oettinger: I'm not trying to be Panglossian. Absolutely, there are times when inertia is fatal. But let me give you an example of this that impressed me. There was a truck driving down the street with the logos of all sorts of dead brands on it—brands that you guys never heard of. He was stopped across the street, and I started a conversation: "What the hell are you doing?" "I bought up maintenance contracts for all these obsolete brands." I said, "Tell me about it." He was an excellent entrepreneur. He had calculated his remaining lifetime and amortized all that's left, so he bought the franchises for all of that stuff and he figured he'd be dead before his income was. He got himself a very good annuity, built essentially on his speculation on inertia.

I said to myself, "This is not stupid at all." That's how you build every company in the world, but this guy was not creating a dot-com. He was doing a reverse, and giving a backward look, and discounting his own life expectancy in the way one discounts cash flows and all the rest, and he had a business. Maybe it was a very small fraction of the whole, but riding inertia is not necessarily a bad strategy, even though, in the context we're talking about here, it may be catastrophic.

Hoechst: I think what's more important, though, is to understand that inertia exists.

Oettinger: There we agree completely. It is an absolutely critical factor. Anybody who fails to recognize that inertia is a major factor is really doomed, especially in this crazy dot-com world.

Hoechst: No matter how big or small you are, one of the things I like to do is say, "I feel your pain." I don't just say, "You guys are big and slow and we're nimble and young," because most of the stories that I tell are about our own inertia. Yet we have stockholders, and I talk to some of the government folks and they say, "Well, we can't just fire people," or, "We can't give people

bonuses if they do good work and not if they don't," and all of these sorts of things. That doesn't prevent inertia.

However, maybe it can ease it if you recognize that it's there. That recognition and acceptance are the first lesson. The second lesson is, how do we work within that inertial framework? The problem comes back to the original question, which is that most of the issues, when getting to the goal of that simple access to technology, are social issues, not technology issues. The technology exists to do it. The social issues related to getting it done are very pertinent.

Oettinger: Let me engage you in a little gloss on this to see if you agree. If you look at the literature on technology and innovation, et cetera, the first chapter of every book on the history of technology and technical innovation, is "What Is Technology?" As you might expect, there are narrow-spectrum folks and wide-spectrum folks. I guess I'm at the wide-spectrum end, and I think, if I hear Tim and Charlie correctly, they are as well. The narrower spectrum end looks at technology as a gadget. The more ecumenical among them will say it's not just a physical gadget and it may also include software. It took fifty years for people to figure out that technology was not just a hammer, but it could be a piece of software and it could, in fact, reside somewhere off in a network and all that kind of stuff.

But then, what good is a tool in the absence of a tool user? What is a tool user? It's rarely a single person, so most tool users are organizations, and most organizations aren't handed down by eons of evolution. They are creatures of a Larry Ellison or the Founding Fathers of the United States or whatever. They are, themselves, engineered artifacts.

In that broad end of technology, you have one hell of a huge design problem, which is the relationship between these tools and the organizations in which they are embedded. You really don't have much that's worth a damn unless you look at the whole. The minute you look at the whole, you have such an order of magnitude that if you try to design it from scratch, you'll never get there, which is really the deep import of his "the internet is everything." It is that conception that large systems that are intractable otherwise can be incrementally improved and evolved without creating catastrophic delays or catastrophic errors. Using examples about "creating and backing off" is critical. That capability, in this neck of the woods, hasn't existed before. That's the importance of what he's presenting: that it enables an evolutionary path for an organization, rather than a cataclysmically revolutionary one or complete stasis.

Hoechst: Absolutely. That's right.

Student: I think I hear both you and Dr. Oettinger writing inertia off as an organizational issue, but in the case, for instance, of the Department of Defense or the National Aeronautics and Space Administration, there is a very good reason why they don't immediately adopt the latest and greatest. Maybe Oracle is not a good company to pick on, but is Windows 98 really more reliable than DOS? How often do you get a blue screen? If you have lives at stake, do you really want to try to overcome this organizational inertia and run the latest and greatest?

Hoechst: You're absolutely right that it plays a role. I don't want to suggest that you should always institute the latest and greatest technology. Coming from a company that likes to think we sell the latest and greatest technology, it's not always the most prudent thing to use it. It's probably prudent to tinker with it in preparation for when it has matured, or else you'll never get anywhere.

That's my whole idea of this procurement process. The government makes us bid current software. We're not allowed to bid software that does not yet exist, yet the actual procurement won't happen for eighteen months. So, by definition, they're buying software that's eighteen months old. They're at the opposite end of that chain. Yet the federal government also has our labs that are doing most of the advanced work. Our most advanced customers and our least advanced customers are all in one box. I'm not suggesting that we always just fire up the latest and greatest, although I do that. The moment Apple says, "We have this new thing," I go on the Web site, I fire it up, and I just live with the consequences because I'm a self-maintaining guy. I like that. I'm an early adopter. But organizations, especially large ones, can't always afford to be early adopters.

I had a conversation with the fellow sitting next to me on the plane today, who was a pilot. My flight was cancelled and so was his, so he was riding up to take his next flight. Every time the landing gear went down I could see flames shooting out, and he would say, "No, it's supposed to work that way." I really enjoyed having a dialogue with him. We started talking about computers, which tends to happen to me a lot. I don't know why. We actually made a point, which was that I want to make sure the software behind the dashboard of that plane is tried and true and reliable and tested. He said, "It's buggier than you think," which did not give me any confidence, but it's all a matter of perspective. His initial point was, "Do you know that all of the most advanced computers in the USAir fleet of airplanes, in the 737, maybe the 707, have a 286 chip in them?" I said, "That's fine. That is a very well understood piece of technology. It has bugs, but we know exactly what they are and so we can make sure we build something that works." There's always the opportunity to make sure that you build something reliable.

My point here, though, is that incremental innovation also works. There are millions of people doing this every day. Does it work in every case, always? Of course not. But for the dissemination of financial data, or payroll checks, or whatever, this works great. Would I want to rely on some new wireless network that just got fired up last week for our soldiers to have connectivity back to their command? No. But in a year or two, maybe. It's a balance.

Oettinger: You can do the experiment yourselves. If you buy Quicken, you can do your income tax on your own machine, and then you have the option of printing it out in paper form and sending it to the IRS by traditional mail or you can file it electronically, by paying Quicken $9.95 or something like that for the federal form and $4.95 for the state. You can use your browser, as Tim has suggested, and go to www.quicken.com and use their software, which is centrally kept, and up to date, so you don't have check to see if you have the latest version. By the way, folks complain that the network is so slow that it is very slow indeed to do your tax by going in and using that capability. From a privacy point of view, whether you file electronically or you use this central thing, either way you've given all your financial information to quicken.com, as well as to the IRS. Whom do you trust more?

In that microcosm, you can take a look at a whole bunch of social, commercial, and technological issues. Do the experiment yourself for a maximum cost of about $29 for the software and another $15 if you want to file electronically. Do it! It's a practical experiment.

Hoechst: I'm a pragmatist, though. I trust the internet to know my tax info.

Oettinger: I do, too. I file electronically.

Hoechst: One of the things I often ask a lot of our customers is, “Who’s bought something over the internet?” The number is larger every time, but half have done it. “For those of you who didn’t, why not?” Everyone says, “Security. I’m afraid of my credit card number being revealed.”

Oettinger: But they give that credit card to some waiter in a sleazebag restaurant ...

Student: …and how do you think that waiter runs the check on you? He sends the data over the Internet.

Hoechst: Exactly, or he photocopies it and then he does whatever he wants with it. I believe that the slowness of the networks is a passing thing. Very soon, network bandwidth won’t be relevant. It will be ubiquitous. Anyway, I think it’s a good experiment, but I also think that this is a train that’s leaving the station. This is where we’re going, and you can exploit it or not.

Oettinger: Yes. I accept that statement without contradicting you, only because the way you’ve presented it is sufficiently abstract: that it is not the “Internet today” technology, but the principle of the thing.

Hoechst: Yes. This is especially important. When I say “internet,” I mean lower case “i.”

Oettinger: He’s not talking about the current implementation. Otherwise, everything he said would be totally ridiculous.

Hoechst: I’m talking about the concept. When I built our little personnel system, it was on Oracle’s private intranet. It uses the same standards, but it is not on the public internet, which has different implications. But it still works.

Oettinger: It’s the architecture he’s referring to, not necessarily the details of the contemporary implementation. That’s important, though.

Hoechst: Was this discussion interesting? I knew it was out of context for what you normally talk about, but I didn’t know if it would be interesting or not.

Allen: It was in context.

Oettinger: Oh, yes. As you know, this was totally unplanned and happened because of various scheduling constraints, but I must say that if I had had to plan it, I wouldn’t have had the wit to do it. I am very grateful to both you and Charlie for sticking around for each other’s presentations, because it created a complementarity and a richness in the presentations and the discussions that we would not have had otherwise. I have something for you, too: a token of our large appreciation.

Acronyms

CFO	chief financial officer
CORBA	Common Object Request Broker Architecture
COTS	commercial off-the-shelf
DHTML	Dynamic Hyper Text Markup Language
DMV	Department of Motor Vehicles
HR	human resources
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
IP	Internet Protocol
IT	information technology
PC	personal computer
XML	eXtensible Markup Language

INCSEMINAR2000

ISBN 1-879716-74-7