

Schussel's

DOWN SIZING JOURNAL

July 1993

In This Issue:

Report from SOFTWARE WORLD *Part II of II*

Editor Schussel acted as reporter for the day when he attended SOFTWARE WORLD. This is his report on some of the show's highlights including speakers Larry DeBoever and Ed Yourdon1

After All, What's the Object of Your Database? *Part I of II*

Editor Schussel compares relational database technology to the newer object oriented database technologies, and gives appropriate examples of uses for each1

Textile Manufacturer Celebrates 100th Anniversary with Strategic Downsizing

A look at one large manufacturer's downsizing from an IBM 3081 mainframe to an IBM AS/400 midrange system6

Upcoming Downsizing Events16

Report from SOFTWARE WORLD

Part II of II

S OFTWARE WORLD is the largest computer software-oriented

conference and exposition—this year there were over 7,000 attendees at the show. In May, I was in Toronto, co-chairing this event with Ed Yourdon of *American Programmer*, and Roger Burlton of SRI. I managed to put in a couple of days of conference attendance and so in this article, I am reporting the highlights from conference sessions I attended. Last month's issue featured points from Will Zachmann of Canopus Research, Mary Loomis of Versant Object Technology, John Tarbox of Caanan Analytics, and Tim Lister of Atlantic Systems Guild. In this month's issue, the speakers highlighted are

(continued on page 9)

After All, What's the Object of Your Database?

Part I of II

W hile the focus in the database world

over the last several years has been on relational technology, there has been much discussion recently about object oriented (OO) approaches. The \$64,000 question is, will object oriented DBMSs replace relational in the same way that relational DBMSs were meant to replace hierarchical systems?

As far as true database engines are concerned, experts believe that most production data is still stored in various types of older, hierarchical, inverted, or network structures such as

(continued on next page)

After All, What's the Object ...

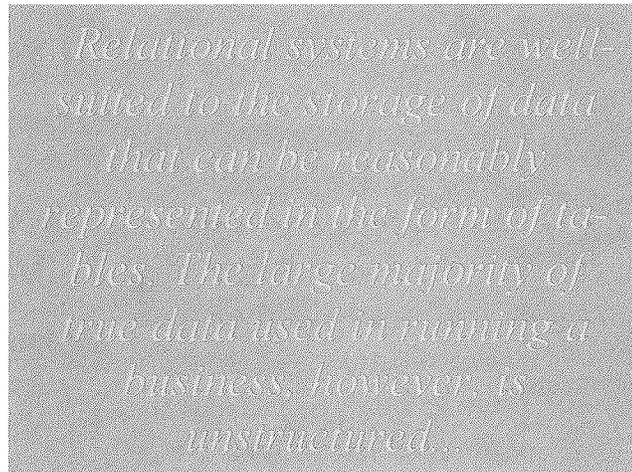
(continued from front page)

IMS, ADABAS, or IDMS. These types of systems predominated in the 1970–1985 time-frame; this was the era in which huge investments were made in building the strategic, on-line systems that now run most of the world's industries and commerce.

By 1985, the *new* (actually it had been first proposed by Dr. Edgar Codd in a 1970 ACM paper) relational model and its leading product implementations, DB2, Oracle, and Ingres, had badly beaten hierarchical and network database approaches in the public's mind. The relational model proposed that a simple two dimensional series of tables be the conceptual data model for programmers and users. This form of representation was easier for users to understand. Another major force behind relational was the fact that its approach to data management was widely supported by many different commercial and academic forces. Researchers at schools including the University of California, Berkeley, and the University of

Toronto combined with early business implementors such as IBM, DEC, Oracle, and Ingres, and together loudly proclaimed the advantages of the relational model.

Another essential element in relational's ascendancy to dominance was the fact that IBM's relational language, SQL, was placed into the public domain by an enlightened IBM. This meant that ANSI could choose SQL as the relational



Relational systems are well-suited to the storage of data that can be reasonably represented in the form of tables. The large majority of true data used in running a business, however, is unstructured.

language standard. As a result, most of the major relational DBMS vendors were free to implement their own versions of SQL in their DBMS products. For the first time in history, a single language (although in multiple dialects) was used for database access across multiple platforms.

A seminal point in the public debate over database standards was the publication of a two part article by Dr. Codd rating various

existing systems on their adherence to the relational model in *Computerworld*, October 1985. In his article series, Dr. Codd pointed out that IBM's relational DBMS, DB2, had moderately adhered to the precepts of relationality, while its competitors, Cullinet's IDMS/R and ADR's DATACOM, offered no more than a 20% compliance with the model. This was disastrous publicity for Cullinet and ADR because both organizations

had publicized and marketed their products as relational. Driving the final stake into the hierarchical/network position was an unfortunate letter written to the editor of *Computerworld* by Cullinet's president, John Cullinane, which castigated and criticized Dr. Codd's knowledge of real database issues. It was clear to people in the field that the future for non-relational DBMS had ended. And, in fact, within a few years, both Cullinet and ADR ceased to exist as independent software firms. Both of their product lines were absorbed into Computer Associates, which continues to support and market legacy products.

Although the vast majority (over 90%) of true DBMS purchase decisions

today are for relational products, these relational systems have not replaced the older hierarchical DBMS products in running most production systems. The conversion of such legacy applications has typically taken at least a decade in mature organizations.

What's wrong with relational?

The movement of older applications to relational has been slow for a number of reasons. In most cases, the older applications were built on mainframe hardware and with mainframe-centric logic. Simply converting these applications to relational could introduce serious cost increases because of longer run times—after all, most relational implementations are now on minicomputers, servers, and PCs. The high levels of robustness and data integrity that are typically built into these older production systems may sometimes be a strain to reproduce at reasonable costs with a relational system. And finally, it just may be that migration to relational offers no serious business benefit.

Relational systems are well-suited to the storage of data that can be reasonably represented in the form of tables. The large majority of true data used in running a business, however, is unstructured

information such as text, handwritten documents, pictures, videos, or sound. The relational model is particularly ill-suited for representing these types of data; there is no reasonable way to represent spatial or geographic maps, for example, in a relational database.

Another large problem with relational technology is also one of its strengths: normalization. The relational model assumes that information about a physical object can be broken down and stored as atomic elements. When it is accessed via query, the DBMS system recombines those atomic elements into a user view of data that represents information. This normalization process requires data to be separated from process. It also implies that data and process can be stored

and managed as if each had a separate existence. Even though in some cases this is a reasonable approach, in most situations it is not.

The typical relational system is created as system designers and implementors normalize data models and then store them as such. Referential integrity statements are then added (in systems that support this feature—without this feature, you have to write programs to perform the same job) to insure that the information in the database is universally consistent and correct. At run time, then, the system recombines the normalized data into usable and displayable physical forms which result in lengthy computer run times. It is also common that these lengthy traversal

(continued on next page)

Characteristic	Object DBMS	Relational DBMS
<i>Data Types</i>	Many, user or system	Limited, system conventional
<i>Data Structures</i>	Complex, user defined	Simple, tabular
<i>Typical Transaction</i>	Hours to weeks	TP or DSS
<i>Programmer Access</i>	Navigational	Value-based, SQL
<i>Programming Language</i>	C++, Smalltalk, LISP	3GLs, 4GLs, Windows, CASE
<i>Data Integrity</i>	Programmer defined	Entity, referential, triggers
<i>Inheritance</i>	Part of system	User developed
<i>Maturity, Utilities</i>	Low	High

After All, What's the Object ...

(continued from previous page)

paths result in unacceptable performance. If this is the case, then the systems implementors can "denormalize" or remove some of the system integrity rules to speed performance. Doing either of these, however, has an associated price which may show up as new applications are added to the database or as errors appear in the retrieved data.

The adherents of object orientation have a different set of tools which are very well-suited to solving some of the relational problems just mentioned. As we will see, however, the use of object oriented approaches introduces its own set of problems.

What is object orientation (OO)?

It isn't the point of this article to describe or define objects in technical detail; there is a great abundance of information available elsewhere on the subject. However, a short description is provided here for the sake of completeness.

In a very simplistic sense, the goal of object orientation is to allow software applications to be built using manufacturing assembly techniques. The idea is to have reusable software components (objects and classes) that can be maintained and enhanced through the technique of inheritance. The building of applications, then becomes a process of software component assembly.

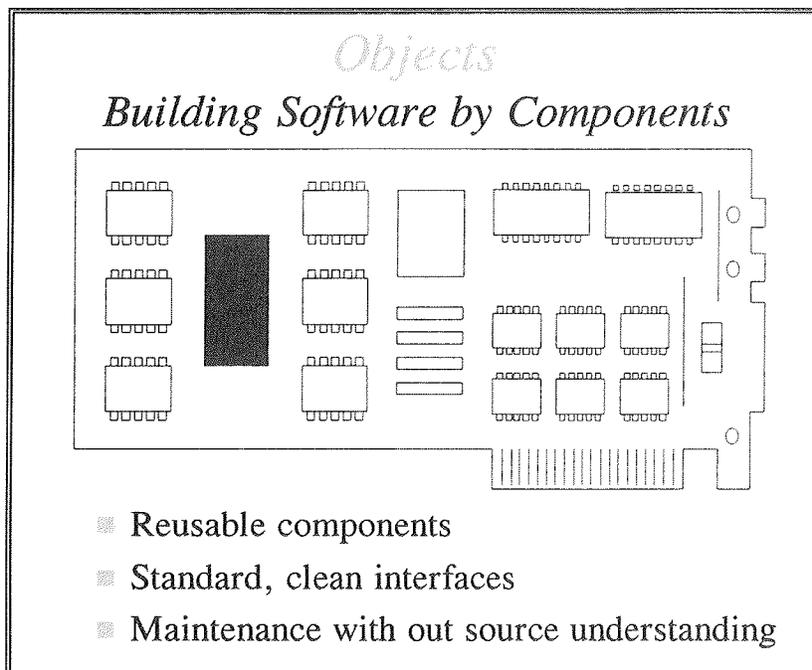
Object oriented development environments are characterized by four principals: encapsulation, inheritance, polymorphism, and late binding. A very short discourse on these principals follows:

Encapsulation refers to the way in which object oriented systems package data and processes together. The only way to access data is through the procedures that surround and maintain them. A mechanism known as messages, which is analogous to function calls, is used to access the procedures and functions of an object. Using an object's functionality only requires knowledge of the object's user interface, not of the object's internals.

Inheritance is the basic mechanism of code reusability. Objects can be organized into "Is a" classes and inherit functions from objects higher in that class. Appropriate maintenance changes are therefore automatically propagated throughout the hierarchy.

Polymorphism simply refers to the behavior of an object oriented system to messages. The same message can be sent to objects of different classes and elicit different but appropriate behavior in each case.

Late binding is an operational characteristic of object oriented systems.



Objects can be created on the fly by the application. Because of this, the system can't be "compile time bound."

Class libraries

OO doesn't offer any magic. What it does offer is a different view of programming organization. In an object oriented system, the programming language and DBMS are constructed to encourage the reuse of code and the building of class libraries. As for the productivity of application developers, that simply isn't going to happen until a development environment has a full, rich set of object classes that can be reused.

The basic assumption of object oriented development is that small groups (the brightest people) will do the difficult work of developing class libraries. Other groups will then assemble applica-

tion systems by using the developed class libraries. Developers of libraries and objects will be done by both software vendors and within businesses' IS organizations. At this point, the market split of work between generic objects by vendors and user specific objects built by IS departments is unknown.

With the currently available user experience, it's possible to formulate some observations about the building of class libraries:

1. It may take years for an organization to build an adequate set of class libraries for use in general purpose applications within businesses, or for very robust applications in the popular software categories.
2. An adequate, general purpose class library may consist of thousands of objects.

3. There needs to be a good discipline in place for the definition and addition of new objects.
4. The dreaded private library syndrome must be avoided if group productivity is the goal.
5. A system of naming standards must be in place for the class library.
6. Success with OO requires a set of functions for library management. Included, of course, has to be an effective way for the searching of objects, perhaps something along the lines of an electronic soundex. *gd*

This is the first in a two-part article on relational and object oriented DBMSs. Next month's article will cover the benefits of object oriented development as well as the problems with the object paradigm.

Benefits of Object Oriented Development

- Programming by component assembly
- Formalism for interoperability and modularity
- Stronger type checking
- Enhances reusability
- Systems can closely model real world
- Simpler maintenance
- Can handle complex or unusual data types
- Can handle certain tough performance problems

Textile Manufacturer Celebrates 100th Anniversary with Strategic Downsizing

When the 6,000 employees of Greenwood Mills celebrated the textile company's 100th anniversary in 1989, they received an unusual birthday present: a new direction.

Fierce global competition through the 1980s had forced a major restructuring in the textile industry. No company was immune; even companies with household names such as J.P. Stevens, West Point Pepperell, and Burlington Industries, were either reorganized or downsized. Only organizations like Greenwood Mills—those who aggressively cut costs and shifted markets—survived. "The competitive companies like ours are staying in," explains Jack M. Hill, Director of System Services at Greenwood Mills, Inc. in Greenwood, South Carolina. "But

we had to make difficult decisions and invest in new technology. We had to become more efficient and give our customers much more support."

The virtues of plain cloth, denim, and downsizing

Greenwood Mills made some critical strategy changes in 1989; top management refocused the corporate product strategy to concentrate on the production of "gray cloth" and denim, rather than the finished cloth they had previously sold. Today, Greenwood Mills is the largest producer of for-sale gray cloth in the United States. Gray cloth is typically sold in million-yard orders; most of Greenwood Mills gray cloth consumers are from the clothing and home furnishing industries. Seven of the company's nine weaving plants produce the material, and the market continues to expand. In 1986 and 1987, the company also purchased two denim weaving plants. It has now added three "wet-processing" facilities to dye constructed garments. Their customers are major U.S. jeans manufacturers.

This new product strategy had immediate repercussions for Greenwood Mills' information systems. With two finishing plants closed, there was over-ca-

capacity on the company's IBM 3081 mainframe. Data processing management recommended downsizing to the IBM AS/400 mid-range system.

To smooth the transition from a mainframe COBOL environment, they also chose an AS/400 CASE system from Synon of Larkspur, California. Hill was convinced that only a complete applications development solution would allow his staff to tightly control application design, construction, and implementation for years to come. "Company-wide, we had to reduce administrative and plant costs," Hill says. "In the data processing department, the downsizing was our part."

Mastering the AS/400

After years in a mainframe environment, Greenwood Mills' users had come to expect the benefits of a powerful computer. The data processing department was faced with mastering its new AS/400 and delivering the advanced functionality to which everyone was accustomed. "Coming from an IMS DB/DC environment, we were used to sophisticated systems," Hill explains. "We had to deliver at least as much function—and maybe more if possible." All of this required an innovative ap-

proach to application development and maintenance.

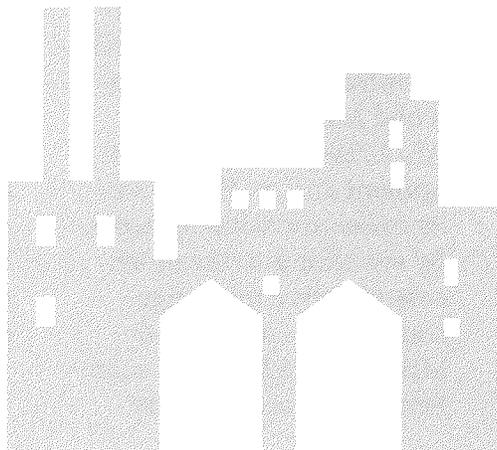
Having used a COBOL generator in the mainframe environment, Hill and the data processing staff were confident that a CASE tool would be the most efficient way to approach development. However, the department faced several challenges. The most significant was language. "I had 13 COBOL programmers, and none who knew the AS/400 RPG programming language," Hill recalls. "We were looking for a productivity improvement product and something that generated COBOL." While they briefly considered several other CASE tools, they quickly decided on Synon and its COBOL code generator. "We knew that Synon was the leader in AS/400 CASE," Hill says.

Greenwood Mills purchased Synon in August 1989. It was the beginning of a three-year adventure, because while Hill's programmers knew COBOL, they "had zero experience" with the AS/400. "That was fun," Hill says with a laugh.

Given the enormity of the task—5,000 programs had to be replaced—Hill and his managers decided to purchase financial and human resources packages from Software 2000

of Hyannis, Massachusetts. A three-person team was assigned the responsibility of implementing these systems on the AS/400. However, 60% of Greenwood Mills' mainframe systems had been custom-built in-house and were leading-edge, mission-critical textile manufacturing and distribution applications. These included raw material inventory, production reporting (tied to payroll and efficiency reports), and customer support.

Over the years, Greenwood Mills had differentiated itself from the competition by developing very strong, on-line customer service systems. These accounted for a third of the mainframe systems, and included saleable inventory (orders, billings/shipment), customer profiles, credit management tied to accounts receivable, and sales analysis. Re-creating these mission-critical systems on the AS/400 was the assignment for the Synon development team.



Learning new skills

The first order of business was learning about the new AS/400. Hardware and operating system training was completed by late 1989. Next, in January 1990, a core team of six Synon developers completed an intensive three-day, hands-on Synon training program. It was custom-designed for Greenwood Mills and delivered on-site by Synon specialist MSI of Memphis, Tennessee. For the next four months, the Synon team was busy absorbing the new operating system and an entirely new approach to systems development.

Hill's team was accustomed to the interactive database on the IBM mainframe, but the relational database built into the AS/400 operating system was implemented differently. Their traditional application development methodology and manual coding in COBOL was totally superseded by Synon's methodology. With Synon, the coding is done automatically by the generator. Instead of writing code line by line, the developer focuses on modeling the data and diagramming the actions that define an application. Their task was even more challenging

(continued on next page)

Textile Manufacturer ...

(continued from previous page)

because they were essentially retrofitting the mainframe systems to a new platform.

By October 1990, the team went live with its first major Synon-built system: a cotton inventory management program was put on-line in all of the weaving plants. "The users were very pleased," Hill commented. "They got all the functionality they had before—and more. Also, through attrition, we were able to reduce the cotton department by four or five people."

By the end of September 1992, Hill's Synon team had rebuilt all of the mainframe applications for the AS/400. They had estimated that the job would take two and a half years—and that estimate was just about right. "It took just two months more," Hill says. "Our group has a lot to be proud of." In less than three years, they had built 25 mission-critical systems, including:

- ☑ a system to track raw material and saleable inventory,
- ☑ a manufacturing production system which incorporates efficiency reporting, production-unit reporting for feed-

ing payrolls, and manufacturing management billing,

- ☑ a customer support system which includes orders, shipping, and billing,
- ☑ a customer information system which includes customer profiles, credit management, and accounts receivable,
- ☑ sales analysis tools,
- ☑ a cost system for developing standard product costs for pricing and inventory valuation.

After years in a mainframe environment, Greenwood Mills users had come to expect the benefits of a powerful computer. The data processing department was faced with mastering its new AS/400 and delivering the advanced functionality to which everyone was accustomed.

Unplugging the mainframe

On October 2, 1992, the entire data processing department "unplugged" the mainframe. Today, Greenwood Mills has two AS/400s—a B60 for development and an E80 to run its mission-critical systems. By switching to the more cost-effective midrange computers, Greenwood Mills saves \$250,000 a year. Personnel costs, too, are

down. Fewer than 30 people manage all of Greenwood Mills' information needs.

Looking back, Hill credits the Synon CASE tool with much of the success Greenwood Mills had in cutting over to the AS/400 system. But, more importantly, Synon forced the company to employ efficient system development techniques. This resulted in crucial savings for the data processing department, and the company. "It is a real productivity tool," Hill

says. "I think everybody on staff would say, if we hadn't had Synon, we wouldn't have made it—and that includes those who had to wrestle with it."

Greenwood Mills' top management has also been happy with the results. The new system has made the company more

productive than ever. "We are a much stronger, more competitive company," Hill says. "Greenwood Mills will be one of the survivors in the U.S. textile industry."

Synon is a vendor of full life cycle application development systems for the AS/400 and other platforms. They are located in Larkspur, CA and can be reached at 415-461-5000.

Report from SOFTWARE WORLD...

(continued from front page)

Larry DeBoever of DeBoever Architectures on the convergence of re-engineering and downsizing, and Ed Yourdon, of *American Programmer* on the silver bullets of software engineering and development.

Larry DeBoever—the convergence of re-engineering and downsizing

DeBoever, also co-chair of DCI's DOWNSIZING EXPO, brought new ideas and a new presentation to SOFTWARE WORLD. He started his session by discussing various companies that are shooting their mainframes. Amoco, for example, is in the middle of a program to move all of its information systems to NetWare-based LANs with UNIX. The company is half way to the goal of dumping all mainframes; the implementation of the plan is to be completed by 1995.

Another example of aggressive downsizing is United Air Lines. Through its Covia data processing subsidiary, UAL has been a leader in the movement to new computer-based tech-

nologies. The company has a plan to displace all of its mainframes. At the current time, their policy is to move all existing applications off the mainframes and onto UNIX servers and LANs—all applications except for passenger reservations which is run from a version of PARS on IBM mainframes. In about two years, Covia plans to shoot the mainframe for passenger reservations, also.

DeBoever went on to say that a few years ago he was

DeBoever stays on top of the hottest new topics by gauging the tone of his incoming phone calls. For the last year, the topics of business and IS function re-engineering have been very hot ...

a mainframe bigot. He then commented, "Hey, I was wrong!" I remember a point in time about three years ago when DeBoever was saying that "UNIX is for sissies!" And, of course, he is now a consultant to USL (UNIX Systems Laboratories, just bought by Novell). For anyone who might complain about his wrong projections, DeBoever offered the analogy that Tom Peters in his landmark book *In Search of Excellence*, talked about the market domi-

nance of companies (such as IBM and DEC) that focused on their customers. DeBoever also pointed out that Peters will now speak to your group for \$50,000 on the topic of why he was wrong. DeBoever added that he is willing to admit to your group that he was wrong for only \$500. (DCI acts as a speaker's bureau for DeBoever, give us a call if you're interested.)

Continuing, DeBoever then gave his analysis of what went wrong at some of this industry's leading companies (such as IBM and DEC). His theory is that they focused too much on their customers—who then focused back on them. The result was that both the vendor and the customer ended up just doing the same old thing.

In the meantime, smaller, more aggressive competitors such as Microsoft developed new technologies which were sold to new customers. By the time the "old" customers and "old" technology leaders noticed, there were newer and better ideas in the marketplace, and it was already too late to catch up because the new "kids" had built up insurmountable leads. An obvious conclusion to

(continued on next page)

Report from SOFTWARE WORLD...

(continued from previous page)

this theory is that firms need quickly adaptive product strategies, or their lunches will be eaten by new competitors with better approaches. If your current product strategy isn't winning (the story with OS/2 in 1991) then try something else and abandon the inferior approach (as Microsoft did with OS/2).

DeBoever stays on top of the hottest new topics by gauging the tone of his incoming phone calls. For the last year, the topics of business and IS function re-engineering have been very hot. His incoming callers say that new hires, MBA types, and company presidents continue to request approaches that can change the fundamental ways companies do business.

A related type of phone call that he has been receiving is from CEO's who want an evaluation of the job their CIO is doing. This statement supported a point made earlier in the day by Ed Yourdon (see page 14) who reported that the average CIO tenure is getting shorter—it is currently at 2½ years.

The incredible shrinking business cycle

One of DeBoever's main points was that business cycles are shrinking. A confluence of technology, competition, and economics is causing business reaction times to shorten. Where basic business change cycles used to take seven-ten years, the current business cycle is more likely to last only 18-24 months. Because of this, the basic approach to business is changing. Companies that are the

... Cutting through all of the PC (politically correct) jargon, DeBoever said that business re-engineering is a code word for getting rid of jobs—the jobs typically held by business professionals...

most successful are those that can most quickly adjust to changes in business strategies and conditions. DeBoever's name for the enablers to this process is *adaptive systems*.

So, on a global basis, company margins are shrinking and reaction times are getting shorter. DeBoever advised that you shouldn't worry about the competitor down the block, but rather the competitor from Malaysia that you haven't heard about yet.

One of the answers is for companies to re-analyze their industry. The leading US airlines, Delta, American and United, just happen to be the same companies that control the computer-based reservations systems. This is no coincidence—the computer-based reservation systems have proven to be more profitable than the business of flying planes.

Business re-engineering equals staff downsizing

Cutting through all of the PC (politically correct) jargon, DeBoever said that business re-engineering is a code word for getting rid of jobs—the jobs typically held by business professionals. IS professionals have a choice here: they can aid in re-engineering business processes with the result of eliminating line business jobs, or if they don't cooperate, chances are that the IS function will be outsourced, and *they* will lose their jobs. This is another example of the fact that downsizing in the computer sense brings with it downsizing in the employment sense.

DeBoever spoke about his favorite examples of inspired uses of new computer ideas to significantly

improve operations and save costs. For example, Northwest Airlines uses expert systems to audit all travel agent ticket accounting when more than one airline is involved in a routing. The result of this program has been the recovery of lost revenue to the tune of \$1 million per week. In addition, this new program has not generated a need for new employees.

In the insurance industry, some companies have moved claims adjusters onto the road with the goal of getting them to the site of accidents before the principals have a chance to leave. By surveying the situation first hand, the adjuster is able to offer a check on the spot to settle the damage claim. If information systems can provide the necessary data to assure the coverage, the process of settling a claim in real time can typically save \$1,000 in administrative costs. Accordingly the adjuster can offer the insured an immediate settlement of \$200 over the estimated loss (to cover uncertainties and possible variances) with no follow-up hassles. The result is a happier customer and reduced costs for the insurer.

Down-scaling vs. re-engineering

DeBoever talked about two fundamentally different

approaches to downsizing. The first he called down-scaling. Down-scaling is about figuring out how to use new technologies to reduce IS costs—in other words, go for the cheapest approach. The other approach is re-engineering. Re-engineering doesn't focus on IS costs, but rather on the overall cost and responsiveness of the entire business process. A typical re-engineering project results in significantly reduced business department costs because fewer line workers are needed. However, the typical re-engineering project also results in increased IS costs.

DeBoever has seen cases where companies started projects with the idea of down-scaling, and then, at some point in the cycle, shifted their strategy towards re-engineering. DeBoever feels strongly that

re-engineering approaches have more potential to positively impact a business' overall than does down-scaling.

A recommended solution is for the IS staff to research and understand the business from an information systems point of view. Then the company should rebuild the information architecture to support that re-engineering. The goal of all this effort is to have adaptive systems that the company can use to react quickly to changed business conditions. In this game, the company with the most adaptive systems wins.

DeBoever's rules for building these more adaptive systems are as follows:

- Make the systems highly granular and loosely coupled. Insure that

(continued on next page)



Shooting the Mainframe

Report from SOFTWARE WORLD...

(continued from previous page)

applications do as little as possible; there should be few function points in each application.

- * Build a message interface (rather than requiring real time) between each of the applications.
- * Localize as much processing as is possible.
- * Maximize bandwidth.
- * All systems must adhere to a client/server model.
- * Install an enterprise-wide backbone image communications net. The LAN model (which is flat) is the right model—anyone can get to anything.

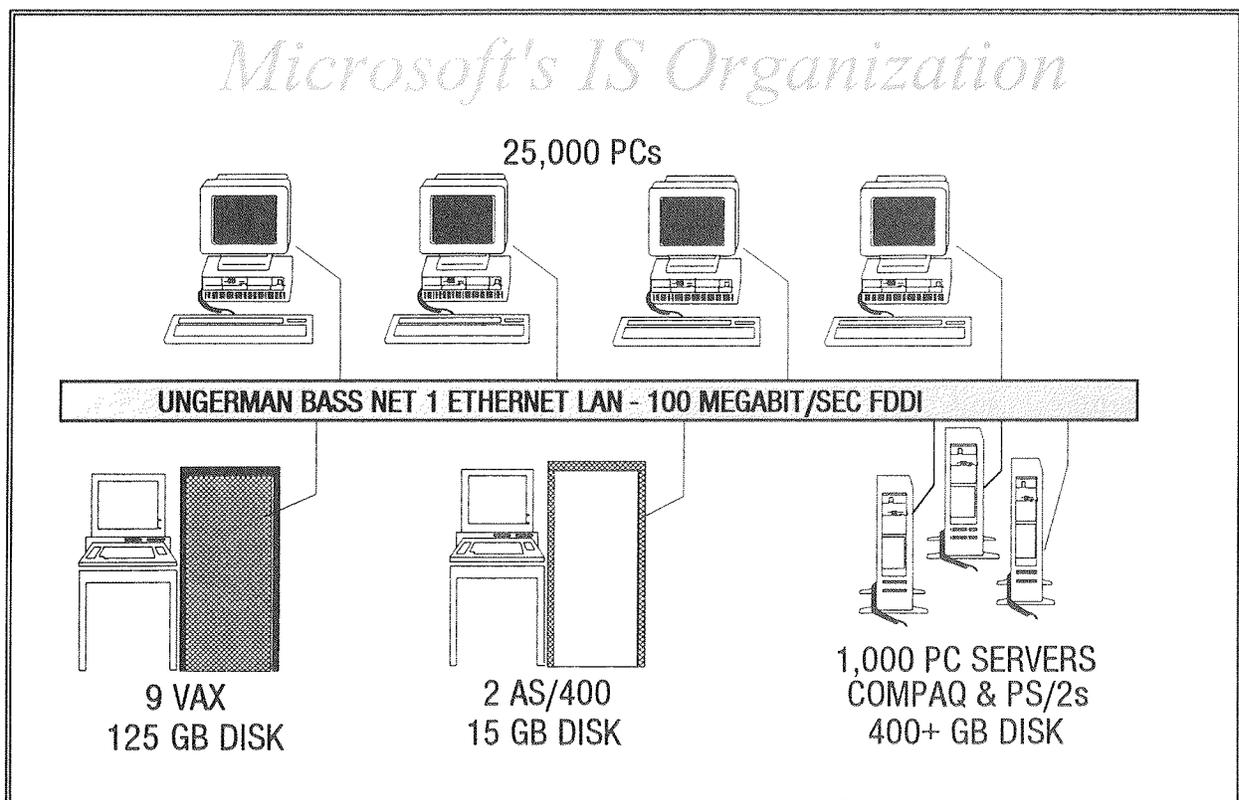
- * Applications must be event driven; batch is dead. Event driven message-based is definitely the way to go.
- * Use RAD to quickly produce new systems. Speed is more important than methodology.
- * The interface should be the most important part of all applications. RAD gets feedback from the user faster.
- * Use business process approaches, rather than data driven approaches, to be the most successful in this new world.

Are the radicals right?

DeBoever also spoke about the political aspects

of downsizing. "Think of this as a cultural issue," he said. On one side you have "the kids" who carry around stacks of *PC Week's* full of Post-It notes, dress like they just auditioned for *Melrose Place*, and sport great hair cuts. This group thinks you can replace a 3090/600 with five PCs running NetWare over twisted pair wire. He called this group the radical left and said that their typical goal is to shoot the mainframe.

On the other side are the nervous types with eye twitches. This group is losing its hair and graduated from school before there was any such thing as computer science. This group is



the right wing and their agenda is typically to shoot everyone in the left wing.

The real answer here is not with either of these groups. What you want to do is shoot the mainframe but keep a data center. *(Editor's Note: Internally, Microsoft should be the paragon of new generation IS technology and management. I've looked at the internal Microsoft IS operation and it surprisingly resembles a traditional shop. The vast majority of the company's processing is done against a client/server architecture with the servers located in a secure, fire controlled environment—a glass house. The main difference from most other \$4 billion organizations is that most of Microsoft's data is located on IBM and Compaq PC servers. See figure on page 12.)*

The correct way for most organizations to approach downsizing, then, is to keep the rigor and discipline of the mainframe world, but use the new generation of downsizing technologies.

UNIX has come a long way in the region of data center utilities, security, and rigor. It is now a real mainframe replacement alternative. SUN has just announced 500 MB/second I/O channels. And if you're not satisfied with this, you can order dozens of them on SUN RISC systems. Note these numbers in compari-

son to IBM channels rated at 4.5 MB/second (unless you use fiber in which case it's 10 MB/second).

The fact that software tools and DBMSs have acquired many of the features necessary for operating robust applications, has enabled downsizing to progress as far as it has. DeBoever believes that a principal reason for this is that the leading downsizing vendor companies have brought in new top management with mainframe backgrounds. People like Roel Pieper, Dave Litwack, John Landry, Mike Maples—the leaders of companies such as USL, Powersoft, Lotus, and Microsoft—are ex-mainframe people who know what is necessary in software if you want to replace "real" data processing systems. SUN's UNIX systems programming staff is almost entirely composed of ex-MVS systems programmers. DeBoever believes that Microsoft's Windows NT will be a success exactly because it wasn't built by Microsoft's PC people; the project's leader is Dave Cutler who's previous job was developing VMS for DEC.

A quick fix for the ailing mainframe

For those of you who end up under the gun to quickly reduce the costs of the corporate mainframe,

DeBoever offered a number of immediate steps that can be taken to quickly cut back on mainframe use and cost:

1. Pull all monitoring tools off your mainframes. This buys you an immediate 15% performance gain.
2. Look for packaged applications such as payroll that can be installed on a client/server network and will result in applications that can be off loaded from the mainframe.
3. Look to move any independently running CICS or COBOL applications. There are compatible packages that allow you to run these applications with almost no conversion effort on either PC networks or UNIX boxes.
4. Check your DB2 applications. With some conversion effort and a little time, these can be moved onto relational DBMSs in a UNIX, or even PC, environment. Any database conversion like this, however, will take some time and isn't quite as fast as the previous ideas.
5. If you're running FOCUS applications, they can be quickly and painlessly moved to PC or UNIX environments since FOCUS runs just about anywhere.

(continued on next page)

Report from SOFTWARE WORLD...

(continued from previous page)

Even if you're only able to implement some of the above suggestions, you're likely to see an immediate reduction in usage of up to 30–40%. If most of the costs associated with the mainframe are fixed (like lease costs) then the remaining users are likely to see their hourly time charges go up by this same 30–40%. As these fixed costs continue to get spread over a shrinking base, your managers will quickly figure out the nature of this game—last one off loses! That's good in one way, but you'll need good management and accounting skills to navigate through these situations.

Ed Yourdon—the silver bullets of software engineering and development

Ed Yourdon started his keynote by listing the top ten software development goals as determined by a survey of project and software development managers at major corporations. They were as follows:

1. Provide rapid responses to user requests.
2. Increase development team productivity.
3. Identify strategic systems for development.

4. Develop an overall information architecture.
5. Perform effective maintenance and manage system obsolescence.
6. Win top management support.
7. Provide quality management.
8. Support cross functional areas.
9. Develop metrics for systems development.
10. Effectively manage the systems development done by end-users.

Yourdon has been following these surveys for the past five years. He has determined that in terms of longevity, the top five issues over the past half-decade have been:

- ☒ The need to align IS and corporate goals.
- ☒ The re-engineering of business practices.
- ☒ Information architecture creation.
- ☒ Better use of data.
- ☒ IS human resource improvements.

Yourdon then stated that software development is still a crisis situation. To provide support for this statement, he quoted from productivity statistics compiled by Capers Jones. (Jones will be speaking on these topics in Chicago, October 26–17, 1993. For more details, call DCI.) Some of these startling statistics indicate that:

- ☒ The average programmer productivity is 10–15 statements/day.
- ☒ The average programmer produces five function points/staff month, but with a huge variance depending on the individual and shop.
- ☒ Programmer productivity is improving at rate of 1–2% per year.
- ☒ Less than 10% of projects finish on time and within budget.
- ☒ Between 25–30% of large projects are never finished.
- ☒ Some bugs are actually classified as *not fixable*. Yourdon talked about a large PC software company that he didn't want to identify by name (it was code-named *Mightsoft*) that actually has a software bug category called "*will not fix*."

A number of solutions to the software development crisis have emerged. The "silver bullets" of these solutions were identified as:

- ⇒ The use of better programming languages (this is frequently proposed by technical staff). However, this suggestion is not helpful in maintaining legacy systems. Is tossing out the old really practical? What about retraining the existing staff? His answer was the same—toss them out. The problem is that better code may help you arrive at a disaster sooner

- than before. Better languages will become irrelevant as CASE tools improve in quality over code generators.
- ⇒ Better training and/or staff usually has a higher payoff than improved tools. Another important issue here is the need to develop a technique for hiring better quality people in the first place.
 - ⇒ The approach of improving the software process is based on the work done at the Software Engineering Institute. This research has been documented by Watts Humphrey in the book *Managing the Software Process*. The essence of this approach is to develop a manageable, engineering process for software development.
 - ⇒ Developers should use automated CASE tools. This has typically been very expensive at the cost of \$30,000–\$60,000 per engineer for hardware, software, and training. However, there are some new tools priced at the \$1,000/engineer point that actually can perform 80% of the expensive tools' jobs. However, the introduction of CASE tools is usually accompanied by a productivity decrease of three to six months. And, if you think that's bad, try moving to object-oriented techniques which typically decrease productivity on average for 18 months.
 - ⇒ JAD (Joint Application Development), and RAD (Rapid Application Development) approaches, which use some form of prototyping, are fine but don't eliminate the need for formal analysis and design on large projects. CASE tools should be partially evaluated on how well they support rapid turn-around approaches.
 - ⇒ The structured approaches popular with older developers (still the most popular methodology in North America) are no longer relevant. *(Editor's note: Mind you, this comment is from one of the original developers of structured methodologies!)* Good CASE tools in this area need to support data flow, entity relationships, and state transition types of analysis.
 - ⇒ Information engineering, which is often referred to as the Soviet Union approach to building software, is best suited for enterprise-wide approaches. However, information engineering is becoming less relevant in many situations. A good question is "Can information engineering adjust to object-oriented approaches?" *(Editor's note: James Martin Associates and IntelliCorp have been collaborating on just such a product approach.)*
 - ⇒ The problem with object-oriented approaches is that most of the attention is being devoted to OOP (object-oriented programming), and not OOA (object-oriented analysis) or OOD (object-oriented design). If you're arguing about whether you should use Smalltalk or C++, you're in trouble. CASE vendors are mesmerized by *object-oriented*; some are moving towards it, some are already using object-oriented technology, and others are still hoping it will go away.
 - ⇒ Employing software reusability is an old idea that is only rarely successful. It is typically hindered by the fact that software metrics are inadequate. But the highest productivity is attained by organizations that master this approach either with or without object-oriented approaches. This approach is popular in Manila and Japan.
- Yourdon concluded with a discussion of the question "Is it too late to start on the road to improved software engineering?" The answer is, of course, no. *JS*

UPCOMING downsizing Events...



DOWNSIZING EXPO is teaming up with a new exposition, **OP/EN EXPO**, *The Open Operating Systems and Enterprise Networks Conference and Exposition*, for two fall shows, August 3-5, 1993 in Santa Clara and September 13-15, 1993 in Toronto. In addition to the various topics to be covered at **DOWNSIZING EXPO**, including *Downsizing Technologies and Architectures, Client/Server Computing, Managing the Downsizing Process, Life After Downsizing, Business Re-Engineering*, and *Enterprise Servers & Midrange Computing*, **OP/EN EXPO** will add *Open Operating Systems, Interoperability, Enterprise Networks*, and *Systems Integration*. Co-Chairmen George Schussel and Larry DeBoever will be presiding over each three-day, joint event. Other featured speakers will include, Dave Andrews, Roger Burlton, Cheryl Currid, John Dunkle, Richard Finkelstein, Ron Peri, and Jeff Tash.

This September, there are two back-to-back seminars being held in Philadelphia: *Implementing Client/Server Applications and Distributing Data* with Herbert Edelstein, September 28-29, and *Finkelstein's Practical Guide to Client/Server DBMS Computing* with Richard Finkelstein. Edelstein's seminar shows attendees how to use client/server technology to effectively distribute data throughout their organization. Finkelstein will cover the differences between cooperative processing and client/server, as well as open systems, network considerations, relational DBMSs, and interoperability. In addition, DCI is offering several one and two-day downsizing seminars with such industry notables as Larry DeBoever, Cheryl Currid, Jeff Tash, and George Schussel.

For more information on any of these classes or conferences, call DCI at (508) 470-3880.

<p><i>Schussel's</i></p> <p>DOWNSIZING JOURNAL</p> <p>July 1993</p> <p>Volume 2 Issue 11</p>	<p>Editor: Dr. George Schussel</p> <p>Managing Editor: Stacey S. Griffin</p> <p>Subscription rates:</p> <ul style="list-style-type: none"> ● \$199 annually for U.S. residents ● US\$225 annually elsewhere 	<p>SDJ is published monthly by:</p> <p>Digital Consulting, Inc.</p> <p>204 Andover Street</p> <p>Andover, MA 01810 USA</p> <p>508-470-3870 FAX 508-470-1992</p>
--	--	--

Copyright © by *Schussel's Downsizing Journal*. This newsletter is fully protected by copyright. Reproduction or photocopying, even for internal use, without the publisher's permission is PROHIBITED BY LAW. Violators risk criminal penalties up to \$100,000 in damages. For permission to photocopy for internal use or to inquire about special reprints, contact Stacey Griffin at (508) 470-3870. *Discounted multiple-copy subscriptions are available.*

<h2 style="margin: 0;"><i>Schussel's Downsizing Journal</i></h2> <p><input type="checkbox"/> <i>I want to subscribe. Please send me the next issue.</i></p> <p><input type="checkbox"/> <i>Please RENEW my subscription.</i></p> <p><input type="checkbox"/> Check payable to Digital Consulting, Inc. is enclosed</p> <p><input type="checkbox"/> Bill me</p> <p><input type="checkbox"/> MC # _____ Exp. _____</p> <p><input type="checkbox"/> Visa # _____ Exp. _____</p> <p>Subscription Rates</p> <ul style="list-style-type: none"> ● \$199 for one year in continental US, US\$225 elsewhere ● All amounts payable in U.S. dollars please <p>To Subscribe: <input checked="" type="checkbox"/> Drop a copy of this form in the mail addressed to: 204 Andover Street, Andover MA 01810 USA</p> <p style="text-align: center;">☎ Phone (508) 470-3870 or 📠 FAX a copy of this form to (508) 470-1992</p>	<p style="color: #808080;"><i>Subscription Form</i></p> <p>_____</p> <p>Name</p> <p>_____</p> <p>Title</p> <p>_____</p> <p>Company</p> <p>_____</p> <p>Address</p> <p>_____</p> <p>_____</p> <p>Telephone</p>
--	---