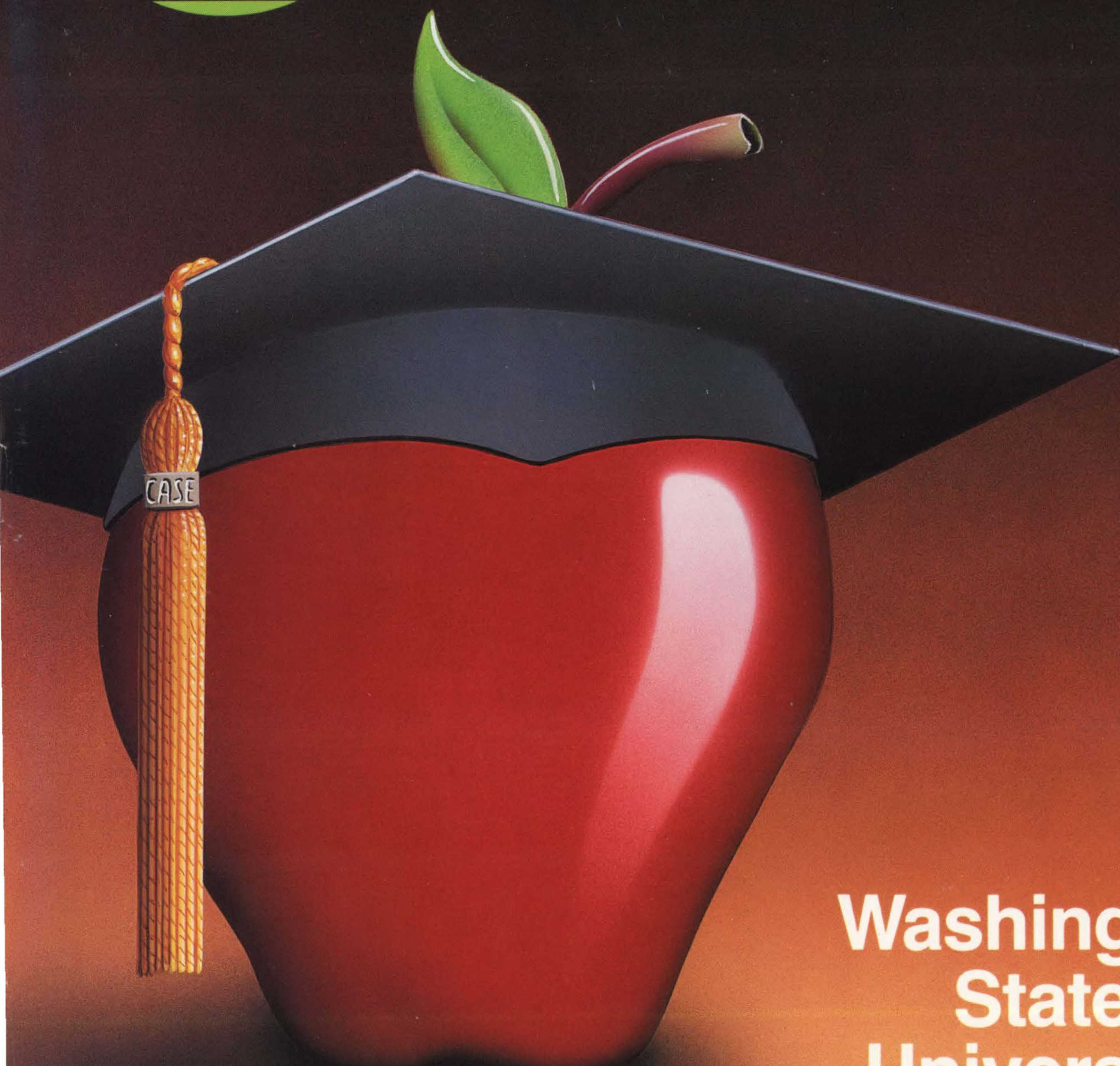
MAINFRAME JOURNAL

For Users of IBM System/370 Architecture & Compatible Systems

June 1989



Washington State University Adopts CASE Strategy

SQL/DS AND DB2 USERS:

It Takes the Right Tools to Create a Masterpiece

Like all great works of art, creating an efficient database requires the right tools. Using Systems Center's full line of database management tools, you can turn your database into a masterpiece.

With the creative edge that Systems Center database products provide, you can improve the performance of your database, increase the productivity of your programmers, and protect the integrity of your data.

- **DB/AUDITOR** for **DB2** provides an easy to use audit reporting system for all DB2 activity. History, summary, and detail reports are available to satisfy your DB2 auditing requirements.
- **DB/REORGANIZER** for **SQL/DS** (formerly VMSQL/REORG) allows the Database Administrator to improve the performance of SQL/DS by quickly and easily altering and reorganizing database objects.
- **DB/REPORTER** (formerly VMSQL/REPORT) for both SQL/DS and DB2 allows programmers to write reports in a fraction of the time required of conventional programming languages.
- **DB/OPTIMIZER** for **DB2** (formerly DB/OPTIMIZE) reports on DB2 access path selection problems, identifies potential problem areas, and recommends possible solutions.
- **DB/EDITOR** for **SQL/DS** (formerly VMSQL/EDIT) makes it easy to maintain SQL/DS tables and to build database entry applications.
- **DB/SECURE** for **DB2** provides a comprehensive means of controlling access to DB2 systems, resources, and data.

With Systems Center's database products, you can quickly master the art of database management. Call Systems Center today to get the tools you need to create a masterpiece!

(800) 359-5559
(703) 264-8000

SYSTEMS CENTER
FORMERLY VM SOFTWARE, INC.

NETWORK ADMINISTRATION PRODUCTS ■ RELATIONAL DATABASE PRODUCTS
VM SOFTWARE PRODUCTS ■ NETWORK DATAMOVER PRODUCTS

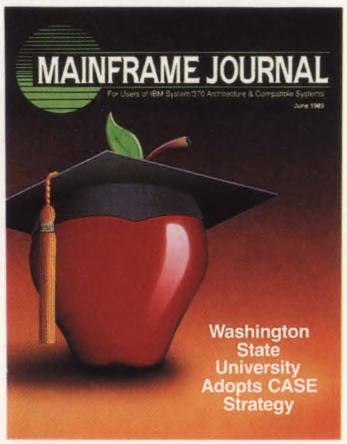
1800 Alexander Bell Drive, Reston, Virginia 22091

CIRCLE #20 on Reader Service Card ▲

6-MFJ-8906

C O N T E N T S

DEPARTMENTS



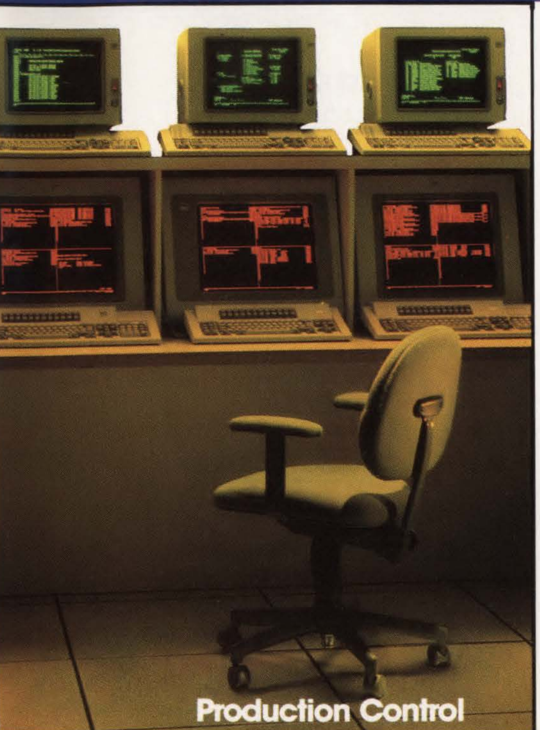
COVER:

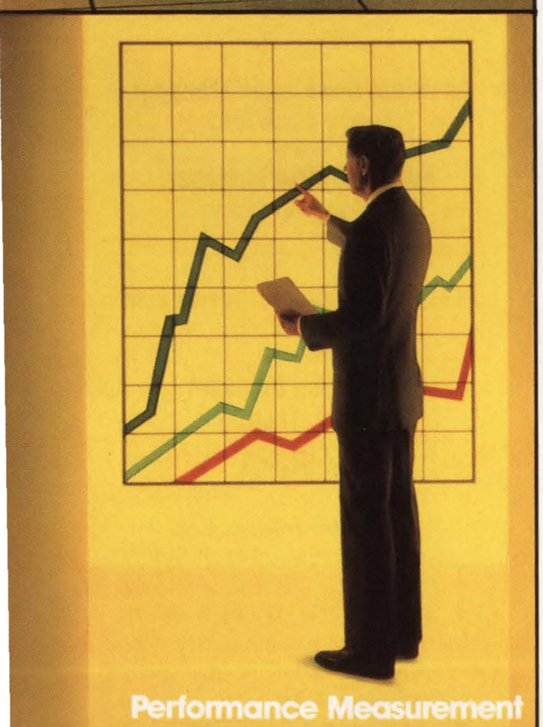
Computer Aided Software Engineering (CASE) technology has the potential to meet the increasing demand for more sophisticated software from today's businesses that require shared, integrated information systems to handle critical functions. For current information about CASE, turn to the CASE articles on pages 12 and 61 and Viewpoint, page 98. Cover illustration by David Kramer.

MAINFRAME JOURNAL© (ISSN 0895-5751) is published monthly by Thomas Publications, Inc., 10935 Estate Lane, Suite 375, Dallas, TX 75238, (214) 343-3717. Second class postage paid at Dallas, TX. SUBSCRIPTION RATE: Subscriptions are free within the USA and Canada. One-year foreign subscriptions are $96. POSTMASTER: Send address changes to: *MAINFRAME JOURNAL*, P.O. Box 551628, Dallas, TX 75355-1628.

F E A T U R E S

Computer Associates presents total data center automation for VSE.





Computer Associates brings you the industry's only complete line of VSE systems software: **CA-UNICENTER®/II VSE**, the total solution for data center automation.

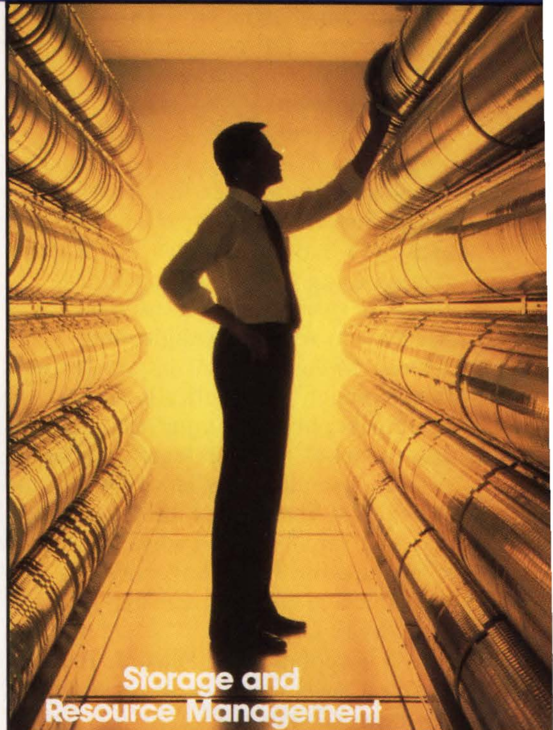
With **CA-UNICENTER/II VSE**, you gain extensive tape and disk management capabilities, automated production scheduling and output distribution, performance measurement and job accounting facilities, the industry's leading security software, essential sorting and report writing utilities and even more.

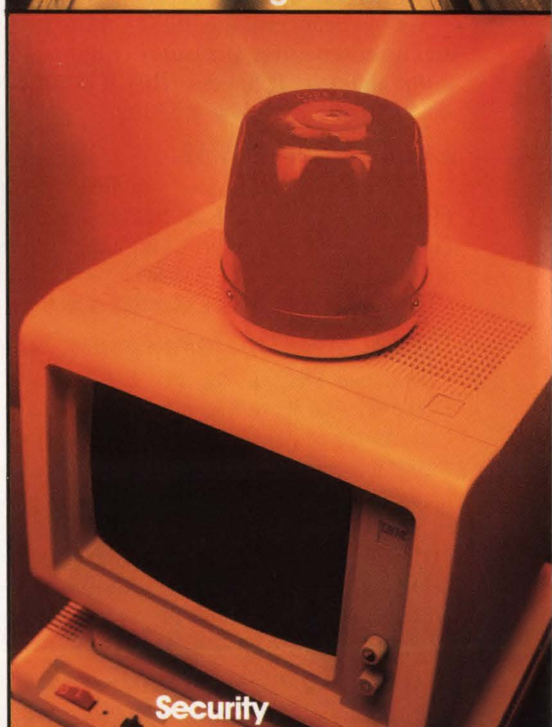
Through advanced integration and automation, **CA-UNICENTER/II VSE** ensures efficient utilization of data center resources, virtually eliminates manual involvement and dramatically improves overall productivity of the data center—enabling you to achieve levels of efficiency never before possible.

Automated service and support, too: **CA-UNISERVICE®/II** provides a secure link between your mainframe and Computer Associates' Customer Service System, 24 hours a day. You get online access to software fixes, interactive problem resolution, product tutorials and more.

Call Dana Williams today:
800-645-3003

711 Stewart Ave., Garden City, N.Y. 11530-4787





COMPUTER ASSOCIATES®
Software superior by design.

- World's leading independent software company.
- Broad range of integrated business and data processing software for mainframe, mid-range and micro computers.
- Worldwide service and support network of more than 100 offices.

Resource & Operations Management • Financial • Banking • Graphics • Spreadsheets • Project Management

CIRCLE #105 on Reader Service Card ▲

Bob Thomas

"Everyone talks about the weather but no one does anything about it." This old adage has a contemporary DP counterpart. In this instance, CASE is the operative word (make that the operative acronym).

Seminars and magazine articles on CASE are running amok in DP land and no end is in sight. Many organizations are in various early stages of evaluating and implementing CASE but relatively few have full-blown applications in production using CASE methodologies.

In this issue we present several CASE-related articles you will find enlightening.

Dean Mohlstrom, in his article titled "CASE and The Balkanization of System Development," emphatically states, "To realize the full potential of CASE, it should be utilized only after your data house is in order with a stable, complete and concise data model of your strategic information plan and corporate data architecture."

When should an organization start using CASE? Pete Privateer, in his article titled "CASE: A Technology In Transition," puts it rather bluntly. "Those organizations that wait for CASE to evolve will find themselves far behind their competitors."

Washington State University has adopted a particular CASE strategy to meet the academic and business demands of a major university. WSU's experiences are profiled by T. Marc Graham.

In Viewpoint, Kent Petzold, President of Viasoft, Inc., rounds out our rather extensive CASE coverage by answering the question, "Reengineering: Buzzword Or Business Advantage?"

As we head into the 1990s, more and more companies will be talking about CASE and as competitive pressures continue to mount, these companies will be doing something about it. It appears that it is time to get busy doing something about CASE — let Willard handle the weather!

Double Shot Of VM

VM users, you lucked out this month. Two excellent VM-oriented articles are in this issue. First, Michael Seadle describes how a development center can take advantage of VM's procedural language, interactive environment and electronic mail in his article, "VM In The Development Center." In addition, Steve Eckols teaches you "How To Use VM's Essential XEDIT Commands" in his article of the same name.

IMS & SQL/DS Take Center Stage

Because DB2 is the hot DBMS, we typically devote quite a bit of space to it. However, we have had a number of requests for articles on IMS and SQL/DS. In this issue those requests are honored by Gary Rygmyr in his article, "IMS Fast Path Data Entry Database Tuning," and by Messrs. Bernik, Herbst, Hendrickson and Ramirez in their article, "SQL/DS In Full Production."

The August *MAINFRAME JOURNAL* will be a special DB2-emphasis issue in honor of the first International DB2 Users Group Conference to be held August 6 through 9 in Chicago, IL. If you would like to attend the conference, contact Jack Thompson, Director of Administration and Conferences for the International DB2 Users Group (IDUG) at (312) 644-6610.

Are We On Target?

Is the editorial coverage of *MAINFRAME JOURNAL* on target from your perspective? Have we hit your "hot button" yet? If we have, great; let us know so we can continue in that direction. If not, sorry; let us know and stand by! Either way, please communicate with us by using the Reader Service Cards between pages 74 and 75. It's your magazine and your enthusiastic interest and support have made *MAINFRAME JOURNAL* the most-read magazine among DP professionals using IBM mainframes. Please take a minute to help us stay on target.

Bob Thomas

MAINFRAME JOURNAL
For Users of IBM System 370 Architecture & Compatible Systems

Publisher and Editor-in-Chief
Bob Thomas

Associate Publisher
Martha Thomas

Editor
Carol M. Hoag

Copy Editors
Judy-Beller
Pat Warner

Art Director
David Kramer

Assistant Art Director
Ken Buerer

Magazine Production
Diane Dishman

Marketing Services
Sally Webb

Circulation Manager
Janice Porter

Assistant Circulation Manager
Nancy Crawford

Administrative Manager
Marian Davenport

Associate Publisher/Advertising
Mark A. Cauto

Advertising Manager
Denise Thomas

BPA

List Rental Information
Contact AllMedia (214) 985-4060

SUBSCRIPTION RATES & INQUIRIES: Subscriptions are free within the USA and Canada. One-year foreign subscriptions are $96.

All subscriptions, remittances, requests and changes of address should be sent to *MAINFRAME JOURNAL* at 10935 Estate Lane, Suite 375, Dallas, TX 75238, (214) 343-3717.

Lighten Up!

Five-Minute Systems Programmer

Move up to the power and prestige of controlling the heart of the mainframe. Become a systems programmer through this in-depth course. This rigorous course, five times as long as our management program, provides you with the technical and personal skills to qualify for these high-paying positions.

Course Topics

Buzzword Basics

Learn the proper use of jargon and acronyms to confound ignorant outsiders.

Four Easy Steps for Dispatching System Problems

1. It's not repeatable. Do it again — then come back.
2. Incomplete problem documentation. Get every conceivable — then come back.
3. It's probably a software problem. Wait 'til the next release is installed — then come back.
4. We've informed the vendor of the problem. As soon as we get the fix, we'll notify you.

Controlling Changes

Learn how to postpone, delay and defer petty maintenance and other needless changes while installing tools and other playthings. Master realistic capacity planning by learning to gather and present data justifying the object of your desire.

What's the Use of Users

Bothered by hundreds or thousands of users sharing your system? Discover their true value in creating unique debugging situations and providing test data for tuning games.

Profiting from Corporate Politics

Discover how to manage managers by upward delegation and other management techniques. Develop User Exits to reward your friends and deter your nemesis.

Interpersonal Techniques

Let our Arrogance Training show you how to act the part of a systems programmer. Our "Dress for Success" segment demonstrates how to appear as though you've been up all night, even after ten hours of sleep.

Just released on videotape is Poindexter's Memory Course offering:

- Practiced at highest levels of military command
- Demonstrated before US Senate
- Witnessed by millions on TV.

Courses that are available include "Maintaining Tape Archives" by Richard M. Nixon and "What You Can Do With PROFS Journaling" by Colonel Oliver North.

Michael Haupt, President
Haupt M.I.S. Consultants
Mt. Prospect, IL

VSAM Tip

I would like to expand on Eugene Hudders' comments *(VSAM Optimization and Design,* March 1989) about the use of the SPEED and RECOVERY options in defining VSAM datasets used in a CICS environment. Hudders points out that SPEED is the preferred option when defining a KSDS. When an ESDS is used in load mode as a data collection file, then SPEED becomes undesirable. In this special case, if SPEED is specified, then any data in the file is lost in the event of a CICS outage. RECOVERY should be specified for an ESDS which is written from CICS. I've seen people go to the trouble of converting an ESDS to a KSDS unnecessarily because they assumed the loss of data was simply characteristic of ESDSes, so I thought it might be worth mentioning.

Another frequently overlooked point regarding ESDSes written from CICS is that their performance is enhanced by the use of Local Shared Resources (LSR). Conceptually, a WRITE is strictly an output operation. For an ESDS in load mode, however, VSAM reads each CI, adds the new logical record to it and rewrites it. Consider an ESDS containing eight logical records per CI. With NonShared Resources (NSR), 16 I/Os are needed to fill each CI; one read and one rewrite per logical WRITE. With LSR, only nine I/Os are needed, since VSAM will get a lookaside hit on each read once the CI has been brought into storage (assuming, of course, that other VSAM activity in the address space does not flush the buffer in the meantime).

Dave Nicolette, CDP
Nicolette Consulting, Inc.
Coppell, TX

COBOL Efficiency

I found the article, *Why COBOL Efficiency Still Makes Sense* (February 1989), enlightening and informative. However, I question the technique used by Harvey Bookman to initialize a table (Figure 4) when there are several hundred table entries. I picked up a technique that I find both efficient and easy to code.

```
DATA DIVISION
WORKING-STORAGE SECTION
01  TABLE-OF-VALUES.
    02  INITIALIZE-TABLE.
        03  FILLER      PIC      S9(3)COMP-3 VALUE ZERO.
        03  TABLE-ENTRIES.
            05  TABLE-ENTRY OCCURS 3 TIMES PIC S9(3)COMP-3.
PROCEDURE DIVISION.
    MOVE INITIALIZE-TABLE TO TABLE-ENTRIES.
```

Mitchell T. Lawn
Senior Programmer Analyst
Fleming Companies, Inc.
King of Prussia, PA

ISPF

I enjoyed the article on ISPF Techniques by Jon E. Pearkins (February 1989). One area where I would like to make an additional comment in this article would be in the section titled, Moving Around Fast. Use of the =X function will use the ISPF main menu (Primary Option Menu) most of the time. It will not work if the ISPF function was a new application (indicated by the NEWAPPL keyword operand).

I know this fact since many times I will attempt to =X out of the IBM SMP/E dialog and will only get back to the SMP/E primary option menu. It will take another =X to exit ISPF.

Tom Bryant
First Options of Chicago
Chicago, IL

Editor's Note: *In Fred Schuff's article, "Optical Disk Technology," on page 12 of the May 1989 issue, the following sentence stated correctly should read, "The media is reliable (one error in 10 raised to the 12th power or 1,000,000,000,000 bits) and cost effective."*

IBM Commitment To Electronic Data Interchange (EDI) Grows

IBM recently announced a series of EDI "translator" software products that enable different business enterprises to electronically exchange standard-format business documents such as purchase orders, invoices and order acknowledgements. With the introduction of the IBM expEDITE DataInterchange Series, IBM now presents a comprehensive series of EDI offerings called the IBM expEDITE Family that is said to significantly decrease the time spent in manual processing and mailing of business documents and to help minimize costly bookkeeping errors.

Goal Systems Acquires Database Products And Goes Public

Goal Systems International, Inc. (Columbus, OH) is expected to complete the acquisition of the principal software products of DataBase Utility Group, Inc. (Seattle, WA). The products, INSIGHT/DB2 and INSIGHT/ADABAS, monitor the performance of IBM's DB2 and Software AG's ADABAS database management systems. This follows Goal System's recent acquisition of Bennett Software and the initial public offering of their stock on NASDAQ. Revenues for fiscal 1989 were in excess of $52 million.

VSE Data Center Automation Targeted by Altai & Smartech Alliance

"Only a small percentage of the world's 20,000 VSE mainframe sites have begun automating their data centers," said Altai President James P. Williams. With that sales potential in mind, Altai, Inc. (Arlington, TX) and Smartech Systems, Inc. (Dallas, TX) recently announced a strategic alliance focused on the development and marketing of comprehensive VSE data center automation software. "Both Altai and Smartech have focused efforts on the data center automation niche," added Smartech Systems President Eric L. Vaughan, "and by consolidating our efforts, we will be able to provide alternatives to the marketplace that might not otherwise have been possible."

The Computer Channel Provides Information Technology Programs By Satellite

The Computer Channel (CCI) is a year-old company in Floral Park, NY that provides information technology programs by satellite to clients nationwide and in Canada and Europe. CCI develops, produces, markets and broadcasts a full line of pay-per-view seminars featuring many top experts in the industry. CCI subscribers use small, KU-band satellite receiving systems (VSATS) and participate in the program at their offices. CCI customers include Aetna, IBM, Dow Chemical, Federal Express, Alcoa, U.S. Navy, Merck, John Hancock, Lockheed, AT&T and more than 75 other companies. For more information about The Computer Channel, contact Joe Rizzo at (516) 352-9490.

Westinghouse Introduces Token-Based Security Software System

Westinghouse Management Systems Software (Pittsburgh, PA) has introduced a computer security software system that provides authentication services for users of IBM mainframe-based networks across all operating systems. The new system is called Personal Authentication Security System (NC-PASS) and according to Mark Potenzone, Manager of Westinghouse MSS' North American Group, it provides a higher level of security by requiring the validation of three separate items — a userid, a user-changeable password and a physical device (token). The introduction of NC-PASS represents Westinghouse Management Systems Software's initial move into the token-based security market.

Sterling Software To Remarket SPACECALC

Sterling Software's Systems Software Marketing Division (Rancho Cordova, CA) will now market SPACECALC, a product developed by Spectrum Concepts, Inc. of New York City. The remarketing agreement was announced recently at Sterling's 10th Data Storage Management Conference held in San Antonio, TX. SPACECALC is an interactive DASD modeling facility that automatically performs storage allocation calculations for sequential, partitioned and VSAM datasets. Sterling Software markets a family of automatic volume allocation products called VAM. In announcing that it would remarket SPACECALC, Jim Hanchett of Sterling Software said, "The new product would be a 'natural fit' with its existing allocation products to provide a complete solution."

CASE And The Balkanization Of Systems Development

By Dean F. Mohlstrom

Most of the claims found in today's over-hyped CASE tool advertising and *in-depth* articles on systems design and applications development are true; and yet, the software industry continues to sell *products* instead of *service*.

The bad news is . . . by succumbing to this latest siren song of the software industry you may be doing your organization a gross disservice. Should CASE stand for "Computer Aided *Software* Engineering" or "Computer Aided *Systems* Engineering?"

Get Your Data House In Order . . . First!

CASE, when *improperly* introduced into an organization, can contribute to and accelerate the *balkanization* of the systems development effort. (Balkanization means *to break up into small, often hostile groups.)* To realize the full potential of CASE, it should be utilized only after *your data house is in order.* You should know where the systems you are developing fit within the corporate information system or you will continue to contribute to the problems created by past development methods and procedures (that is, redundancy, systems development backlog, high maintenance, inflexible systems and systems developed in isolation that will not talk to each other).

Help For The "Doers"

The organization that does not identify the vital strategic information, critical success factors and *data* needed to run its business with a data-oriented strategic information plan will not be around to compete in the future. I am not talking about the back-office accounting, payroll, inventory and other systems. I am talking about the information systems needed to support the front-office decision makers. The people in the *do* function of the organization, "The people that design, make, sell or service!" to quote John Akers, IBM's Chairman of the Board.

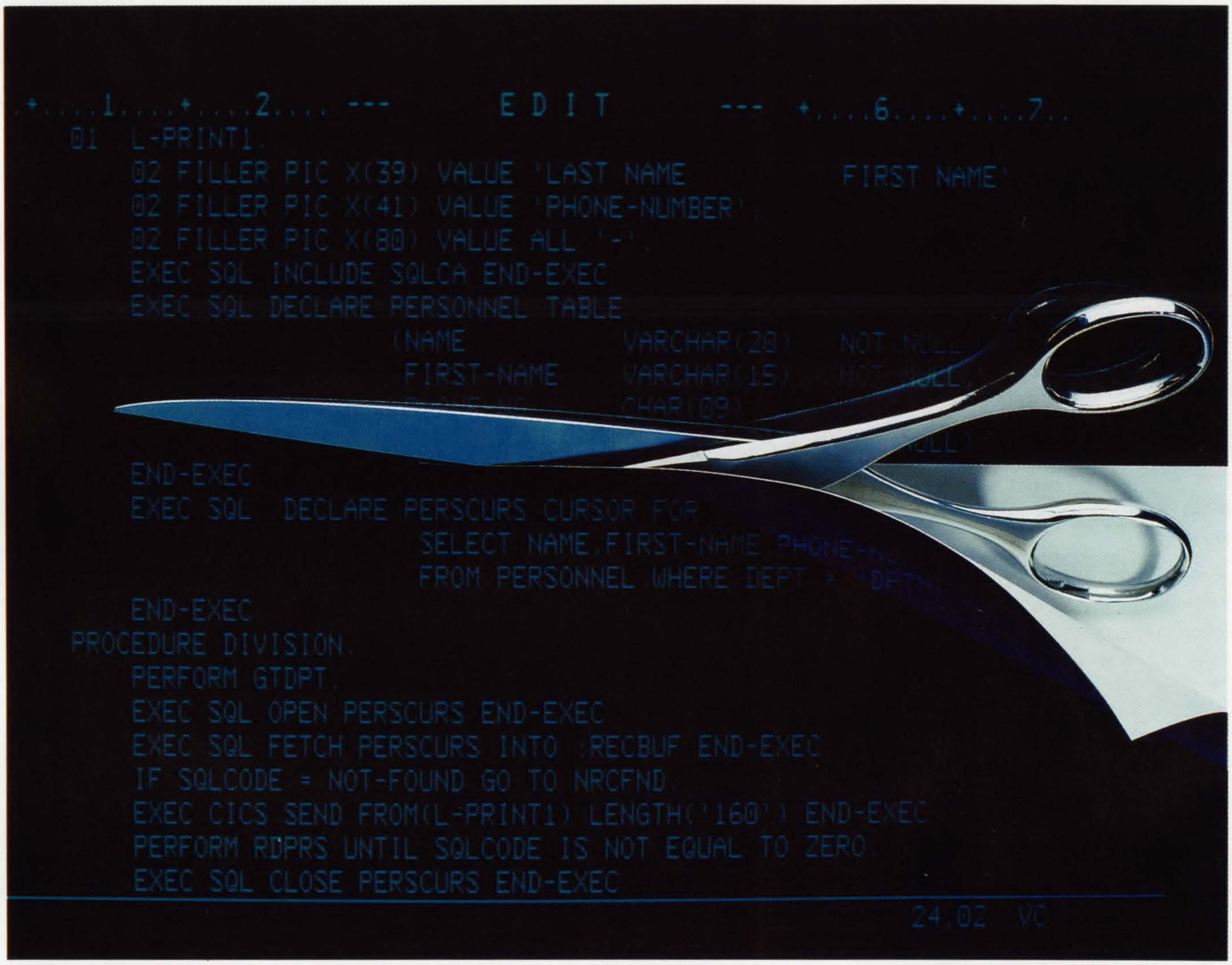
The long-term survival, efficiency and competitiveness of your company depends on the development and implementation of corporate information systems that reflect, define and support your organization's business mission, purpose, policies, strategies, tactics, goals, objectives, markets, channels, products, services and the Strengths, Weaknesses, Opportunities and Threats (SWOTs) to all of the above. Few, if any, of the CASE tools being marketed today will do this. They only contribute to the balkanization of the systems development effort.

Since few organizations have their *data house* in order, they continue to use the same old *process-oriented* methodologies of the past. Since the *processes* of doing business are constantly changing, the maintenance of systems continues to devour resources — resources that *could* be used to develop critical new systems if the proper corporate data structure were in place. The critical *data* of an organization remains stable over the years unless a company changes strategic direction or business orientation. Processes change every time a law is enacted or someone is promoted, dies or leaves the organization.

Why Old Methodologies Do Not Work

The process modeling, data flow analysis or current systems modeling methodologies of the past dictate that someone

Use NATURAL 2 to cut your application development backlog in half.



At thousands of sites around the world, NATURAL 2 is increasing application development productivity by at least ten times over conventional technologies.

If that sounds amazing, it is! The comprehensive 4th generation environment of NATURAL 2 provides a vastly superior approach to data processing. Its structured mode, minimum maintenance requirements and increased functionality cut out the 'hackwork'. So your DP people have more time to do the real work–thinking.

NATURAL 2 uses fewer lines of code to accomplish more than ever before. Thanks to its outstanding performance, it marks the beginning of the end of application development delays.

With NATURAL 2, the drudgery associated with using COBOL, FORTRAN and PL/1 disappears.

And NATURAL 2 supports a wide range of data structures, operating systems, TP monitors and hardware. So your investment is secured. Instead of trying to catch up with the past, you'll keep up with the future.

Cut out wasted time and effort. Call toll-free: 1-800-843-9534, for details on the beginning of the end of application development delays: NATURAL 2.

software AG
PROGRAMMING BUSINESS SUCCESS

CIRCLE #178 on Reader Service Card ▲

from DP or MIS must become an *expert* in some area of the business in order to design a system for that organization. This is usually attempted through a series of interviews. The functional expert supposedly tells the analyst everything he knows about his area of expertise or concern. The analyst goes off with what he *thinks* he heard and designs a prototype or the systems specifications. These are reviewed, changed, reviewed and so on ad nauseam until the specs are ready for *user sign off*. The user is asked to sign off on a bubble chart or data flow diagram that is usually incomprehensible to non-DP users. By the time the system is ready to go on-line, it is obsolete. The user's requirements have changed again. The metabolism of business is increasing every day. The old methodologies of systems design have not, and cannot, keep up with this increased metabolism.

Attaining A Competitive Advantage

Information Systems (IS) came up through the clerical side of the company. However, today you must focus on what is *strategic to the company*. This strategic emphasis is composed of goals such as improving quality, increasing productivity, lowering costs, enhancing management's efforts and differentiating customer services from those of your competitors. Faster, better, more efficient and more effective information systems will dictate who will survive, die or excel in the future.

A competitive advantage occurs after your technology investment is coupled with organizational transformation and changing the way the work is done. Organizational change and databases are the keys to new strategic systems. You must have intra-organizational information systems. All corporate data and information must become a shareable resource. All data must be available to all people. Only that information or data specifically designated by the board of directors, laboratories or senior officers of the company as being secret or proprietary to the company should be shielded from access by the *doers* of the company — the people that design, make, sell or service. This transformation will lead to greater effectiveness and efficiency throughout the organization. Leaders and leadership that you did not even know you had will evolve — leaders at the middle and lower levels in quality control, productivity and morale.

The Integrated Enterprise

The decision to affect this transformation to an *integrated enterprise* must come from the chairman of the board or CEO of the company. The decision must be seen as coming from the top and bear the full support of the executive staff. (Having said this, however, I must inject this proviso, "One man's tactics is another man's (or woman's) strategy." The *top* may just be a determined and knowledgeable division or department manager.)

Once the decision is made to develop a company-wide strategic information plan and data architecture, a Data Administrator (DA) must be appointed if there is not already someone in this position. The DA must be elevated as high as or higher than the current Manager of Information Systems (MIS) in the corporate hierarchy. All new development must first pass through the DA.

The DA must be able to cross organizational boundaries with tact and impunity. He must be able to communicate successes and failures to top management and both administer and monitor the derived end-user environment that will result from the data architecture. He must be a leader who will help people understand the systems that lie beyond the boundaries of their own system. There must also be a clear separation between the DA and the Database Administrator (DBA). The DBA functions on the technological level while the DA administers the data architecture on a planning and design level.

Once a *stable data model* has been generated and documented, the fleshing out of specific systems can be accomplished and the database design created.

It is at this point that CASE Tools can now be used most effectively! (The Computer Aided *Software* Engineering kind.)

This type of CASE can now attack a specific *system cluster* with the full knowledge of exactly where it fits into the overall strategic information plan. The analyst and programmer will know where the data is, how to get to it and how it inter-relates to other corporate systems. They can now utilize not only reusable code, but also *reusable design*. Teams can be assigned to various systems clusters within a major system for concurrent implementation.

Management can now get at *all* of the data in order to affect the expert business systems needed to run the organization and to generate *what-if* scenarios of anticipated policy changes or experimental marketing and distribution ideas.

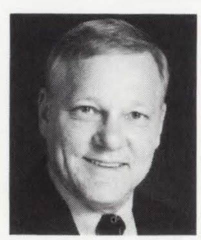
Prioritization And Quantification

Because senior management and functional experts have now been forced to explicitly state their informational needs on a prioritized basis during the modeling workshops, MIS is finally off the hook from constantly having to oil the *informational squeaky wheels* of the organization. They also now have a communications vehicle (the data map) from which to talk with users. Resources needed and systems priorities can be identified easily in a common language. Both user and DP now have become fluent in the common language of data modeling.

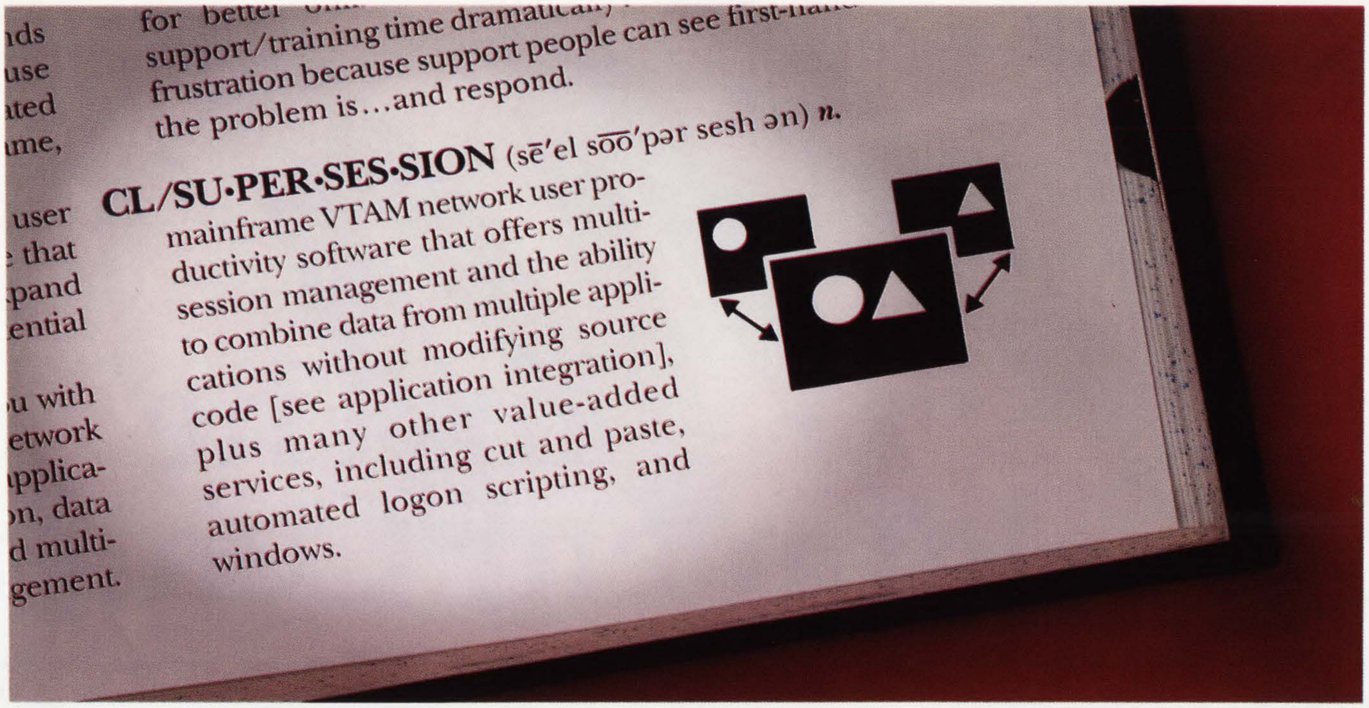
If you combine data modeling with some of the code generators on the market today, it will have a significant impact on the systems maintenance factor. Instead of 80 percent of your resources devoted to maintenance and 20 percent to new systems design, the data model can reverse those percentages to 20/80. Changes to a system are not made in the code — they are made in the *model* and the code regenerated.

In summary, investigate, use, accept and embrace CASE but only after you have put your data house in order with a stable, complete and concise data model of your strategic information plan and corporate data architecture. All of the above can be accomplished with a *data oriented* information engineering methodology. Software tools that automate the methodology backed by trained facilitators are also a must. Combine these with a properly staffed group of users and you are on your way to integrating your enterprise and getting your data house in order.

ABOUT THE AUTHOR



Dean F. Mohlstrom is President and CEO of Information Engineering Systems Corp. He has more than 28 years of experience in the data processing industry and is an active member of ADAPSO, DAMA, DPMA and The International Society For Planning and Strategic Management. IESC, 10935 Estate Lane, Dallas, TX 75238, (214) 340-8395 or Arlington, VA (703) 553-9222.



Only one product fits the definition of "application integration."

Application integration: we defined it before we designed it. Candle's CL/SUPERSESSION® for MVS offers VTAM network customization and application integration *by design*. Every feature is there for a simple reason: to expand your network's scope by giving you greater flexibility and important value-added services.

CL/SUPERSESSION makes it simple to customize applications across systems *without modifying a single line of source code*. Its dynamic cut-and-paste facility lets you quickly combine information from multiple sources into a single display - or distribute information from one screen to a multitude of applications. So you can virtually eliminate duplicate data entry.

CL/SUPERSESSION's smooth application integration is just one of many productivity payoffs. Because its multi-sessioning features exploit the latest advances in MVS and VTAM, an unlimited number of VTAM end users can have access to unlimited concurrent sessions. For additional power, CL/SUPERSESSION's unique Dialog Manager helps you quickly customize complex network procedures. The once frustrating task of manual logon can now be fully automated...and tailored to each user.

To improve network financial planning, CL/SUPERSESSION's Network Accounting Facility provides complete activity recording, including all network logons, session initiations/terminations, and traffic volumes for precise chargeback billing and usage auditing. And for at-a-glance monitoring of multiple sessions, our new windowing feature will let you view sessions side by side on a single screen. What's more, you'll never compromise on security. CL/SUPERSESSION interfaces with RACF, ACF2, and TOP SECRET.

Application integration and network customization *must* be simple, secure, and streamlined for your shop to boost productivity. That's why there's only one definition worth looking up: CL/SUPERSESSION for MVS. We'll be glad to spell out the details. Just call us today at **(312) 954-3888** (in Illinois) or **(800) 762-7608** (outside Illinois).

Candle®

CIRCLE #120 on Reader Service Card ▲

VM In The Development Center

VM provides a complete environment in which project teams can work from the initial requirements study to the final production turnover.

By Michael Seadle, Ph.D.

This article is the first in a series which describes how a development center can take advantage of VM's procedural language, interactive environment and electronic mail. The term development center refers here to those teams of programmers and users who create new application systems generally for MVS or VSE systems.

The main question is how to improve productivity for system development. VM offers three ways to do this. The first is through the use of tools like REXX that help automate the development process much as traditional systems have automated the way companies do business. The second is to involve users in design and testing as part of the project team in a VM information center environment. The third is to generate project documentation from electronic mail records.

Technical virtuosity alone builds no new systems. Productivity packages which focus on how to improve coding speed touch only one narrow aspect of the development process. VM provides a complete environment in which project teams can work from the initial requirements study to the final production turnover.

The Real Issue

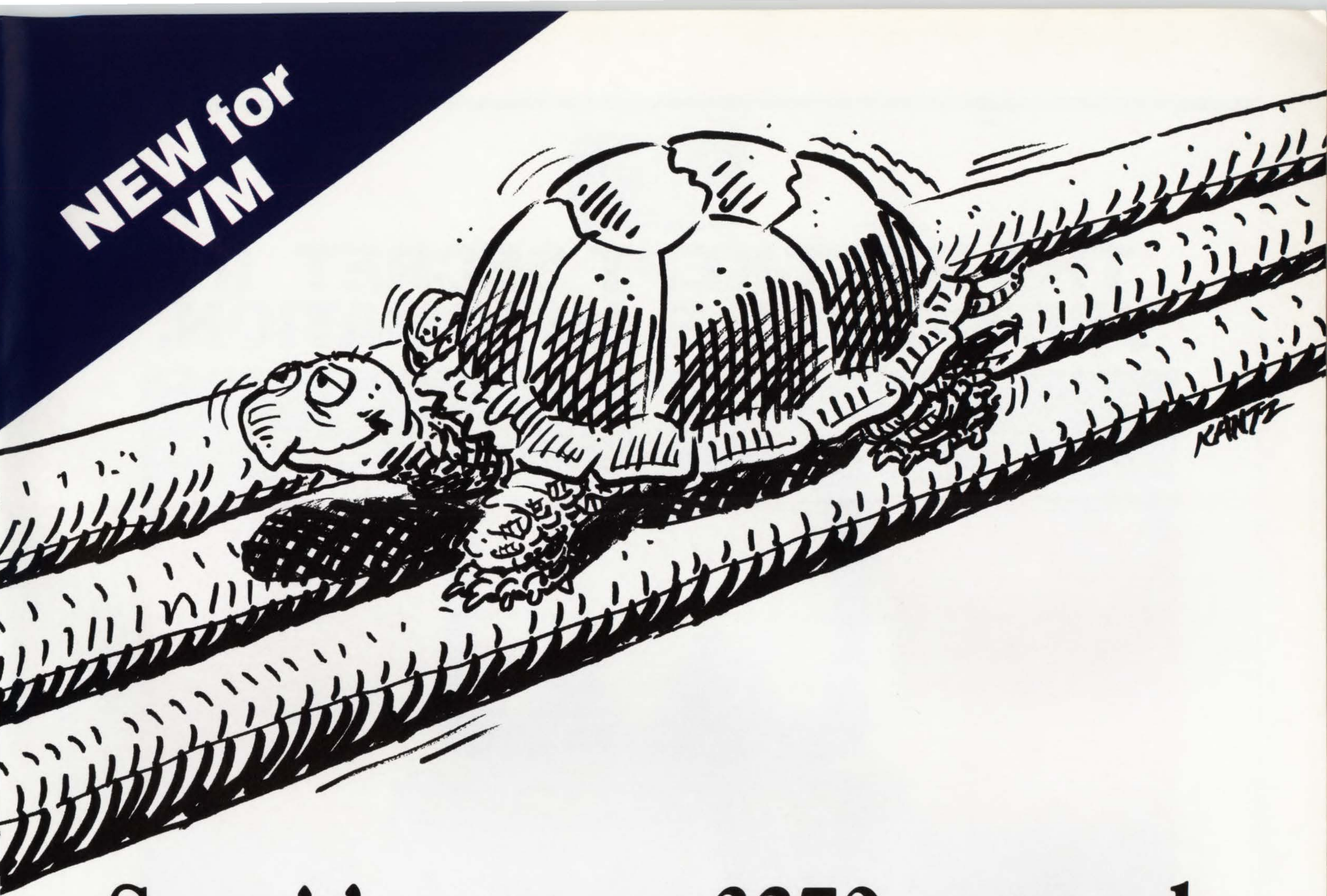
The real issue for a VM development center is whether its cost effectiveness justifies abandoning more traditional development centers in MVS or VSE. Many companies believe it does. Four common reasons are: VM's responsiveness, its portability among IBM processors, its tools and its information center capabilities.

Productivity today for users and programmers depends on their having quick response time. Each lag reduces the amount of work a person can do. Even occasional slowdowns hurt long term productivity as frustrated staff members begin to avoid on-line tools. VM is one means of keeping response time short. Applications programmers began moving to VM in the late 1970s because CMS offered a better interactive environment than TSO. VM's scheduling algorithm automatically favors interactive users over longer batch processes. Other IBM systems do not.

Portability from one model to another is another reason why development centers have migrated to VM. VM runs on all systems from the smallest 9370 to the largest 3090. A VSE shop contemplating growth into a high-end processor can migrate its development center staff into VM well before switching to MVS. At one time MVS shops discounted the reverse prospect of having to support smaller processors. However, departmentally based 9370s have begun to change that as have software products which link small- and medium-sized models into powerful clusters.

Productivity Tools

VM's productivity tools are widely considered the most user-friendly of any IBM system. Chief among these tools is



Sure it's a strong 3270 network. But is it moving your information fast enough?

No matter how finely you have tuned your system, or how often you've invested in hardware and telecommunications upgrades, information never moves fast enough for today's data processing demands. Except when you install 3270 SUPEROPTIMIZER™* from BMC Software, Inc.—the one-time solution for response time problems.

*U.S. Patent No. 4,750,137.

BMC's patented technologies of IMAGING™* and Input Suppression * assure you the maximum optimization of outbound and inbound data streams, which means messages travel faster across existing lines and process faster with existing hardware. You'll see up to 85 percent data stream reductions on local and remote 3270 networks, depending on the environment.

3270 SUPEROPTIMIZER is available for IMS, CICS, TSO and VM. You can have a dramatic response time improvement in less than one hour's installation time. To learn more, or to begin a free, 30-Day-Plus Trial of 3270 SUPEROPTIMIZER, call BMC Software at 1-800-841-2031, in Texas call collect, 713-240-8800.

BMC SOFTWARE

1-800-841-2031

BMC Software, Inc. • P.O. Box 2002
Sugar Land, TX 77487-2002

ICP MILLION DOLLAR AWARDS WINNER

Winner of the prestigious $10 million in sales award from ICP:
1988, 3270 SUPEROPTIMIZER™*/CICS
1989, 3270 SUPEROPTIMIZER™*/IMS

 BMC Software also has offices in France, Italy, Japan, the United Kingdom and West Germany.

CIRCLE #103 on Reader Service Card ▲

CIRCLE #65 on Reader Service Card ▲

REXX with its full range of programming capabilities. REXX allows development center staff of all sorts to write short ad hoc programs for almost any purpose, including prototyping. Similar capabilities exist in theory in MVS CLISTs; however, CLISTs are much less friendly and efficient. Proof of this is IBM's strategic decision to introduce REXX into MVS.

The savings such tools offer is equivalent to automating a business function. Development center staff rely on many pre-computer techniques. They might, for example, read a program page after page to find certain variables instead of scanning the code in a tenth of the time with an edit macro. Of course, the availability of tools like REXX or XEDIT does not guarantee their use. Many staff members will persist with old habits. But if the tools are there, some will adopt them, gain from them and teach them to others. Just as businesses began to use computers increasingly two decades ago, data center managers today must concern themselves with automating their own system development procedures.

The way VM most helps automate development is through the regular presence of users in a VM information center. This sets VM apart from other interactive environments. Communication among members of a project team has always been a key element in developing systems. In MVS and VSE shops this communication often takes place with the same tools available before computers: meetings, memos and telephone calls. In a VM shop the team members can use the power of the mainframe to make communication faster and more effective.

VM turns the machine readable record of these electronic discussions into useful documentation. Team members may spend hours in meetings discussing detailed questions and find the minute taker (if there is one) produced no more than a page or two of notes. The result is that the same questions appear again and again. If team members hold the same discussions electronically in VM the answers are there for anyone who cares to look.

Automating the development process through VM will require changing the way people work. For users this means learning how to do simple programming tasks in REXX and using electronic facilities like NOTE. For programmers it means learning to write clearly and succinctly. Frequently they have excellent literary skills in COBOL or PL/1 but lack training and confidence in ordinary English. Without changing work methods, VM still offers a solid platform for development center work, but it is like driving a sports car and never going over thirty.

A VM development center is no panacea. However, it is the best environment currently available for automating the way project teams do their work.

Electronic Discussions

The first stage of new system development is a study to define the system's requirements. This study involves broad discussion between programming staff and one or more groups of users. The result is a document describing specific features

The way VM most helps automate development is through the regular presence of users in a VM information center.

of the new system with supporting material on each point. VM facilitates this process by providing both an electronic mail system to speed communication and the means for turning that communication into a draft document.

Notes

Traditionally the first stages of the requirements study are verbal, either meetings or phone calls. Frequently minutes reflect only a small portion of the discussions in meetings and nothing at all of the phone calls. The authors of the final document must rely heavily on memory and must prepare draft after draft until one emerges that satisfies all parties. The normal medium of these drafts is paper. The turnaround time rarely allows more than one per week, depending on the internal mail system. Many project teams tolerate incomplete work because the process seems too slow.

The process need not, however, be slow. VM's electronic mail allows the exchange of several messages per day. For this to work effectively, both users and programmers must have easy access to VM and be prepared to use it. The whole project team must use the same tools to gain any of the productivity advantage.

A new project team should begin by using the NAMES command to set up an *address book*. The main purposes are: 1) to establish group names (NICKNAMEs) for the project team and simple entries for individual team members and 2) to designate a project NOTEBOOK for all project related correspondence — otherwise notes will automatically go into the default ALL NOTEBOOK file. Using the correct NOTEBOOK is important because the information will become part of the final requirements definition.

After setting up an address book, team members can send electronic mail by typing NOTE and the appropriate NICKNAME. NOTE sets up a full screen XEDIT session in memo form and automatically enters the Date, To and From information. These entries appear in the middle of the screen, not at the top as most people expect. You can change the appearance if you wish.

NOTE is nothing more than a program written in the old EXEC2 command language. It resides on the CMS system "S" disk as does PROFNOTE XEDIT, the XEDIT profile that NOTE uses. This ready access may tempt you to make substantial revisions but remember that IBM provides ongoing fixes and updates for the original version of NOTE. Major changes could mean debugging updates later on.

The safest modifications affect only PROFNOTE XEDIT. You can change the default location of the Date, To and From lines by adding the following to a copy of PROFNOTE XEDIT:

COMMAND SET CURLINE 5

The CURLINE is normally 12 lines from the top of the screen. This XEDIT command moves the *current line* (CURLINE) down only five lines from the top. The Date, To and From display shifts upwards as a result.

This solution creates a new problem. If you use the PF2 command to add lines, all previous text moves upwards and only a few lines remain visible. You need an-

See VM *page 96*

San Bernardino County's Checklist For Change Control

By Julio A. Hernandez

Two Los Angeles street gangs, the Crips and Bloods, are moving into the largest county in the continental United States — San Bernardino County. Law enforcement is ready to combat the problem. The data processing arm of San Bernardino County, the Office of Management Services (OMS), is providing a helping hand with its Jail Information Management System (JIMS). JIMS is a computer system that will keep track of the *gang-bangers* once they are apprehended. The County has a 3081 and a 3090 running under MVS/XA 2.2.

JIMS will process a suspect from intake, book him, check and serve warrants, automatically calculate bail, assign a cell and keep track of the inmate as he winds his way through the system. As jail populations increase and determinate sentencing laws put even more pressure on overcrowded facilities, it becomes imperative that JIMS be up 24-hours-a-day. To that end and to prevent "serious harm to the integrity" of all County systems, Marland Howard, director of OMS, decreed that all application changes must be closely monitored to avoid production instability. The stated purpose was *change control* within OMS.

Change control is not unique to San Bernardino County. It is a common bond shared by all MIS departments. At OMS, the quest for tighter controls on application changes resulted in a fact-finding mission. The following guidelines are a by-product of that study.

Defining Change Control

Every system carries with it the potential to be changed whether requested by the user or necessitated by a program abend. Managing those changes requires asking the "who, what, where and when" of change control.

Who Authorized These Changes?

There should be one focal point for authorizing a change. In JIMS, that responsibility falls on the project leader. He acts as a coordinator to formally authorize, review, document and assign a change. This approach ensures that potential problems are minimized if not eliminated. No one in JIMS is allowed to make changes without the project leader's authorization. This authorization process uses a standard form so that the user and other appropriate people *sign-off* the requested change. (In other MIS shops, this approach could be handled electronically through an *on-line form.)* Once the necessary approvals are obtained, the programmer can proceed with the request. The only exception to this signing-off process would be a production problem, that is, where "putting out a fire" would require immediate action.

What Is Being Changed?

The authorizer acts as the liaison between the user and the programming team. It should be his responsibility to coordinate what is being changed and to monitor *concurrency* fixes should they arise. Moreover, he not only identifies what is being changed, but also he prioritizes the changes based on the business needs.

Where Is This Change Being Done?

There should be a standard as to what libraries the programming team uses. If the shop has its own test library, then that standard should be enforced. Program-

mers should be discouraged from creating their own *temporary* library — carrying duplicate source code. An exception may be considered if a programmer is enhancing source code not yet unit-tested and a production abend occurs on the original source. This is the *concurrency* fix mentioned above.

When Will This Change Be Moved?

No one needs to bump heads when implementing the changes. It is frustrating and pointless in getting the job done. For example in JIMS, if my colleague, Dave, is doing an extensive change to a copybook and scheduling it for production, then he should alert the rest of the team when that change will be in effect. In that way, we can delay implementing our program(s) in order to pick up the changed copybook. Or we may prefer to *synchronize* our implementation with Dave's. It will be an all-or-nothing proposition. Should Dave's changes fail to go across, then our changes will not be moved and the integrity of the system is maintained. Therefore, the key again is *coordination*. Accordingly, the shop should enforce a policy as to when changes may go in, that is, either daily, at a specified time or on a weekly cycle.

Defining Production Turnover

Another facet of change control is *production turnover*. Turnover is the next logical step once a change is completed. It is an orderly process in which the changed source is implemented into the production environment. Three new "who, what and where" questions pinpoint the turnover process.

Who Is Authorizing The Move?

A policy should be formulated designating who approves a move. Ideally, it should be the same person who authorizes the change. In addition, the authorizer, through the approval process, should communicate all change activity to various departments within the shop. Management, operations, project leaders and users need to know if everyone affected *bought off* the changes and are receptive to implementation. For example, a report could be generated nightly that would show all change activity to the production libraries. This report would be distributed to the aforementioned people.

What Are The Backup Procedures?

In JIMS, there are more than 4,000 bookings per month. Statewide, county jails had an average of 60,802 prisoners during 1987. When an application program causes the system to go down, the programmer *on call* for that night attempts to correct the problem as quickly as possible. The urgency, however, must be tempered so that the programmer avoids straying from a prudent course. A recent incident in JIMS underscores this point.

Awakened at 3 a.m. by computer operations, the operator informed me that a Sgt. McKay was unable to *book* an inmate. The booking screen, I was told, was out of alignment. Calling to mind the team motto — "Keep our users happy — they carry guns!" — I dressed quickly and arrived at work within half-an-hour. After the third cup of coffee, I uncovered the problem. It was, as a writer once wrote when referring to human frailties, an occasional *faux pas*. We all make mistakes

"Keep our users happy – they carry guns!"

and this one was mine. Meticulous as I had tried to be, nonetheless, I overlooked moving a changed copybook. I was faced with a decision: back out the changes (which were extensive and involved several sub-modules) or fix the change by implementing the new copybook and recompiling all affected programs. On the surface I was inclined to rollback the original source until I recalled the changes made to the database structure. The old module could no longer be used without rolling back the database structural changes. That was a time-consuming approach. Consequently, the changed copybook had to be moved and the attendant programs re-compiled. By 5 a.m., Sgt. McKay had his man *booked* into the system.

In this situation, the decision was made not to back out the changes. In other shops, that may not have been the course to follow. In any event, this true-to-life example underscores why a policy should be established on backup procedures. What should be backed up, both the source and load? How many versions do you retain? And for how long? Do you reassemble the original source or rollback the original load module? Do you create an emergency environment for quick fixes? Do you then bypass security? How much *downtime* should you allow before backing out the changes?

What audit trails do you create in situations like this? What are the mediums? Do you use tape, disk or a combination of both? What happens if you experience a *head crash* with both the backup and original on the same disk? What recourse do you then have? These are the areas that typically need to be addressed when formulating a backup procedure.

Where Is The Source Being Moved To?

A decision should be made to either use a staging library or to move, compile and link-edit straight into the production nucleus. A staging environment affords the opportunity to capture any problems that may have bypassed careful measures. It is an extra check to ensure the integrity of the production system. In either case, whether using a staging library or not, the basic principles of backup, move, compile and link-edit should be followed. These are the building blocks that cement the production turnover process. Variances in this mix weaken the foundation and invite trouble. Using utilities to separately copy load modules and to then copy source may not be the best method to employ. The shop must ensure that the executable load is always from that source. This is the goal of change control. Moving source to a library and then compiling from that environment is a guarantee that the load and source are in sync.

This article is meant to serve as a guide for shops managing and designing a change control system. Every MIS shop needs to manage its application changes. The absence of controls only debilitates the production system — be it retail, distribution or even a jail management system.

ABOUT THE AUTHOR

Julio A. Hernandez is a senior applications programmer in the Office of Management Services, San Bernardino County, CA. In data processing for 12 years, he serves as chairman of the Application Change Control Committee. Prior to his position with the County, Hernandez was McDonnell Douglas' west coast project leader as a programmer/analyst.

ICF Catalog Structure

Having two datasets in the ICF catalog solves backup and recovery problems.

By Blair Svihra

The redesign of the old style VSAM catalog produced a new catalog structure called the ICF catalog. That redesign solved many of the backup/recovery problems with VSAM by splitting the catalog into two datasets: the Basic Catalog Structure (BCS) and the VSAM Volume Dataset (VVDS).

The BCS is functionally equivalent to a CVOL in that it is primarily a dataset locator. Vital information (AMDSB information, extent information, dataset attributes and so on) is located in the VTOC and the VVDS on the volume where the datasets reside away from the catalog. BCS information includes the dataset name, volser and device type the component resides on, ownership information, dataset security information (if applicable) and possible VVDS and VTOC RBA pointers. To facilitate quicker access to the components on a volume, the Relative Byte Address (RBA) of the Format-1 DSCB VTOC entry and, if the component is VSAM, the RBA of the CI containing the VVR records in the VVDS that describe the dataset, are stored in the catalog. This saves a sequential search of the VTOC and/or VVDS to locate the associated information. If the RBA is incorrect, the correct RBA values will be updated in the catalog after a sequential search is done.

The basic catalog structure is a pure Key Sequenced Dataset (KSDS). It can be opened and processed as a normal keyed VSAM dataset. The cluster name in every ICF catalog is binary zeros (x145'00'). That is right, I said 45, not 44 as in the Format-1 DSCB. This forces the catalog's self-describing record to a fixed location in the BCS where it can be read directly from within the BCS by channel program I/O before the BCS has ever been opened. The name specified in the IDCAM's 'DEFINE' is actually the name of the catalog's data component. The associated index component is automatically given a generated name of

CATINDEX.Ttimestamp.VIDyyddd.Ttimestamp.

Some of the key concepts of VSAM catalogs were removed in the design of ICF catalogs. Suballocated space, suballocated datasets, Catalog Recovery Areas used with recoverable catalogs (CRAs) and volume ownership have been eliminated in the ICF environment.

Unlike the old VSAM catalog internal format (low-key and high-key fixed length structure), some clever, clear-headed thinking went into the design of the internal structure of the BCS and VVDS. The use of variable length records, based strictly on how much information has to be in a record, has significantly increased utilization inside the BCS and VVDS. Since the BCS is processed as a regular KSDS, records can be updated (expanded and shortened), inserted and deleted just like a regular keyed VSAM dataset. Overall flexibility is gained by allowing the ICF catalog to be defined with user specified dataset attributes (CI/CA Sizes, record size, free space and so on) and performance tuning parameters (imbed, replicate, bufferspace and so on).

To achieve the variable length record capability, a cellular structure was designed for BCS records. Related fields of information are grouped into cells, cells are grouped into components, compo-

nents are grouped into subrecords and subrecords are grouped into sphere records. If a particular unit of information is unavailable or not needed, then it is simply left out. If multiple cells are required, they are arranged in a predetermined sequence (one cell after another). If an additional cell needs to be added later, the cell is inserted at the appropriate location and the record is written out at greater length.

With the advent of System Managed Storage (SMS) in Data Facility Product (DFP) Version 3, subcells containing storage class, data class and management class information were introduced. Instead of introducing new cells to contain SMS information, IBM chose to attach a subcell to the end of a select number of fixed length cells in the BCS (only if the dataset is SMS managed). Why follow standard cell conventions when another unit (subcells) can be introduced to keep us on our toes? New SMS subcells, cells and records in the VVDS will be discussed later on.

Every cell and subcell has a cell type code in the third byte that uniquely identifies it. Table 1 illustrates all of the types of cells/subcells that are used in the construction of BCS records.

TABLE 1

Type Code		Usage And Contents
x'01'		Ownership Cell containing ownership (historical) information for a component.
x'02'		Security Cell containing security information for VSAM components.
x'03'		Association Cell containing the names of: Paths associated with a cluster or AIX Cluster and/or AIX associated with a path Aliases associated with a non-VSAM dataset Dataset associated with an alias Base cluster and AIX, if any, associated with a truename.
x'04'		Volume Cell containing volume information for a VSAM or non-VSAM dataset component.
x'05'		Generation Aging Table Cell containing the control information for a GDG (such as limit, scratch/noscratch and empty/noempty attributes).
x'06'		Relation Cell containing the names of AIXs associated with a base cluster.
x'26'		BCS SMS Subcell containing SMS storage class, data class and management class information.
x'C1'	A	Non-VSAM Name Cell indentifying a non-VSAM dataset.
x'C2'	B	Generation Data Group Base Name Cell indentifying a GDG.
x'C3'	C	VSAM Cluster Name Cell identifying a VSAM base cluster.
x'C4'	D	Data Name Cell identifying a data component of a base cluster or AIX.
x'C5'	E	VSAM Extension Name Cell identifying an extension record for a VSAM cluster.
x'C7'	G	Alternate Index Name Cell identifying the cluster component of an AIX.
x'C8'	H	Generation Dataset Name Cell identifying a non-VSAM dataset that is a generation of a GDG.
x'C9'	I	Index Name Cell identifying an index component of a base cluster or AIX.
x'D1'	J	GDG Extension Name Cell identifying an extension record for a GDG.
x'D9'	R	Path Name Cell identifying a path name of a base cluster or AIX.
x'E3'	T	Truename Cell identifying a D, I, or G component that allows access to the individual component through its own name.
x'E4'	U	ICF Connector Name Cell identifying a VSAM or ICF user catalog connector record.
x'E7'	X	Alias Name Cell identifying an alias of a non-VSAM dataset, generation dataset or pointer to a specific user catalog.

For a description of the individual fields that comprise a cell, refer to IBM's *Catalog Diagnosis Reference* manual (LY26-3956).

VSAM Cluster Sphere Record

The VSAM cluster sphere record for a KSDS, ESDS and RRDS is composed of a variable number of subrecords. For a KSDS there will be at least a cluster subrecord (made up of a cluster, data and index component). If an alternate index is associated with a KSDS, then an additional AIX subrecord will be present (made up of a cluster, data and index component). There will be one AIX subrecord for each alternate index defined over the base cluster. For an ESDS or RRDS, a cluster subrecord (made up of a cluster and data component) is also required. An AIX subrecord will also be present for each associated alternate index. The general format of a base cluster sphere record will be as illustrated in Figure 1.

For KSDS, ESDS and RRDS datasets, the cluster component will be composed of the cells in the order shown in Figure 2. The SMS subcell, security cell and association cell are optional and will only be present in the cluster component if required.

All data and index components have the same cellular structure whether they are base clusters or alternate index clusters. The security cell is, again, optional and additional volume cells may be present. The general format of the data/index component will be as shown in Figure 3.

The alternate index cluster component is similar to the base cluster component except for a few modifications. The security cell is optional and the SMS subcell and the relation cell are not present. The general format for the alternate index cluster component will be as shown in Figure 4.

FIGURE 1

Base Cluster Sphere Record

record length	cluster comp.	data comp.	index comp.	cluster comp.	data comp.	index comp.

|------------- cluster subrecord -------------| |------------- AIX subrecord -------------|

|-- cluster sphere record --|

FIGURE 2

Cluster Component

name cell x'C1'	SMS subcell x'26'	owner cell x'01'	security cell x'02' optional	relation cell x'06'	association cell x'03'

FIGURE 3

Data/Index Component

name cell x'C4' or x'C9'	owner cell x'01'	security cell x'02' optional	volume cell x'04'

FIGURE 4

Alternate Index Cluster Component

name cell x'C7'	owner cell x'01'	security cell x'02' optional	association cell x'03'

FIGURE 5

Path Record

record length	name cell x 'D9'	owner cell x'01'	security cell x'02' optional	association cell x'03'

FIGURE 6

Truename Record

record length	name cell x'E3'	association cell x'03'

Path Record

There will usually be at least one path defined for each alternate index that is built for access into an ESDS or KSDS. The primary purpose of the path is to describe open access into the base cluster through the alternate index. The name of the base cluster and the alternate index cluster are stored in the association cell. The general format of the path record will be as shown in Figure 5.

Truename Record

A truename record is created for every data component (D), index component (I) and alternate index cluster component (G). The association cell will contain the name of the components' base cluster and in the case of truename records for the data and index components of an alternate index, the alternate index name will be present. The general format of the truename record will be as shown in Figure 6.

Non-VSAM Record

In the non-VSAM record the SMS subcell and the association cell are optional and additional volume cells may be present. The general format of the non-VSAM component will be as shown in Figure 7.

Generation Data Group Sphere Record

The generation data group sphere record is composed of one GDG base subrecord and a variable number of GDS base subrecords. The GDS base subrecord contains all of the descriptive information about the GDG. Immediately following the GDG base subrecord will be one GDS base subrecord for each generation of the GDG that is being maintained. It is interesting to note that LISTCAT interprets this single sphere record as multiple non-VSAM records (generations) and adjusts the record count at the end of the LISTCAT. There is not a one-to-one correlation of the record count to the actual number of physical records in the catalog. The general format of the generation data group sphere record will be as shown in Figure 8.

The generation data group base subrecord contains descriptive information on the GDG base itself — limit, current number of GDGs, attributes (empty/noempty, scratch/noscratch) and so on. The general format of the generation data group base subrecord will be as shown in Figure 9.

There will be one generation dataset subrecord for each active generation in the GDG. The name cell identifies the particular generation and version numbers. The SMS subcell and the association cell are optional and additional volume cells may be present. The general format of the generation dataset subrecord will be as shown in Figure 10.

Catalog Connector Record

Catalog connector records normally reside in the system's master catalog. They contain the name of a user catalog and the volume and device type where it resides. An association cell, if present, contains a list of alias entries related to the user catalog. The general format of the catalog connector record will be as shown in Figure 11.

Alias Record

Alias records are usually found in the system's master catalog but can be found in user catalogs for non-VSAM and GDG relations. The most common usage of aliases is to establish user catalog alias values that are matched with high-level qualifiers to help determine which VSAM user catalog is to be used or searched during a dataset definition or locate operation. The general format of the alias record will be as shown in Figure 12.

FIGURE 7

Non-VSAM Record

record length	name cell x'C1'	SMS subcell x'26' optional	owner cell x'01'	association cell x'03' optional	volume cell x'04'

FIGURE 8

Generation Data Group Sphere Record

record length	GDG base subrecord	GDS base subrecord optional

FIGURE 9

GDG Base Subrecord

name cell x'C7'	owner cell x'01'	GAT cell x'05'

FIGURE 10

GDS Base Subrecord

name cell x'C1'	SMS subcell x'26' optional	owner cell x'01'	association cell x'03' optional	volume cell x'04'

FIGURE 11

Catalog Connector Record

record length	name cell x'E4'	owner cell x'01'	association cell x'03' optional	volume cell x'04'

FIGURE 12

Alias Record

record length	name cell x'E7'	association cell x'03'

Notice that record lengths are stored at the beginning of each BCS record. VSAM record management uses control information at the end of each Control Interval (CI) to determine the length of an individual record. The length at the beginning of each record should match its corresponding Record Definition Field (RDF) value. If a descrepency between the two exists, the RDF value takes precedence.

The VSAM volume dataset is a special Entry Sequenced Dataset (ESDS). Its name, SYS1.VVDS.Vvolser, is a reserved name to open. The volser in the VVDS name must match the volser in the Unit Control Block (UCB) of the device it resides on. The VVDS contains extent information, high allocated/high used RBA values, complete Access Method Data Statistics Block (AMDSB) information, dataset attributes, SMS class information and so on for all VSAM (and SMS managed non-VSAM) datasets on the volume. By removing this information from the catalog and having it reside on the volume with the datasets it describes, recovery problems are vastly improved.

The VVDS is created explicitly by IDCAMs 'DEFINE CLUSTER' or is explicitly defined when the first VSAM dataset is defined on the volume. The default space allocation is TRK 3,2 for pre-DFP Version 3 and TRK 10,10 for systems with SMS support. Its CI Size is always 4K (4096) and it is considered a high activity dataset that should be placed near the VTOC.

TABLE 2

Type Code		Usage And Contents
x'21'		VVR Dataset Information Cell containing attributes of a VSAM data or index component.
x'22'		NVR Dataset Information Cell containing SMS related information for the first volume of a non-VSAM dataset.
x'23'		VVR Volume Information Cell containing allocation information for a particular volume for a VSAM data or index component.
x'24'		VVR/NVR SMS Subcell containing storage class, data class and management class information for a VSAM data or index component or a non-VSAM SMS managed dataset.
x'60'		VVR AMDSB Cell containing the Access Method Data Statistics Block (that is, statistics fields) of a VSAM data or index component.
x'D5'	N	NVR Header Cell identifying an SMS managed non-VSAM dataset.
x'D8'	Q	VVR Header Cell identifying the secondary VVR.
x'E9'	Z	VVR Header Cell identifying the primary VVR.

FIGURE 13

VVCR Record

record length	CI count	c'VVCR'	reserved	36 BCS catalog names	CI free space map
bytes 2	2	4	16	2016	2048

FIGURE 14

Primary VVR Record

record length	primary header cell x'E9'	dataset information cell x'21'	SMS subcell x'24' optional	AMDSB cell x'60'	volume info. cell x'23'

Like the BCS, the VVDS contains a number of records composed of cells and subcells. The list in Table 2 shows all of the types of cell/subcells that are used in the construction of VVDS records.

A number of record types exist in the VVDS. The VSAM Volume Control Record (VVCR) is a full CI in length (4096 bytes) and is always located at RBA = 0. It contains the names of up to 36 basic catalog structures having VSAM dataset components residing on the volume. It also contains an internal space map detailing each allocated CI in the VVDS. The space map is comprised of two byte entries describing the available free space in an allocated CI (that is, it matches the CIDF+2(2) of its corresponding VVDS CI). The process of adding a new VSAM Volume Record (VVR) to the VVDS during a 'DEFINE' uses the VVCR space map. A CI with available free space to hold the VVR being added can be located quickly without having to do a sequential search of the VVDS. Also, when updates are made to a VVR causing it to increase in size, the space map is used to locate a CI that has sufficient free space if the record will not fit back into its original CI. This facility allows lengthening of records, an update type that is normally not allowed in an ESDS. If the value in a VVCR space map entry does not match the value in its corresponding CIDF, the CIDF value takes precedence.

The VVCR is one large record and is not composed of cells. The general format of the VVCR record will be as shown in Figure 13. The eyecatcher c'VVCR' was added in DFP Version 3. Pre-DFP systems have 20 bytes of reserved storage.

The VVDS self-describing VVR is the second record in the VVDS. It is always located at RBA = 4096 and is the only record located in this CI. This allows ample room for the self-describing VVR to expand as needed. The VVDS self-describing VVR contains references to the VVDS name — SYS1.VVDS.Vvolser and fully describes the VVDS. This is the only reference in the VVDS to the volser that the VVDS resides on (that is, the volser in the VVDS name).

FIGURE 15

Secondary VVR Record

record length	secondary header cell x'D8'	volume information cell x'23'

FIGURE 16

NVR Record

record length	NVR header cell x'D5'	NVR dataset information cell x'22'	VVR/NVR SMS subcell x'24' optional

The World's Most Successful Software Companies Are Sold On Software Management.

Compuware, Legent, Management Science America, Policy Management Systems and Systems Center are five of the most successful and respected names in the software industry. Together, their customer bases include almost every IBM site around the world–representing over 40,000 installed mainframe products.

These companies have all taken yet another step to provide their customers with the highest quality software. They are implementing the industry's only automated software management system–ENDEVOR.®

Using ENDEVOR, these vendors are able to manage software development and distribution within a unified environment. ENDEVOR automatically tracks every component of a software system from the time it's designed through testing and release. The results–shorter development cycles, fully reliable systems, on-time delivery to users and a more sophisticated approach to customer support.

Now you can bring all the benefits of the ENDEVOR Software Management System to your organization. With ENDEVOR, you can manage the creation, evolution, distribution and operation of your vendor-supplied and internally developed software–automatically. Whether it's traditional, 4GL, CASE or DBMS based.

The ENDEVOR Software Management System provides:

- Inventory and library management for classifying and tracking software components
- Change control that produces audit trails, source-to-executable links, and ensures standards enforcement
- Configuration management that automatically captures application interdependencies for point-in-time recreation and change impact analysis
- Release management that automatically controls the distribution of applications from test to production and to remote sites

If you're concerned about automating the management of your organization's application systems and are interested in improving the quality of production turnovers, software distribution, vendor application updates, software documentation, application stability, change implementation or change impact analysis, follow these software industry leaders.

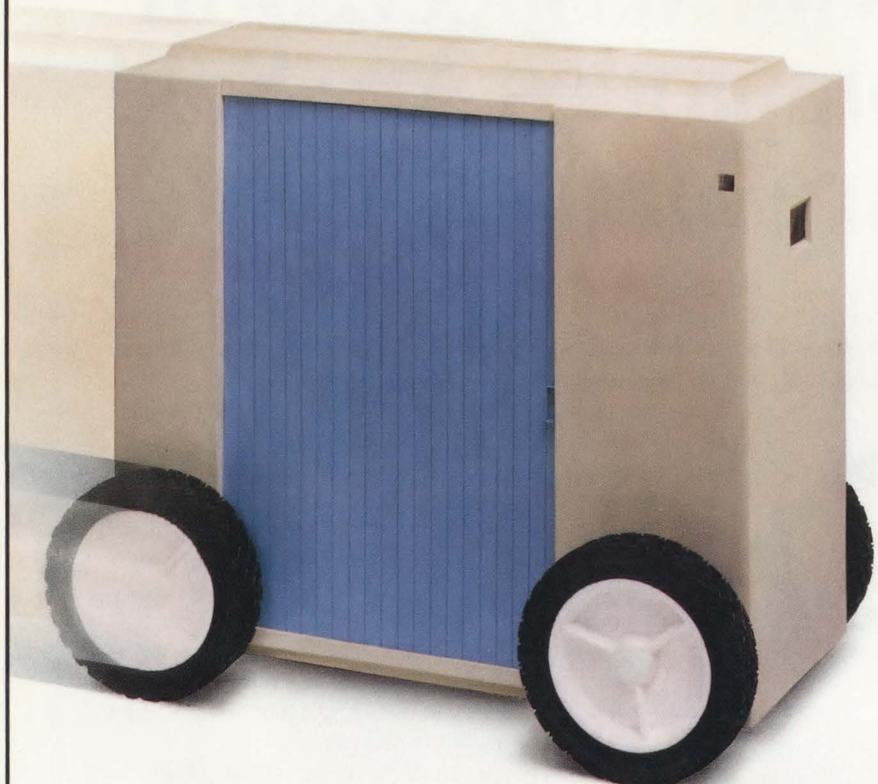
Call us at (508) 870-1900 and ask us how ENDEVOR has already worked wonders at over 200 top installations like yours, and at the top software firms in the world.

BST

The Software Management Experts

Business Software Technology, Inc., Westboro Executive Park, 114 Turnpike Road, Westborough, MA 01581-9990
ENDEVOR is a registered trademark of Business Software Technology, Inc.

CIRCLE #117 on Reader Service Card ▲



VVRs describing the VSAM components on the volume and the SMS managed non-VSAM datasets begin at RBA = 8192. There are three types of VVRs in a VVDS — Z (primary VVR, VSAM only), Q (secondary VVR, VSAM only) and N (SMS managed NVR, non-VSAM only). The type 'Z' and 'Q' VVRs are composed of multiple cells which fully describe their corresponding VSAM components on the volume. Type 'N' NVRs do not fully describe their corresponding component as a VVR does. NVRs only contain SMS related information — no AMDSB, extent and so on statistics. The general format of the VVRs and NVR will be as shown in Figures 14 through 16.

Earlier I mentioned that because of the 36 catalog name restriction in the VVCR, only 36 BCSes could have VSAM defined on a particular volume. APAR OY15152, and others, removes the 36 catalog name and VVDS size limitation. Two new record types, VVCM (VVCR space map extension) and VVCN (VVCR catalog name extension) were created to address this problem. Whenever a VVDS runs out of catalog name slots or space map entries, an empty CI will be located and devoted to either a VVCN or VVCM. These records are chained off the VVCR by RBA. The VVCM enhancement seems like an adequate solution to the problem of undersized VVDSs when shops begin implementing SMS. VVDSes with the default space allocation (TRK 3,2) have a high chance of their VVDSes going into multiple extents. This will cause fragmentation and possible performance problems when extra SMS subcells, cells and records are created. It is advisable not to use IBM's defaults when allocating a VVDS but to overallocate the VVDS and allow room for growth in one contiguous extent. As for the VVCN enhancement, the more catalogs that are affected in a recovery situation, the more complicated the recovery can become.

So just when you thought it was safe to recover VSAM, think again!

ABOUT THE AUTHOR

Blair Svihra is the product manager for VSAM MECHANIC and CAT SCAN at Softworks, Inc. He has more than 10 years of technical expertise with MVS operating systems and has worked extensively with VSAM and catalog management internals. Softworks, Inc., 7700 Old Ranch Ave., Clinton, MD 20735, (301) 868-4221.

PLAYING For KEEPS In Data Center Chargeback

With the cooperation of motivated users, the chargeback system becomes the ultimate system tuning tool.

By Kenneth R. Whitaker

Most companies do chargeback but only for reporting purposes. Users are told the quantities and costs of the resources they consumed but are not held accountable.

If users are made aware of the true costs of their data center usage and are held accountable, they will act more responsibly. The savings to the company could be enormous.

How can you slow the ceaseless growth in data center costs? The best way is to make your users aware of the effect of their actions.

Users often submit jobs without considering the cost. Many think of the data center as a free and plentiful resource like water. They do not act this way with any other expenses. Compare the decision process in deciding to buy a $200 airline ticket with that of deciding to run a $200 job.

You (the data center manager or systems programmer) cannot judge the necessity of each job. You have no choice but to assume that all jobs are necessary and to provide enough computer power to supply the demand.

The users are the only ones who can judge the necessity of the jobs they run. If forced to think about it they might realize that some jobs can wait until the off-peak shifts and that other jobs are not needed at all.

Reducing the workload, especially the peak workload, results in improved performance and can postpone the eventual upgrade to a larger system. You will actually get more for less. When this philosophy is extrapolated to disk space, printing and so on, the amount of money that can be saved is staggering.

What is needed is motivation to get the users to start thinking about data center resource consumption.

Funny Money

Users have no problem accepting other types of charges. Their budget is charged when an airline ticket is bought. They understand this and accept it. Why is a data center charge harder to understand or accept than an airline charge?

The difference is that the airline charge is incremental and direct. The data center charge is more abstract. When a user submits a job, he knows that no real expense was incurred because of that single job. No new operators were hired and no new system was installed. The company's expenses would be the same had he not submitted the job. Any charge, then, is considered by many users to be artificial.

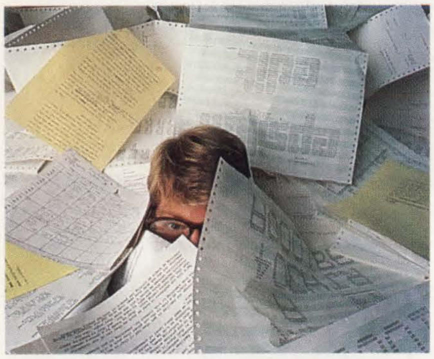
There really is such a thing as funny money. It is a concept in people's minds. The company's fiscal policy creates funny money when it allows these concepts to persist. You can create funny money in several ways.

- You can tell your users what their charges would have been in the real world and then not charge them at all. Your users will see their invoices and then they will throw them away.
- You can charge your users inconsistently. Charge them $100 one day, then $1,000 the next day for the same thing. Your users will have no confidence in your system.
- You can charge them at a modest amount for each job all year, then surprise them at year-end with an enormous adjustment. They will feel that their actual usage has no relationship to their charges.

Funny money is not a motivator. A $1,000 on an artificial invoice does not

New for your MVS or VM environment!

Are you under the gun to create documentation?



HERE'S AN EASY WAY TO BITE THE BULLET.

Everyone knows that their Data Center should document important policies, standards, and procedures. The trouble is, with all the crushing pressure of work, documentation always seems to go on the "back burner." These days, that can land you in hot water. With all the strict regulations and tough compliance requirements you face, it becomes *absolutely necessary to take action* and finally solve the "documentation dilemma."

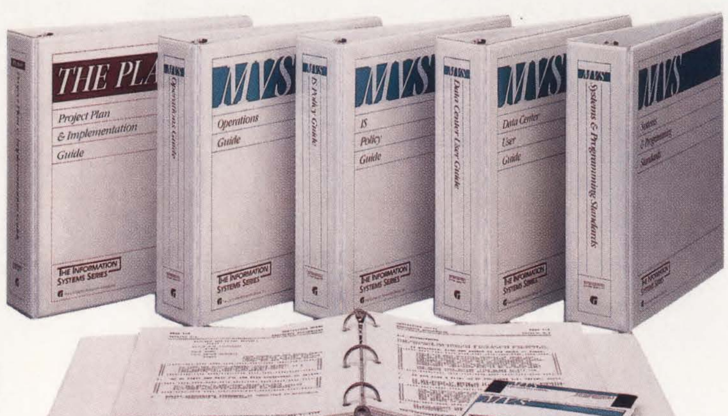
Luckily, we can make compliance easier than you ever dreamed possible!

Introducing the Information Systems Series.™ High caliber documentation the painless way.

The ***Information Systems Series*** documentation framework is designed to help you create comprehensive policies, standards, procedures, and user support materials for your entire MVS or VM Data Center.

Created by highly experienced data processing experts, the ***Information Systems Series*** is an interlinked set of five manuals and accompanying, corresponding diskettes: *The IS Policy Guide, The Data Center User Guide, Systems and Programming Standards, The Operations Guide,* and *The Project Plan and Implementation Guide.* All you have to do is follow the simple, imbedded instructions. In other words, you *personalize* the documentation using your own IBM PC or compatible – which means you can easily integrate any existing documentation quickly and easily!

An important part of the ***Information Systems Series*** is our unique *Project Plan and Implementation Guide* that leads you effortlessly through the entire documentation process. No wonder many forward-looking government, business, banking, medical, and telecommunications IS Departments have put the power of the ***Information Systems Series*** to work.



FREE Demo Disk!

See for yourself what the ***Information Systems Series*** can do for *you.* Call for your free Demo Disk right now!

Isn't it worth a few moments of your time to prove to yourself that the ***Information Systems Series*** can save you endless hours of work and many thousands of dollars? Don't miss this no-obligation offer. Call right now!

Call toll-free 1-800-777-0970

The Computer Resources Group, Inc.
303 Sacramento Street
San Francisco, CA 94111

CIRCLE #69 on Reader Service Card ▲

have the same impact as five dollars disallowed on an employee's expense report.

Making Funny Money Real Money

Data center expenses must be raised to the same status as other company expenses. People accept the fact that real money is given to the airline and that their budget is rightfully charged. People care about budget money. They feel a responsibility to watch it carefully. They may even get some real rewards in the form of favorable performance reviews and bonuses when they control it carefully.

How do you accomplish such a drastic change in people's minds? It is surprisingly easy. All you need to do is to increase each department's budget to allow for data center usage and to charge actual usage against the budget. Then running a $200 job will have the same effect as buying a $200 airline ticket and will merit the same caution.

The Ultimate System Tuning Tool

With the cooperation of motivated users, the chargeback system becomes the ultimate system tuning tool. Users start to scrutinize their own consumption of every data center resource. They delete unneeded datasets, stop running unnecessary jobs and schedule other jobs for overnight or weekend runs.

Without this cooperation, the systems programmer must rely on traditional system tuning techniques, purchased software products and hardware upgrades in order to effectively process the workloads.

The users can do a better job. You can employ techniques to optimize jobs but users can eliminate jobs completely.

A chargeback system truly is the ultimate system tuning tool because of the following reasons:

- The results are better
- One tool deals with every aspect of the data center
- Best of all you do not have to do the work — it is done by the entire user community.

Look at a few important areas. You can extrapolate this same logic to many other areas too. In each one, I will discuss the behavior of unmotivated users, the improvements data centers can make without user cooperation, and I will try to imagine the possible savings that might be obtained if the users could be educated and motivated.

Savings On Jobs

First consider jobs. Users often submit jobs without thinking of the cost or of the demands being placed on the system. Certainly they will submit a $500 job without considering whether it is worth $500, perhaps not even knowing that it costs $500. Sometimes people will even resubmit a job just because the listing was lost without waiting an hour or two to give the listing a chance to turn up. Some reports are produced daily, weekly and monthly *just in case*. Many jobs that could be run in the evening or the weekend are run during the day.

Before going on, I must clarify something. I am not saying, for example, that users will not be allowed to resubmit a job when a listing is lost. I am only trying to get them to think about it first. They may decide to resubmit it anyway but at least they considered the alternatives and made a conscious decision. A programmer's time is expensive. However, so is the job about to be submitted. Say the job costs $500. How long will the programmer have to sit idle until $500 is wasted? What if that listing turns up in two hours? What if another user (who is also newly conscientious) finds the missing listing at the end of another job and gives the programmer a call? What if the programmer spent a little effort trying to locate the listing by examining the system log? Or the programmer can do something else for two hours.

> ***Savings are twofold — a lighter demand at peak periods and fewer total jobs.***

What can you do? You cannot judge whether a job is necessary. You can only assume that it is and provide the resources necessary to run it with the least impact on the other work being done.

You can upgrade the hardware. You can buy more real storage to reduce paging or buy larger CPUs. You can always throw more money at a problem but we all agree that this is not the best answer.

You can buy software products that optimize disk accesses, program loading and so on. These are good products. They do what they say they do and your system will work more efficiently. However, these products must assume that every disk access or program load is necessary. All they can do is make the operations as smooth and as quick as possible. They cannot eliminate unneeded ones.

You can tune the worst programs. First, you can produce listings of the resources consumed by different programs in order to isolate the worst ones. These are the best opportunities for improvement. Then you can run other products that tell where these programs are spending most of their time in order to isolate the worst of the worst. Finally, you can rewrite these segments more efficiently. However, the price is high for a relatively modest gain. What would you consider a good return on this investment? Would a 25 percent reduction in CPU be good? Then if you would attack a program that uses an hour of CPU per day, you will only save 15 minutes of CPU time. What if it turns out that the program produces a report that the end user only references occasionally? Such a program could be eliminated completely.

Users can do a better job. Where the data center can only optimize, the user can absolutely eliminate. Some regular reports would be run only when needed and some of them would then never be run at all. There would be a reason to schedule big jobs for the evening and the weekend.

The savings are two-fold. First, there will be a lighter demand at peak periods. Second, there will be fewer jobs in total. This means better on-line response times when most needed. You may be able to postpone a CPU upgrade.

Savings On Disk Space

Disk space is like closet space. What you have is soon used up. When you buy more, it is quickly used up too. You know there is a lot of space that can be freed up; however, you are frustrated when you try to find things to discard.

What can you do? You can buy more DASD. You might be surprised to see how much DASD you already have. DASD expands slowly, so nobody is alarmed when another unit is installed. You might have more money invested in DASD than in your CPU.

You do not know which datasets are unnecessary, so you try to find strategies or heuristics to identify datasets that are unnecessary with a reasonably high probability of accuracy. In other words, how would you go about cleaning someone else's closet?

Many people run HSM (also known as DFHSM by IBM) or a similar product from another vendor. HSM finds datasets that have not been accessed in a month or so, compresses them and moves them to other DASD. After another month or so of inactivity, it moves the datasets to tape. This is fairly effective as a global strategy but is not perfect for many reasons.

- You have to wait 30 days for datasets to age sufficiently.
- Some datasets are moved to tape that are actually needed and must be moved back to disk.
- These little surprises annoy everybody. They are an interruption to the productive work flow. Some programmers even write a short TSO CLIST that accesses each of their personal datasets so none of them will ever disappear again. Imagine, you have actually alienated your users, not won their cooperation.
- Some of the savings on hardware are offset by the expense of the software product.

Users are better at deciding which datasets are necessary and they will take the trouble if there is a price associated with each one. Some datasets are program outputs. These can be deleted because they can be regenerated easily if needed. Some others were kept *just in case*. These can be moved to tape. There may also be some mysterious datasets whose purpose no one knows or remembers. Some users even keep large decoy datasets, so they know space will be there when they need it. This practice will stop when it becomes too expensive and when a surplus of space makes it unnecessary.

How much can be saved? That depends on how many unneeded datasets exist and how expensive it is to keep them. Can you save 10 percent or 25 percent from each volume? Then entire volumes will be freed. It could be a long time before you need a new volume.

Savings

Should you buy a new graphics or statistics package or a new sort package? The answer is no for the graphics or statistics packages unless the requesting users are willing to pay. If it is worth the money to the end user, then it is worth the money. What if the user claims that the package is so marvelous that everyone else will start using it once it is installed? Then ask if that user is willing to underwrite that risk. If he is right and everyone starts using the package, then it will be inexpensive per use. However, if he is wrong and he is the only one to use it, is it still worth that much money to him?

The answer for the sort question is harder. Here, no individual user is asking for the package. Rather, data center management is evaluating the package for the use of all. Of course, the new package will sort faster with less resources consumed. However, will it save enough to offset its cost?

Just one refused product can balance out the cost of the chargeback system.

Why not let the chargeback system solve this question for you? If you buy the new sort you will be passing its cost on to the users, so why not let them participate in the decision?

Allow users to use either the old free sort or the new expensive sort. If they use the new expensive sort, they pay a fee per use. In any event, they pay for the CPU and I/O that are used during the sort.

For large sorts, it makes sense to use the new sort. The savings on CPU and I/O is so much greater than the usage fee that the profit makes the new sort the only choice; however, it still makes sense to use the old package on small sorts. The small total CPU and I/O used by the old sort is a limit to how much the new sort can possibly save. The amount that can be saved will be less than the usage fee.

As the motivated users get involved and make their choices, fewer and fewer will choose the new sort and the usage fee will climb; then some more users will opt for the free sort and so on. Just as water seeks its own level, this formula will stabilize in the new free marketplace.

Also this logic works on specialized hardware devices like plotters and scanners. If you pass the cost on to those who use them, some will decide that they do not need them especially if you give them cheaper alternatives.

How much can you save? Of course, that depends on many factors. The opportunity to save is larger in avoiding future products than in trying to remove older, entrenched products. However, even just one returned or refused product can balance out the cost of the chargeback system itself.

Et Cetera

You can apply this logic to many other areas, too. How much can you save on printing if users consider the cost and their need for large reports? Do COBOL programmers always need their procedure and data maps? Does a dump have to be printed? Can a big reference report be replaced by an on-line inquiry?

How much can you save on tapes? How many tapes are in your library that are not needed anymore? How many files should really be on disk?

How about disk accesses? What if there were a penalty for inefficient blocking or for excessive arm thrashing due to poor dataset placement?

The list can go on forever. However, the essential ingredient is user participation. None of these savings can be realized without it.

The Downside

The disadvantages of this theory are obvious. First, there will be more management. Managers will have to estimate their consumption, watch their consumption and talk to subordinates who are wasteful. Managers already do these tasks in other areas. Maybe they should monitor data center usage too. It is a big enough expense to warrant the attention.

Second, there will be more resource management by individuals. Programmers will now have to $pend some time evaluating the need for each dataset. Users will have to reevaluate the need for each

GIANT

IS NOW ON LINE

The Global Information Access Network

Making decisions about hardware, software and data processing services is a difficult task. Just trying to identify the available products can be an enormous dilemma.

The time has come to benefit from the same technology you work with day-to-day. Explore the numerous products and services available from the comfort of your office. GIANT provides a complete data base at your fingertips through our dial-up network. And best of all, it's FREE!

Free Access — No User-id Required — No Password — On-line Instructions.

Dial-up access is available from your PC. Use any of the popular communications packages that emulate a VT100 terminal (or use a VT100 terminal). We have successfully tested using PROCOMM Plus™, CROSSTALK™ and KERMIT.

2400 → (414) 546-8480
1200 → (414) 546-8492
Terminal type VT100
Line speed . . 2400 or 1200
Parity. . . . . None
Data bits 8
Stop bits 2

Dial-up is available Monday through Saturday, 24 hours a day. (BISYNC and SNA dial-up support coming soon)

If a dial-up terminal is not available, then FAX your request to us. We will search our data base and FAX a list of products or services to you.

(414) 546-8499 (FAX)

FAX is available 7 days a week, 24 hours a day.

Or you may call us direct, and we will search our data base immediately to find the products and services you need.

(414) 546-8496
(Voice Lines for Personal Service)

Personal assistance is available Monday through Thursday from 9:00 AM to 1:00 PM Central Time.

Software Manufacturers!! Hardware Manufacturers!! Service Suppliers!!

Advertise your products and services on GIANT today. Put your message into every data center across the country . . . everyday. Join with GIANT and become part of the GIANT team: we're making the marketing media of the future a reality today!

Give us a call and we will send our GIANT information kit to you immediately.

PSI

Productivity Systems, Inc.
P.O. Box 20958
Milwaukee, WI 53220
414/321-8688

PROCOMM Plus is a registered trademark of DataStorm Technologies, Inc. CROSSTALK is a registered trademark of Data Communications Associates, Inc.

CIRCLE #135 on Reader Service Card ▲

If you have article ideas, comments or suggestions concerning MAINFRAME JOURNAL, write or call: Bob Thomas, editor-in-chief, MAINFRAME JOURNAL, PO Box 551628, Dallas, TX 75355-1628, (214) 343-3717.

report. They will argue that when they do this they are not doing their primary job. However, these users should not be the ones to decide how they should spend this time. Will they earn more for the company in their prime job or will they save more for the company in this new activity? People must realize that this is now a part of their primary job.

Third, no one will like the new system. People will think they are being told not to run their jobs. Actually, they can run the jobs they need and they have been given an adequate budget to do so. All we ask is that they consider the cost of the job. Is it worth it? Is it necessary? Can it wait until the evening or the weekend?

The Upside To The Downside

There is even an upside to the downside. The capacity planners will finally get the information they need.

Capacity planning is a discipline in which systems programmers try to predict the company's future needs for data processing power. Part of this prediction is based on statistically trending historical data and extrapolating into the future.

However, it is generally agreed that the best trending techniques are invalid without also interviewing the users and blending in the users' plans. Now the users' plans will be known as part of the budget planning process. They will be known in terms that both users and programmers understand, that is, dollars and cents, not CPU seconds and EXCP counts.

Conclusion

The bottom line is whether this new system with all these problems is worth the trouble. The users will resist the change and will complain about the increased bureaucracy.

On the other hand, the amount of money that can be saved is enormous. It is time that data center expenses are treated with the same respect as other expenses. It is time to play for keeps.

ABOUT THE AUTHOR

Kenneth R. Whitaker is a product development manager for LEGENT Corporation. He is the author of BDBF, a comprehensive data center accountability and chargeback system. Also, he speaks on chargeback topics worldwide, most recently at CMG Australia. LEGENT Corporation, Two Allegheny Center, Pittsburgh, PA 15212, (412) 323-2600.

I • B • M Conquers Space

Software Takes Advantage of ESA's Facilities

By Michael Haupt

MVS/ESA provides two new facilities for handling data during the processing of programs. Data spaces are virtual storage structures similar to address spaces but dedicated to storing data. Hiperspaces (high performance data spaces) take advantage of expanded storage, reducing delays associated with disk I/O.

Previously in "Through the Data Window: A Look at Data Spaces and Hiperspaces" *(MAINFRAME JOURNAL,* March 1989), how applications can directly use data spaces and hiperspaces was examined. As was noted, it takes some effort for applications to take full advantages of these facilities.

However, these facilities need not lie dormant until your installation has the manpower to evaluate and integrate them gradually into your applications. Many IBM strategic software products already capitalize on ESA's new capabilities.

MVS/ESA's new Virtual Lookaside Facility (VLF) allows data spaces to be used as high speed libraries, supplementing partitioned datasets. VLF is used by three of MVS' most active components: the Library Lookaside Facility, ICF Catalogs and TSO/E command processing. Other system software takes advantage of ESA's facilities such as Advanced Address Space Facility (AASF), data space and hiperspace.

Virtual Lookaside Facility

The Virtual Lookaside Facility (VLF) is a new set of services which can improve performance by using a data space to reduce disk I/O. Essentially, VLF is an intelligent buffer or cache manager. It stores and retrieves named data objects from a data space.

VLF Advantages

Traditional techniques also reduce I/O by keeping some data in virtual or real storage. These techniques are normally oriented to characteristics of the physical device. Access method buffers are based on the physical record or block size. DASD cache controllers store entire tracks of data. VLF instead stores and retrieves entire objects such as PDS members.

Traditional buffers or cache contents are controlled by a simple and inflexible algorithm. Usually, incoming data overlays the least recently used data in storage. VLF has the capability to keep track of how the data objects are used.

VLF Processing

VLF runs in its own address space and uses a data space for storage as shown in Figure 1. Whenever a data object such as a PDS member is needed, VLF attempts to supply it from its data space rather than going to the disk.

For example, when a user calls for object "B" from Library 1, VLF provides "B" from its data space without disk access. If the user calls for object "W" from Library 1, VLF retrieves it from the disk and stores it in the data space as well as providing it to the user.

FIGURE 1

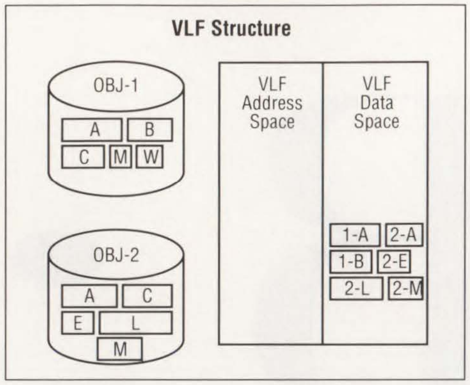
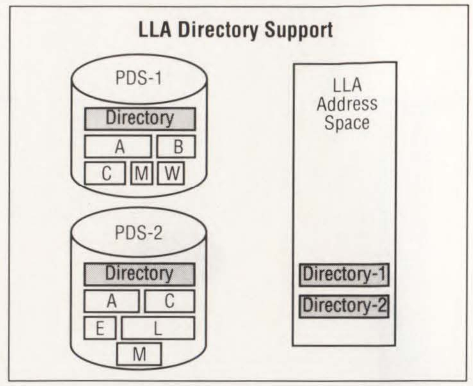


FIGURE 2



Installations can control the amount of resources VLF consumes. The installation specifies the virtual storage size of the VLF data space. As the size of the data space increases, disk I/O is reduced but the demand on paging is also increased.

VLF can be controlled by operator commands. Some commands add or remove libraries from VLF control. Others purge or refresh data objects either specifically or in groups.

VLF PDS Support

PDS members will frequently be the "named data objects" manipulated by VLF. Accordingly, VLF has incorporated special support for PDSes.

VLF correctly handles concatenation of multiple PDSes. VLF's data space can store like-named members from different PDSes. VLF will supply the proper member based on the requestor's concatenation sequence.

The requestor can intermix VLF controlled and non-VLF controlled PDSes within a concatenation. The requestor will always receive the same member, based on the concatenation sequence, whether it resides in VLF or on disk.

VLF also handles PDS updates. Whenever a member in a VLF controlled PDS is updated, VLF automatically purges that member from its data space. With its next use, the updated member will be placed in the data space.

VLF Services

VLF's facilities are available to authorized programs. It will maintain the data objects on behalf of those authorized programs.

The VLF interface is documented. This means that third-party software products, as well as IBM's, will be able to use VLF's facilities.

Library Lookaside

Library Lookaside (LLA) manages the contents of load libraries. LLA reduces the time fetching load modules for execution. LLA under MVS/ESA is a major revision of MVS/XA's LLA (LNKLST Lookaside).

Under XA, LLA maintained the directories for Link List (LNKLST) libraries in its address space. This eliminated directory searches for the LNKLST. The actual load modules still need to be fetched from disk for each execution.

LLA Directory Support

The ESA version of LLA in Figure 2 also executes in its own address space which contains executable code and PDS directories.

The new LLA is not restricted to LNKLST libraries but accommodates directories of any libraries the installation chooses. Operator commands dynamically add or remove libraries from LLA control without the need for an IPL. Other commands refresh selected directories of specific directory entries.

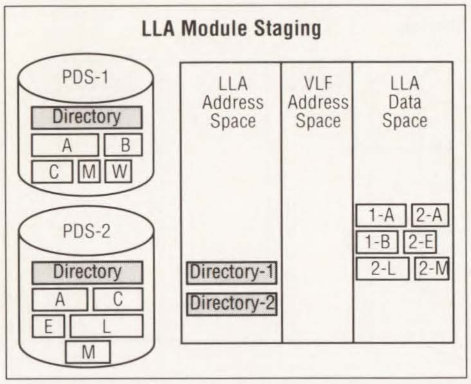
Freezing Directory Versions

LLA can freeze the directory of read-only libraries. Contents of a frozen directory will not be refreshed in the data space. Even if a program in the load library is updated, the directory changes but LLA's version of the frozen directory remains pointing to the original program location.

Freezing the directory shields users from partial or incomplete library updates. Updated programs will not be used until the directory is unfrozen and refreshed.

The ability to selectively freeze directories greatly reduces the interruption to systems when installing new versions. Using frozen directories, production could

FIGURE 3



continue while the program libraries were being updated. Once the update was verified, production would need to be interrupted only long enough to enter the commands to unfreeze and refresh the directory.

Load Module Staging

LLA optionally uses VLF to manage an associated data space to stage the most heavily used load modules (see Figure 3). When they are needed, the load modules can be retrieved from the data space directly by the contents supervisor without program fetch and disk I/O.

Unless the directory is frozen, whenever a program is relinked, LLA automatically refreshes the directory and purges the staged module from the data space.

Staging Dynamics

LLA dynamically controls the modules staged in the data space. LLA continuously tracks information on the usage of all modules under its control. It calculates the anticipated savings of staging each module through VLF. Its calculation considers: frequency of use, difference in program fetch's and VLF's retrieval time, reduction in I/O contention and cost to keep the module in VLF's storage.

LLA's decision to stage a module is not final. As time progresses, the calculations will change. Other modules may become more suitable for staging. LLA exchanges modules keeping the most costly in the data space at all times.

This dynamic adjustment of staged modules is vital for today's installations where the workload mix changes from day-to-day and hour-to-hour.

ICF Catalog Improvements

The DFP portion of MVS/ESA makes two major enhancements to ICF Catalogs. DFP Version 3 improves both the performance and the sharing of ICF Catalogs.

ICF Catalog and VLF

The Catalog Address Space (CAS) has been reorganized. The CAS processes catalog requests similar to the MVS/XA version. Catalog data may either be stored in the address space, as it is with MVS/XA, or kept in a VLF controlled data space.

Use of a data space permits larger amounts of catalog data to be kept in storage, speeding access and reducing catalog contention. The installation designates which catalogs are eligible to take advantage of the hiperspace's performance.

Shared ICF Catalogs

Increasing the amount of catalog data in virtual storage creates complications when a catalog is shared between two systems. Prior to DFP Version 3, changes made in storage on one system were not reflected in the other system. Because of this, shared catalogs were frequently excluded from the CAS. Major performance bottlenecks occurred when the shared catalog was heavily updated by both systems.

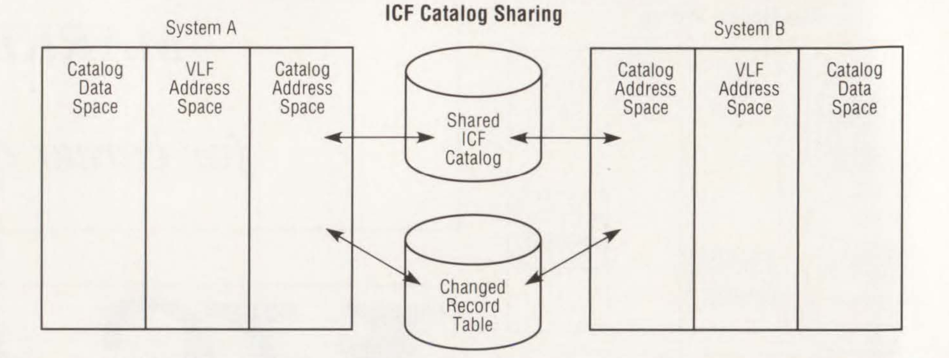
Under the new DFP, ICF Catalogs shared between multiple systems can also gain performance benefits afforded by CAS. Figure 4 illustrates the new facility for catalog sharing.

The ICF Catalog resides on shared DASD where it can be accessed by both systems. Both systems include the catalog in their CAS which uses a VLF data space to keep the catalog data.

A new feature, the Changed Record Table, also resides on shared DASD. When System A changes a catalog record, the CAS writes its key to the Changed Record Table. The CAS on System B reads the key from the Changed Record Table and directs VLF to refresh the applicable catalog data. Using this new feature can substantially reduce I/O to the shared catalog while maintaining the catalog's integrity.

FIGURE 4

ICF Catalog Sharing



TSO/E Version 2

The new version of TSO/E has two significant improvements over Version 1. The use of VLF increases performance and the addition of the REXX language provides new function.

TSO/E With VLF

TSO/E can optionally use VLF to improve response time when executing CLISTs. Active TSO environments consume considerable resources processing CLISTs. Every time a CLIST executes, the system:

- Dynamically allocates SYSPROC
- Executes an OPEN and BLDL macros for SYSPROC
- Creates control blocks for the CLIST
- Reads the CLIST into storage
- Executes a CLOSE macro for SYSPROC.

Version 2 of TSO/E optionally takes advantage of VLF. CLISTs can be preprocessed and cached in the VLF data space.

When a user executes a CLIST, VLF moves it from the data space into the user's address space. Not only is VLF faster than disk I/O, it avoids the overhead associated with SYSPROC dataset processing.

The use of VLF to cache CLISTs is entirely optional. The installation specifies through a PARMLIB member (COFVLFxx) the names of CLIST datasets to be cached by VLF. When VLF is started, CLISTs in those datasets are preprocessed and cached in the data space. Normal CLIST processing takes place for CLISTs which are not cached by VLF or whenever VLF is not active.

REXX Language Support

The other significant improvement in TSO/E is the inclusion of the REXX language. VM/CMS users have had Restructured EXec eXecutor (REXX) available for a number of years.

REXX is similar to, but more powerful than, the CLIST language. Besides its power, REXX is clear, well-structured and easy-to-learn. For these reasons, REXX is the procedural language selected for SAA.

REXX is flexible. It is simple enough for non-DP users to create their own macros. REXX is also a full-function, structured, interactive language in which DP professionals can develop entire applications. Finally, REXX's power makes it more than adequate for many systems programming tasks.

REXX programs, called EXECs, are interchangeable with CLISTs. An EXEC can be:

- Invoked as a command from the READY prompt
- An ISPF dialog or edit macro
- A subcommand to other TSO/E commands (like EDIT or TEST)
- Executed from a batch Terminal Monitor Program (TMP)
- Called from another CLIST or EXEC
- Called from a program via the TSO Service Facility.

REXX EXECs can be cached along with CLISTs in a VLF data space.

Under TSO/E, REXX is a superset of the VM/CMS language. It provides all the functions found in the VM/CMS version. In addition, it supplies other functions to accommodate MVS and provide CLIST compatibility.

Further Use of ESA Facilities

ESA is IBM's strategic operating system environment for large mainframe installations. Its facilities are well known to other IBM product groups developing software for the same arena. A sampling of other products which build on ESA's base follows.

VPS

Tames The Jungle of MVS Network Printer Support

VPS has been used to replace other products, such as: IBM's 328x/ADMPRINT/DSPrint, CICS supported printing, SASWTR®, RJE and many others, with a single task to drive all your 3270 family printers directly from the JES spool (including cross domain VTAM printers).

1700 Sites use VPS as their shop standard 3270 family printer driver.

- ✔ Printers supported include the full array of 3174/3274 attached printers, including IPDS support for 42x4's, HP and Xerox lasers, plotters and PC printers.
- ✔ VPS runs as a VTAM application. NO system modifications. NO JES maintenance.
- ✔ Automatic forms control, full FCB support, dial-up PC printers, printer pooling and "hot" printers are all supported.
- ✔ Full screen "ISPF-like" command interfaces for CICS, TSO and ROSCOE® permit end-user control of printers with totally menu driven command entry.

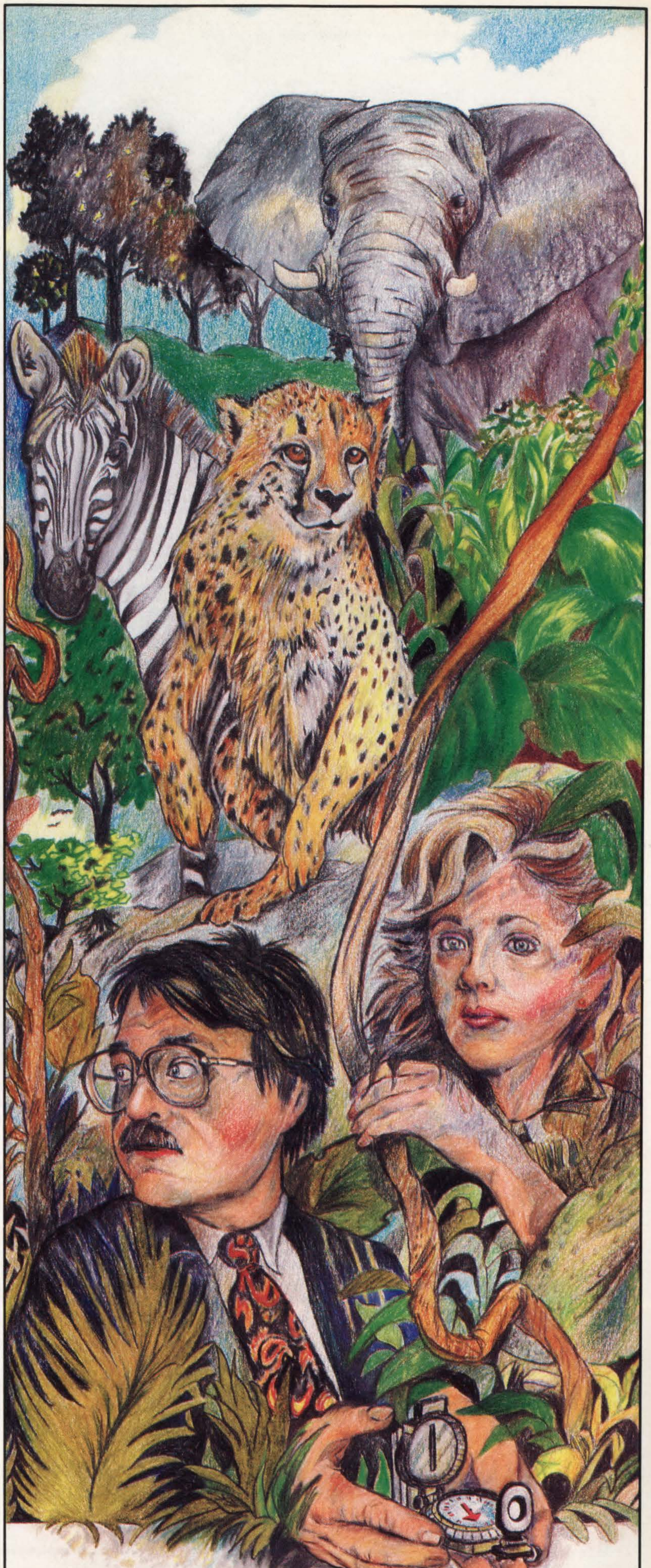
Call or write for more information, or to arrange for a FREE TRIAL

ATTENTION: Marketing Department

LRS *Levi, Ray & Shoup, Inc.*

Specializing in Computer Systems Software

2387 West Monroe
Springfield, Illinois 62704
(217) 793-3800 ◆ Fax (217) 787-3286



JES

JES2 3.1.1 takes advantage of ESA in several ways. It places the internal readers' buffers in a space, permitting a dramatic increase in the number of internal readers (from 255 to 32K). This gives a performance boost to TSO environments frequently submitting background jobs.

JES2 exploits ESA with Functional Subsystems (FSS). It uses AASF for cross-memory communication to an FSS address space. Enhancements to the Program Call (PC) instruction allow JES2's recovery routines to protect code in the FSS address space.

JES3 3.1.1 supports the use of a data space to store the Job Control Table (JCT).

DB2

DB2's structure requires multiple address spaces to operate. Version 2 exploits AASF reducing MVS' overhead for communication between the address spaces.

IMS

IMS/VS permits the use of LLA to manage programs. Using LLA instead of preloaded programs makes more virtual storage available in the Message Processing Region. As an alternative to Virtual Fetch, LLA reduces I/O delays during fetch processing.

IMS/VS 2.2 also permits the use of hiperspaces to provide large VSAM buffer pools in expanded storage. New IMS products have been announced which will only operate in an ESA environment.

CICS

Release 1.7 has limited use of ESA. Hiperspaces are only available for DL/1 buffer pools. CICS/MVS 2.1 supports both DL/1 and VSAM buffer pools in hiperspaces. IBM has also made a statement of direction indicating a forthcoming version of CICS will only run under MVS/ESA.

DFSORT

Release 11 of DFSORT promises improved performance through hipersorting, the use of hiperspaces during sort processing.

ImagePlus

ImagePlus is IBM's system designed to store and retrieve images. Documents are scanned into the system, stored on optical or magnetic disk, retrieved and routed to be displayed or printed.

ImagePlus consists of three separate software components and all three require MVS/ESA. ImagePlus is a bellwether of IBM's software trends. In the future you will see more "leading edge" software from IBM built on the ESA base.

Conclusion

A host of products from IBM that capitalize on ESA's facilities to deliver performance improvements with minimal effort have been examined. The use of data spaces and hiperspaces reduce disk I/O, improving response time and overall performance.

The lower I/O rate, in turn, gives flexibility in managing disk space management. It permits active datasets to be concentrated on fewer volumes without I/O contention degrading performance. It eliminates the need for the deliberate underutilization of disk capacity to avoid I/O activity bottlenecks.

As with all performance issues, eliminating one bottleneck creates another. The amount of storage allocated to a data or hiperspace must be controlled. I/O savings need to be balanced against storage and paging costs..

The availability of ample real and expanded storage is a prerequisite. If it is not plentiful, data spaces and hiperspaces might even degrade performance. Paging delays must be minimized through the use of cache controllers, high speed channels and short queues.

MVS' trend is to spread the processing between multiple address spaces as with DB2. This reduces response time by permitting concurrent processing of the address spaces. Concurrent processing can only occur in multiprocessor complexes. Uniprocessors will be at an increasing disadvantage as this trend continues.

As you can see, ESA offers enormous benefits but not without some costs. How soon ESA is in your future depends on when your business can profit from its new features.

ABOUT THE AUTHOR

Michael Haupt, CDP, is a senior associate with Austin Consulting in Rosemont, IL. During his 15 years in data processing, he has worked extensively with large scale IBM systems and has been active in professional associations and user groups. Austin Consulting, 9801 W. Higgins Rd., Rosemont, IL 60018, (312) 696-0500.

DB2 MANAGER
TAKES THE WRAPS OFF DB2 PERFORMANCE.

When you start using DB2 as a real production tool, you need more than a little insight into the performance of DB2 internals. You need to see what's happening in and around your DB2 subsystem. And for that, you need DB2 MANAGER. It's the only online DB2 performance tool you can integrate with a full range of performance software to provide complete DB2 performance management.

You'll see the impact of entire transactions, from initiation to completion. From CICS and IMS to DB2 and back. So you can easily identify the real cause of problems.

You can also manage multiple, even remote, DB2 systems from one terminal. For the really big picture, you can manage CICS and IMS subsystems in a single session. That's a performance picture you can't get anywhere else.

For further information on DB2 MANAGER, call Marty Johnson today. In California: 800-624-5566. Outside California: 800-822-6653.

Boole & Babbage, Inc. 510 Oakmead Parkway, Sunnyvale, CA 94086.

Boole & Babbage

International sales and support provided through The European Software Company, a member of the Boole & Babbage worldwide network of companies.

IMS Fast Path Data Entry Database Tuning

By Gary Rygmyr

The purpose of this article is to document the effects of different reorganization techniques and DBD definitions on record distribution in a Fast Path Data Entry Database (DEDB). The information presented here could be helpful to DEDB users or potential DEDB users in determining from an actual production example the performance and storage characteristics of a DEDB. Also, the IBM DB tools utilities are discussed in the course of the article and could serve as a review for those already familiar with DEBD. It is important to note that the standard IBM randomizer, DBFHDC40, was used in all tests. The data in this article would probably not apply if a different randomizer were used, especially if it were area specific.

The Database

The database, DBAFPTST, is composed of three levels. There are no Sequential Dependent (SDEP) type segments. Tests One through Three had the following segment distribution:

FPTSEG01 :	85,346
FPTSEG02 :	143,879
FPTSEG03 :	2,166,477

Tests Four through Twelve had the following segment distribution that represents the change to the database during one week of production processing:

FPTSEG01 :	85,669
FPTSEG02 :	145,016
FPTSEG03 :	1,993,864

It has been in production for more than three years and was recently expanded from four areas to eight to hold more data. Three transactions access this database for read only. It is updated by BMP jobs nightly.

A regular reorganization was not being run. Records are distributed evenly among all eight areas as would be expected using the IBM delivered generalized randomizer. Other DEDBs with two, three and four areas show the same type of record distribution with this randomizer.

Procedures

The overall objective was to maximize the number of records in the Root Addressable Area (RAA) and minimize the number of records in Dependent Overflow (DOVF) and Independent Overflow (IOVF). Records in the overflow areas tend to increase the number of I/Os required to access these records.

In these tests the randomizer, DBFHDC40, the CI Size and space allocation were not changed. The focus was to evaluate the effect of reorganization and values for the UOW and ROOT DBD parameters on record distribution.

A test database was created with only one area but in every other way exactly like the production version with eight areas. This allowed the reorganization of only one area. The DB Tools Unload/Reload Utility will not allow you to unload only one area of a multi-area DBD when changing the DBD if a generalized randomizer is used. Each reorganization was followed by a DB Tools Pointer Checker. Tables 1 and 2 summarize the Pointer Checker data for each test.

Two different strategies were tested. The first strategy was to try various reorganization techniques without changing the DBD. Table 1 shows the results of these tests. Test 1 is the initial profile of the database. The second strategy was to adjust the UOW and ROOT parameters in the DBD for optimum DASD utilization and I/O performance. The objective was to shift space allocated to the overflow areas to the RAA since so little of the overflow areas were actually being used. Table 2 shows the results of these tests. A discussion of each test follows.

Discussion

Test One

The initial profile. It appeared that there were far too many segments in both DOVF and IOVF.

Test Two

Reorganization was accomplished using the DEDB Direct Reorganization Utility (DBFUMDR0). This is one of the DEDB on-line utilities that allows concurrent on-line application processing. It runs in the DB dependent type region. A CI is held exclusively while being reorganized. This may affect application per-

INTERNATIONAL
DB2 USERS GROUP

If You Attend One Conference This Year Make It IDUG

Are you looking for answers to questions that only other DB2 users can understand? Looking for a group that focuses on the distinct nuances and intricacies of IBM's DB2 product?

Get involved in the International DB2 Users Group—the only user group dedicated solely to the DB2 community.

Existing user groups touch on DB2 issues in their programs, but no international group addresses all the concerns of DB2 users—until now. You're invited to:

- Network with peer users in your field
- Learn from noted industry experts
- View exhibits, hospitality suites and product sessions from participating DB2 vendor members
- Receive complimentary user membership in IDUG throughout the year by attending the conference.

A sampling of speakers scheduled include: Dr. Carma McClure, Extended Intelligence Corp.; Colin J. White, *InfoDB* magazine; William Inmon, American Management Systems; Jeffrey B. Tash, Database Decisions; and many more.

THE DB2 EVENT OF 1989
"The Impact of DB2 on Information Processing in the 1990s"
August 6-9
Hyatt Regency Chicago

For more information contact:
IDUG
111 E. Wacker Dr., Suite 600
Chicago, IL USA 60601
(312) 644-6610
FAX (312) 565-4658
Telex 62896249

Return this coupon for more information on user membership in IDUG and "The DB2 Event of 1989" to: IDUG, 111 E. Wacker Drive, Suite 600, Chicago, IL USA, 60601; (312) 644-6610;
FAX (312) 565-4658; Telex 62896249

Name ______________________________

Title ______________________________

Company ______________________________

Address ______________________________

Phone ______________________________

☐ Vendors interested in membership, exhibiting, hospitality suites and/or product sessions at the conference should check the box and return this coupon.

CIRCLE #173 on Reader Service Card ▲

formance as no other program can access this CI until the utility is finished with it. This run was done during normal on-line activity and ran a little more than four hours. For a comparison, the same reorganization was run during no on-line use and low overall system activity and it still took more than two hours. This reorganization produced a significant change in record distribution and reduced the calculated Average Record I/O by about 25 percent.

Test Three

Reorganization was accomplished using the DB Tools Unload/Reload Utility with the REORG control statement. For this reorganization the database must be taken off-line and the DBDS must be deleted and defined. The advantage here is that the job runs in about 20 minutes. It also produces a segment count report that the on-line reorganization does not. If any DBD changes are to be made, this is the method to use as changes cannot be implemented with the on-line reorganization. A user-written program can be used to reload the database as well. This reorganization produced further change in record distribution and calculated average I/O.

Test Four

Reorganization was accomplished using the DB Tools Unload/Reload Utility with the REORG and LOADCTL control statements. The LOADCTL control statement specifies an Insert Limit Count (ILC) for a given dependent segment. This provides for controlling how long twin chains are loaded. The LOADCTL statement was LOADCTL=(FPTSEG03,25). In this case, reloading any occurrence of the third level segment, FPTSEG03, after 25 occurrences, will be loaded after all the other database records for the current RAP are processed. Again, there was a shift in record distribution that should improve performance. Note the slight difference in record count. This was the result of one week of normal production processing. This data was used for the remainder of the tests.

Test Five

DB Tools Unload/Reload using LOADCTL=(FPTSEG03,20). This reorganization was done to see if any further improvement could be made by further reducing the length of the FPTSEG03 twin chains. Still more records moved to the RAA base area but not nearly as significant as the previous runs. There is a warning in the Unload/Reload User's Guide stating that specifying a low value for the ILC can seriously impact IMS/VS access to the database and that a good rule of thumb is that no more than 10 percent of the database records being processed should exceed the ILC value.

Test Six

DB Tools Unload/Reload using RMODTYPE=G control statement option on unload and no LOADCTL card. Using this option will allow changes to the database as the randomizer is called for each root segment. The results here were similar to test three which would be expected.

Test Seven

UOW=(13,3),ROOT=(5300,500)

Here the DOVF was reduced by one CI. Since 77 percent of DOVF is freespace, it would be better to shift some of this space to the RAA Base. The result was that indeed more records were shifted to the RAA base.

Test Eight

UOW=(13,2),ROOT=(5300,500)

Since there was still 72 percent freespace in DOVF, another CI was taken from DOVF and placed in the RAA base. Again, more records were shifted to the RAA base and average I/O decreased.

TABLE 1

DEDB Tuning Statistics Profile

	TEST 1	TEST 2	TEST 3	TEST 4	TEST 5	TEST 6
DBD Information — area ALLOC in CYL						
UOW	13,4	13,4	13,4	13,4	13,4	13,4
ROOT	5300,500	5300,500	5300,500	5300,500	5300,500	5300,500
ALLOC: DDAHIS1A	460	460	460	460	460	460
FREESPACE ANALYSIS — values are % freespace						
RAA BASE	58	55	55	56	56	56
DOVF	68	70	71	77	77	77
IOVF	87	98	98	99	99	99
OVERFLOW USAGE						
UOW Using DOVF	4,591	4,322	4,282	4,069	4,069	4,069
UOW Using IOVF	669	128	100	45	45	46
DB Record Profile Analysis						
NO. DB RECORDS	85,346	85,346	85,346	85,669	85,669	85,669
AVG REC LENGTH	1,191	1,191	1,191	1,096	1,096	1,096
STD. DEV.	1,470	1,470	1,470	1,342	1,342	1,342
MAX REC LENGTH	14,049	14,049	14,049	17,334	17,334	17,334
MIN REC LENGTH	22	22	22	22	22	22
RECORD PLACEMENT ANALYSIS — values are number of records						
BASE ONLY	55,754	63,448	69,437	70,361	70,561	71,047
BASE + DOVF	23,398	14,597	9,223	12,867	13,763	8,020
BASE + DOVF + IOVF	3,979	343	38	46	49	18
BASE + IOVF	1,167	350	50	50	56	25
DOVF ONLY	782	6,254	6,458	2,306	1,215	6,482
DOVF + IOVF	218	266	61	27	19	26
IOVF ONLY	12	88	79	12	6	51
SEGMENT I/O ANALYSIS — average DB record						
AVG RECORD I/O	1.20	1.18	1.21	1.17	1.16	1.17
AVG ROOT I/O	1.01	1.09	1.09	1.03	1.03	1.09
SYNONYM CHAIN STATISTICS						
LONGEST CHAIN	10	10	10	10	10	10
AVG SYN CHAIN	2.91	2.91	2.91	2.91	2.91	2.91
AVG ROOTS/RAP	2.29	2.29	2.29	2.30	2.30	2.30

Effects of different reorganization techniques.

Succeed

PLATINUM *technology, inc.*
The DB2 Company™

PLATINUM *technology, inc.* is the only full service software company specializing exclusively in DB2. All our effort and energy is focused on delivering the broadest array of quality solutions for the DB2 user. As a result, IBM® has designated PLATINUM a Business Partner through the Authorized Application Specialist program specifically for DB2.

PLATINUM PRODUCTS

PLATINUM offers a complete line of software tools, education, and published products to ensure your success with DB2.

Software Products

The PLATINUM product portfolio consists of a complete family of administration, development, and end user DB2 software products. All are compatible with DB2 V1.3 & V2.1. The software products include:

- **RC/Query**™ - A comprehensive DB2 catalog query tool.
- **RC/Update**™ - The industry's best DB2 object management and data editing tool.
- **RC/Migrator**™ - A complete object and data migration tool.
- **RC/Secure**™ - An extensive DB2 security management tool.
- **PLATINUM Database Analyzer**™ - The DB2 database and DASD analysis, audit, and management tool.
- **PLATINUM Report Facility**™ - The DB2 query and reporting system for developers and end users operating in TSO and CICS environments.

DB2 Education Courses

A complete series of hands-on DB2 training courses. PLATINUM courses cover all aspects of DB2, QMF, and CSP. All courses are available either at your facility or at our Corporate Education Center.

- Introduction to DB2
- DB2/SQL Application Programming
- DB2 Application Planning and Database Design
- DB2 Database and System Administration
- Using DB2 and QMF
- CSP Application Programming

Published Products

The most recognized and authoritative DB2 standards, methods, and guidelines for DB2 implementation.

- **PLATINUM DB2 Guide/Online**™ - The industry's leading standards manual for design, development, and administration of DB2 systems.
- **The PLATINUM Reference**™ **for DB2** - The quick, pocket-sized reference for DB2 information.

Support

All products and services carry our PLATINUM Quality Assurance Guarantee. Support is available 24 hours a day, 7 days a week via our toll-free hot line.

WORLDWIDE AVAILABILITY

PLATINUM's products and services are available around the world through our Affiliate Network. PLATINUM's full service capabilities include local support, education, and superior DB2 professional services.

with DB2!

Test Nine

UOW = (13,1),ROOT = (5300,500)

With the DOVF reduced to one CI per UOW, more records went into IOVF but a warning message, FABC0309W, was issued a number of times. This indicates that all CIs within the IOVF overflow unit were in use. When this happens the reload program sequentially searches the overflow directory entries to find an overflow unit with an available IOVF CI. Since this situation can seriously impact on-line performance, the UOW was changed back to (13,2).

Test Ten

UOW = (13,2),ROOT = (5300,300)

From Test Eight, IOVF had 95 percent freespace. Shifting some of this space to the RAA base again produced a small improvement and still left 93 percent freespace in IOVF.

Test Eleven

UOW = (13,2),ROOT = (5300,100)

A further reduction in IOVF still left plenty of freespace there and produced another small increase in the total number of records in RAA.

Test Twelve

UOW = (10,2),ROOT = (6890,100)

To take advantage of full-track I/O on a 3380 DASD device where there are 10 4K CIs per track, a UOW of 10 CIs was tested. The result was close to the results shown in Tests Ten and Eleven.

Summary and Conclusions

The effects of different reorganization techniques and values for the UOW and ROOT parameters of the DBD were tested to determine if an improvement in database organization could be achieved that would reduce I/O.

As with other types of non-fast path databases, a periodic reorganization can definitely maintain database performance. In this example an on-line reorganization improved the calculated average I/O by 25 percent. A reorganization using the DB Tools Unload/Reload Utility achieved a 43 percent improvement in calculated average I/O. An on-line reorganization may not achieve the same result as an unload/reload and can take many hours to run even when on-line access is minimal to none. Total elapsed time for the unload/reload jobs averaged less than 20 minutes. If the database can be taken off-line for a short period of time, this may be the best way to reorganize. The LOADCTL control statement in the Unload/Reload Utility can have a significant effect on database organization but may not necessarily improve performance significantly.

Different values for the UOW and ROOT parameters clearly affect record distribution and some are definitely better than others. In this case it appears that the final test, Test Twelve, might be the best combination given the constraints of this test. It achieves a modest six percent improvement in record I/O over the value in Test Three as calculated by the DEDB Pointer Checker, full track I/O can be utilized and will most likely cause somewhat less CI contention and contention for fast path buffers. Programs issuing SYNC calls when a GC status code is received will make those SYNC calls a little more often with a smaller UOW, thus freeing up the fast path buffers more often.

TABLE 2

DEDB Tuning Statistics Profile

	TEST 7	TEST 8	TEST 9	TEST 10	TEST 11	TEST 12
DBD Information — area ALLOC in CYL						
UOW	13,3	13,2	13,1	13,2	13,2	10,2
ROOT	5300,500	5300,500	5300,500	5300,300	5300,100	6890,100
ALLOC: DDAHIS1A	460	460	460	460	460	460
FREESPACE ANALYSIS — values are % freespace						
RAA BASE	60	63	67	64	65	64
DOVF	72	63	46	65	68	74
IOVF	98	95	70	93	81	81
OVERFLOW USAGE						
UOW Using DOVF	4,009	3,895	3,310	4,012	4,053	4,740
UOW Using IOVF	153	423	1,811	381	347	289
DB Record Profile Analysis						
NO. DB RECORDS	85,669	85,669	85,669	85,669	85,669	85,669
AVG REC LENGTH	1,096	1,096	1,096	1,096	1,096	1,096
STD. DEV.	1,342	1,342	1,342	1,342	1,342	1,342
MAX REC LENGTH	17,334	17,334	17,334	17,334	17,334	17,334
MIN REC LENGTH	22	22	22	22	22	22
RECORD PLACEMENT ANALYSIS — values are number of records						
BASE ONLY	71,997	73,139	71,047	73,442	73,995	73,461
BASE+DOVF	7,724	7,169	5,454	7,115	6,927	7,308
BASE+DOVF+IOVF	55	149	873	151	127	162
BASE+IOVF	66	217	1,736	193	174	100
DOVF ONLY	5,595	4,387	2,917	4,177	3,959	4,281
DOVF+IOVF	96	267	908	225	212	179
IOVF ONLY	156	341	2,734	366	275	178
SEGMENT I/O ANALYSIS — average DB record						
AVG RECORD I/O	1.20	1.18	1.21	1.17	1.16	1.17
AVG ROOT I/O	1.08	1.06	1.09	1.06	1.06	1.09
SYNONYM CHAIN STATISTICS						
LONGEST CHAIN	10	10	10	10	10	9
AVG SYN CHAIN	2.81	2.71	2.91	2.67	2.63	2.67
AVG ROOTS/RAP	2.16	2.03	2.30	1.97	1.92	1.99

Effects of different UOW and ROOT values.

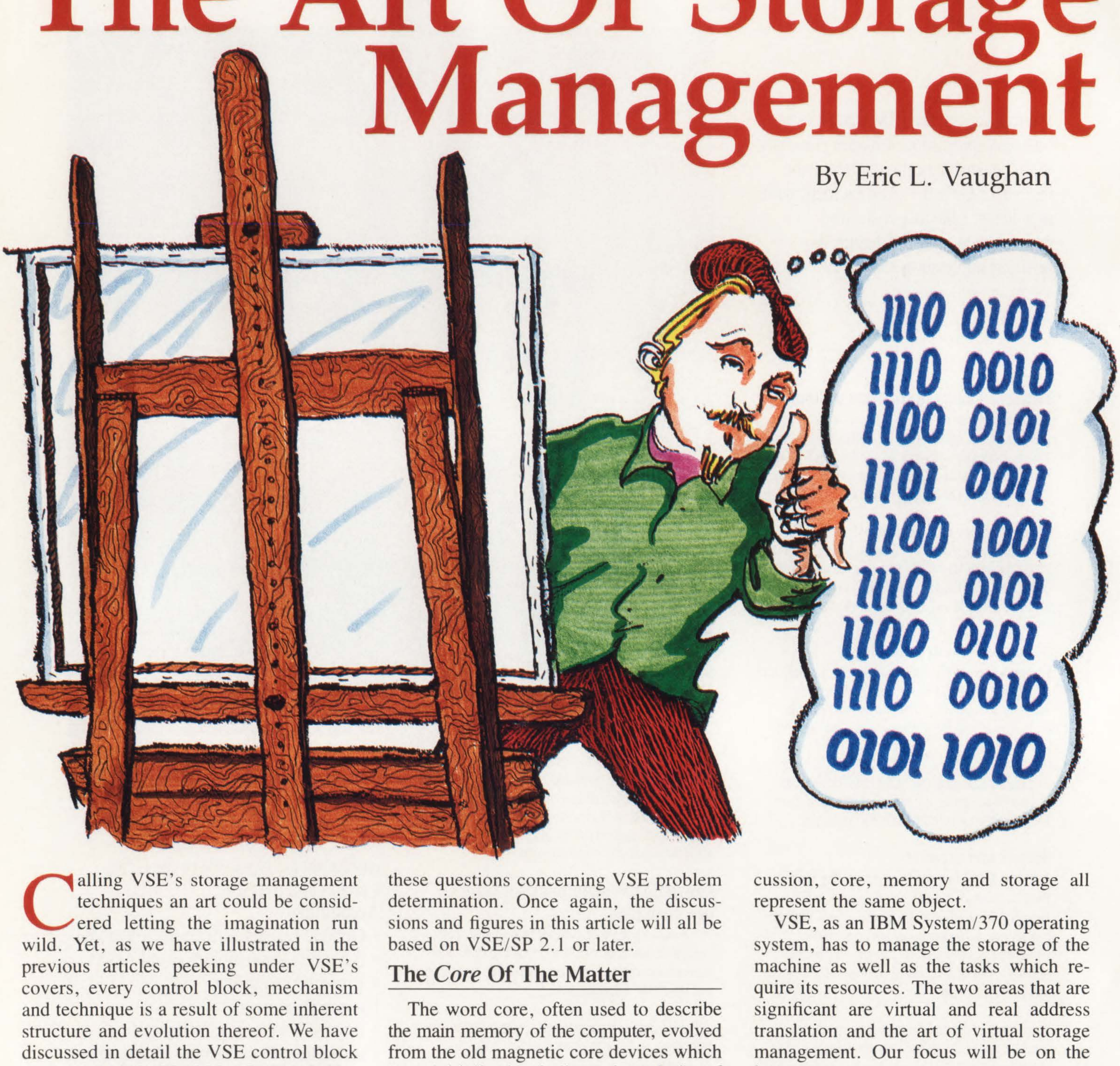
ABOUT THE AUTHOR

Gary Rygmyr's experience includes logical application design, physical design, performance analysis and tuning, disaster recovery and ongoing production recovery support in a DBRC environment. He is presently a senior database analyst for Security Pacific Automation Company, Inc., Networks and Computer Services C2-26, PO Box C34040, Seattle, WA 98124, (206) 431-4840.

VSE Under The Covers

The Art Of Storage Management

By Eric L. Vaughan



Calling VSE's storage management techniques an art could be considered letting the imagination run wild. Yet, as we have illustrated in the previous articles peeking under VSE's covers, every control block, mechanism and technique is a result of some inherent structure and evolution thereof. We have discussed in detail the VSE control block structure (*MAINFRAME JOURNAL* November/December 1988) followed by the heart of the operating system, the Dispatcher, and its mechanism for selecting the task ready to use the system resources (*MAINFRAME JOURNAL* February 1989). However, where do these tasks reside? How is the memory of the mainframe used in the VSE implementation? In this article, I will thoroughly explore these questions concerning VSE problem determination. Once again, the discussions and figures in this article will all be based on VSE/SP 2.1 or later.

The *Core* Of The Matter

The word core, often used to describe the main memory of the computer, evolved from the old magnetic core devices which were initialized to indicate the polarity of a magnetic field, positive or ground. A positive polarity represented a *bi*nary digi*t* or bit with a value of one and a ground polarity represented a bit value of zero. Thus a simple implementation of the base two or binary method of representing data was affected. The equipment properly labeled "core" is antiquated; however, this term still remains. For purposes of discussion, core, memory and storage all represent the same object.

VSE, as an IBM System/370 operating system, has to manage the storage of the machine as well as the tasks which require its resources. The two areas that are significant are virtual and real address translation and the art of virtual storage management. Our focus will be on the latter area.

Fixed Allocations

If you were to let your imagination run wild, imagine a VSE without a fixed number of partitions or even a VSE without fixed address limitations on partition allocations. There have been many suggestions for these types of features over the last several years, yet these attrib-

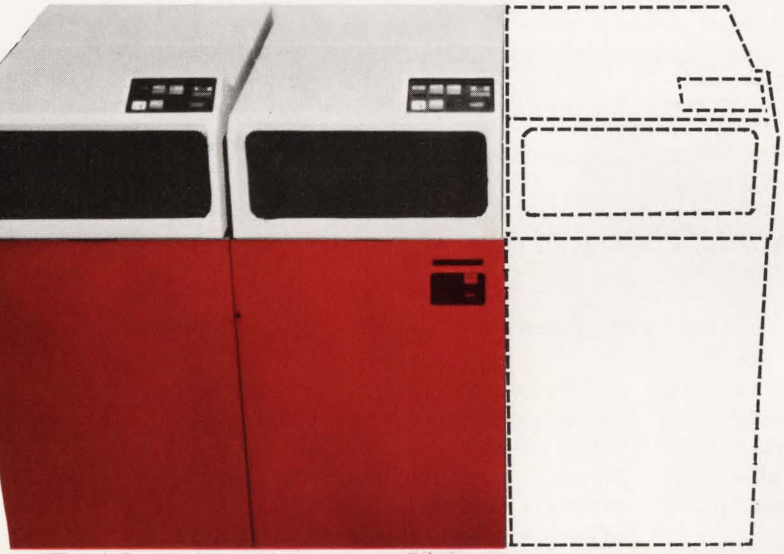


FIGURE 1

VSE Virtual Storage Layout

Partition	Area	Address
	Supervisor Area LTA PTA RTA CRT	<= X'000000'
	SDAIDS area — set by SDSIZE parameter	<= IJBSSBEG
BG	Partition Save Area Problem Program Area Partition Getvis Getvis Anchor Table	<= IJBPPBEG
Fx..	Partition Save Area Problem Program Area Partition Getvis Getvis Anchor Table	<= SMPSAVE + X'78' <= SMVGVIS <= IJBGVCTL
Fx..	Partition Save Area Problem Program Area Partition Getvis Getvis Anchor Table	<= SMPSAVE + X'78' <= SMVGVIS <= IJBGVCTL
	Shared Virtual Area (SVA)	<= SMSVABEG
	System Directory List (SDL)	
	Shared Phase Library	
	System Getvis	<= SMSGVIS <= X'FFFFFF'

utes are not found in the VSE operating system. Reality is a fixed number of areas, called partitions, divided into fixed amounts of storage or allocations which are configured at system initialization time.

All programs, including the supervisor, run in a conceptual single address space called virtual storage. Because of the 24-bit addressing limitation, VSE only addresses a maximum of 16MB of storage *at one time*. The virtual addressability extension or VAE mode of VSE supports *multiple* virtual address spaces, yet only addresses a maximum of 16MB at one time. With the *swift switch* of an address space, however, a different segment of core can be addressed, thus effectively expanding VSE's ability to manage more than 16MB of storage.

The address space is divided into fixed, allocated areas consisting of the supervisor, the partitions and the Shared Virtual Area (SVA) (see Figure 1). This figure assumes a System/370 mode system and shows a full 16MB allocation for simplicity of illustration.

Conceptually, this is the VSE virtual address space design. At low core address X'000000' the heart of the VSE system or the supervisor begins. It extends up to its generated size and consists of the resident routines, system tasks and control blocks necessary to manage the system.

Within the supervisor, there are five routines specifically involved with managing VSE storage:

1. GETVIS (SVC 61)
   — used to allocate dynamic storage in partition or system getvis
2. FREEVIS (SVC 62)
   — used to free the storage acquired by GETVIS

3. CDLOAD (SVC 65)
   — used to load phases into the partition getvis area
4. ALLOCATE (SVC 83)
   — used to allocate or reallocate partition sizes
5. SETLIMIT (SVC 84)
   — used to change partition sizes. This SVC is used by job control to apportion the partition and getvis allocations from a command or size card.

The supervisor area also consists of four separate transient areas for various areas of processing including the Logical Transient Area (LTA) for all $$B transients, the Physical Transient Area (PTA) for all $$A transients, the Recovery Transient Area (RTA) for all $$R transients and the CRT transient area for all $$BOCRT transients. These areas are in function, single threaded areas where various supervisor routines are loaded as needed. If one of the areas is occupied and another task needs to invoke a routine in that area, the supervisor will suspend that task until the area is freed. Both the CRT transient area, as the execution location for the SYSLOG routines, and the LTA, as the execution location for several system modules including some OPEN and CLOSE routines, have created throughput *bottlenecks* for years.

The first byte following the supervisor area is pointed to by the System Communications Region (SYSCOM) +XE'8' or label IJBSSBEG using the SYSCOM DSECT. This area begins the SDAIDS debugging tool reserved area. The size of this area is variable and is adjusted by the IPL command SDSIZE. At SYSCOM +X'20' or label IJBPPBEG using the

FIGURE 2

Partition Save Area

Program or Task Name (8 Bytes)		<= PIBPBEG X'00'
Interrupted PSW to return to		X'08'
General Reg 9	General Reg 10	X'10'
General Reg 11	General Reg 12	X'18'
General Reg 13	General Reg 14	X'20'
General Reg 15	General Reg 0	X'28'
General Reg 1	General Reg 2	X'30'
General Reg 3	General Reg 4	X'38'
General Reg 5	General Reg 6	X'40'
General Reg 7	General Reg 8	X'48'
Job Start Time		X'50'
Floating Point Registers		<= PIBPBEG + X'78

DSECT, the beginning of the *problem program area* can be found.

My Program Is A Problem?

Newcomers to the System/370 environment may struggle at first with the description of the problem program area. No, it truly does not indicate a predefined area in the system for you to run your programs with problems. The concept is one of *problem program* versus *system program* and is controlled by bit 15 of the Program Status Word (PSW). This bit controls, for example, whether or not a program may execute privileged instructions.

The problem program areas begin with the BG partition and continue with the variable number of partitions as specified in the system. Each area has the identical layout as shown in Figure 1. The size of each of the areas, however, will be determined by the ALLOC statements provided during IPL. The beginning of a partition's area can be found using several techniques, for example, by locating its PIB plus X'04' or label PIBPBEG using the MAPPIB DSECT. Job control initializes each partition into three separate areas. The first X'78' bytes of the problem area are reserved for the partition's save area where PSW and register information is stored during interrupt processing (see Figure 2). A common technique to simplify dump analysis with a program is to explicitly start the program X'78' bytes ahead using the START X'78' Assembler control command. This will cause the Assembler generated location counters to align themselves with actual partition storage.

Unless a load address is specified during link-edit, a program is loaded into partition storage starting at PIBPBEG

F I G U R E 3

Storage Management Control Block (SMCB)

HEX DISP	LABEL	DESCRIPTION
0	SMAXPFIX	Partition: PFIX limit in pages System: SVA PFIX limit in pages
2	SMPFIX	Partition: PFIX count in pages System: SVA PFIX count in pages
4	SMPSAVE	Partition: Save area address System: Reserved
8	SMVFLAG	Partition: Getvis area flags x'80' : Setlimit given indicator
8	SMVGVIS SMSGVIS	Partition: Getvis area address System: Getvis area address
C	SMVPBEG SMSVABEG	Virtual partition begin address SVA begin address
10	SMVPEND SMSVAEND	Virtual partition end address + 1 SVA end address + 1
14	SMRPBEG	Real begin address
18	SMRPEND	Real end address + 1

SMCB Address Table:

Address of SVA SMCB	Address of BG SMCB	Address of Fn-1 SMCB	* * *	Address of F1 SMCB
0	4	8	12	4*n

n = Number of partitions or NPART

The pointer to the SMCB address table is found in SYSCOM + X'DC'.

+ X'78'. The beginning of the partition getvis area is determined using several methods. For example, if an explicit SIZE = nK parameter is specified on a JCL EXEC card, the problem program area is allocated based on the value of "n" (provided that the minimum amount of getvis space, 48K, remains after allocation). This causes an invocation of SVC 84 or SETLIMIT as discussed earlier. Various other SIZE = parameters (for example AUTO or phasename) can have different effects on the allocation. If no SIZE card is specified at execution time, the size of getvis defaults to the SIZE = parameter provided at IPL or any subsequent attention routine SIZE = command. A SIZE = parameter on an EXEC JCL statement only *temporarily* alters the partition's storage layout; the default values are reset at end of job.

At the heart of partition storage management is a control block appropriately called the Storage Management Control Block (SMCB) (see Figure 3). The SMCB contains several important pointers such as the pointer to the save area at label SMPSAVE and the pointer to the partition or system getvis area. The SMCB is physically a part of the Partition Control Block (PCB) beginning at +X'50'; however, a partition's SMCB *can* be directly addressed by using the SMCB address table as illustrated in Figure 3. The address table contains the four-byte address of a partition's SMCB in PIK sequence.

Anchors Away . . .

The getvis anchor table is used to manage the dynamic requests for storage within the partition. The pointer to the anchor table can be found by using the partition's communication region (COMREG) +X'C8' or label IJBGVCTL using the MAPCOMR DSECT. In VSE/SP 2.1, getvis is organized using a concept called *subpools*. When issuing a GETVIS macro, it is advisable to specify a subpool ID as an operand for different classes of storage. Each subpool is organized into a 4K block and each request has the potential to occupy a particular subpool. It is possible to experience an actual "no available getvis" condition when, for example, several of the subpools are currently occupied by subpool "A" and "A" has 8K available but "B" issues a request and there are no available subpools. This circumstance is curious without revealing this design because it will appear to the user that the partition has adequate getvis allocated yet it cannot be obtained.

The various subpools are managed in the anchor table and there are several valuable pointers in the control block including label GTVSHIGH at +X'428'. This value shows the getvis *high-water mark* which indicates the largest demand for getvis during the execution in this partition and can be extremely valuable when tuning storage requests especially with VTAM or CICS. This field is displayed, for example, when issuing the undocumented AR GETVIS Fn command on VSE/SP 2.1.6 and later.

The System's Managed Storage

The SMCB address table's first entry contains the pointer to the VSE system's SMCB or SVA entry. This entry points to the beginning address of the SVA at +X'0C' or label SMSVABEG when using the SMCB DSECT and to the beginning of the *system* getvis area at +X'08' or label SMSGVIS.

The SVA is divided into three areas: the system directory list or SDL, the shared phase library and the system getvis area.

At the beginning of the SVA there is a map containing the pointers to each of these areas and the SDL immediately follows this map. The SDL contains copies of 72-byte core image library directory entries and is used for three performance-related functions.

For SVA resident phases, that is, phases that are actually loaded into the shared phase library at IPL time or subsequently, the SDL entry is used to point to the phase in the SVA to execute when a problem program requests the phase. The advantage is that the phase *actually* executes within the SVA and can be used by multiple tasks without causing a load of the phase into the requestor's partition. By definition, these phases must be reentrant and must be link-edited as SVA eligible. These phases are loaded using the $XXXSVA,SVA statement in the SET SDL IPL procedure.

See VSE *page 92*

CASE
A Technology In Transition

By Pete B. Privateer

"The new source of power is not money in the hands of the few, but information in the hands of the many."

John Naisbett
Megatrends

Since its introduction in 1984, Computer Aided Software (or systems) Engineering (CASE) has been touted as the newest *silver bullet* to data processing problems. Because of this marketing hoopla, some industry commentators question whether CASE has lived up to its promises. Does this mean that CASE is a disappointment or are the expectations so great that no technology this early in its evolution can live up to them?

For a long time DP management has been seeking a way to reduce application development backlogs, decrease the amount of time spent on maintaining existing applications and more efficiently solve *mission critical* business problems through Information Systems (IS) technology.

In *An Information System Manifesto,* James Martin writes that computers can and should be used to "automate automation." CASE is not the first new concept introduced to fulfill that challenge. Many technologies which existed before the introduction of CASE have been inaugurated with much fanfare only to have expectations subsequently lowered. When teleprocessing was first announced in the mid-60s, database in the late 1960s, relational database in the late 1970s and SQL in the early 1980s, there was an initial euphoria soon followed by a backlash of pessimism.

However, after the early market gloom, users settled down to figure out how they could exploit these technologies. In the meantime, galvanized by user skepticism, vendors modified their products in response to market demands. The natural dynamics of the marketplace, as always, helped the vendors take what has been learned during the first generation of experimentation and apply this to the next generation's products. Therefore, in the second and third generation of a new technology, greatly enhanced usefulness and return on investment are seen.

In evidence of this, the second and third incarnations of teleprocessing, database and relational technology have all developed into well-established concepts. Although each one suffered early suspicion and setbacks, they now are vital to modern IS departments. It would be hard to conceive of an enterprise running its business without on-line transaction processing and modern database management systems. In fact, these tools are virtually taken for granted.

Consider one of the recent new technologies; SQL-based Database Management Systems (DBMSes). SQL technology for the IBM mainframe environment first made its debut in the early 1980s with SQL/DS and, more recently, DB2. Few, if any, shops took it seriously then and it was several years before there were significant SQL-based applications up and running. Today few experts doubt that SQL will be the standard database language well into the next century.

However, did the first generation SQL products immediately dominate the market? Were you able to immediately quantify the productivity improvements that SQL provided? Did early benchmarks demonstrate the superiority of SQL over other technologies? Hardly!

SQL has become a pre-eminent force and it is an elegant enabling technology because it holds clear promise for the future. It is by no means a finished product but both vendors and users have passed through most of the first generation learning curve.

You can apply the same paradigm to CASE. Since the first PC-based CASE tools became available in 1984, CASE technology has been in the experimentation stage. After five years of investigation, CASE technology is moving into the next generation. What hope does the IS community see in CASE that will sustain this transition?

As technology in general (aerospace, hardware, manufacturing, communications and so on) becomes more complex and intricate, the underlying software systems become increasingly complicated. More is demanded of software applications today than in the past. The software that carries the space shuttle into low earth orbit requires *30 times* as much code as the computer programs that took men to the moon and back.

Companies are no longer content with simple accounting packages that maintain the general ledger. A modern enterprise today requires shared, integrated information systems that handle complex mission-critical functions. These include applications like on-line reservation systems which will soon handle all your travel reservation needs from airlines, to rental cars, to hotels, to the tee time at your favorite golf course. Or they might include integrated banking applications that can manage all your accounts and give you on-line access to your investments 24 hours-a-day, seven days-a-week. Last, they might include computer integrated manufacturing applications that drive numerically controlled machines on the shop floor from CAD/CAE workstations.

While these kinds of technological solutions to practical business problems and opportunities are sought after by executive management, most IS departments today are still dealing with redundant, chaotic databases and application software made up of spaghetti-like programs. When management wants to introduce new products or methods or use information resources to a competitive advantage, it becomes bogged down in the quagmire of old, antiquated systems.

The need for more and better software is further accelerated by the quantum increase in the availability of fast, cheap computing power afforded by chip technology. It is estimated that by 1992, desk top and mainframe MIPs will be equivalent.

In short, without a radical new approach to software design and construction, application development cannot keep pace with the dynamic, constantly changing needs of the business. As the *Wall Street Journal* quipped a couple of years ago, "Oil and software are the two principal obstacles to economic progress."

Trying to meet the demands of modern software development without automated support would be like trying to design the next generation's microchips with pencils and sliderules. No one contemplates designing buildings, computers or modern aircraft without Computer Aided Design and Computer Aided Engineering (CAD/CAE) tools. Designing the interlocking computer applications of a modern enterprise is no less of an engineering challenge.

Despite these valid, if not somewhat emotional arguments for CASE technology, recent studies have shown that less than seven percent of IS professionals are using automated tools to design and build applications. If you believe that CASE tools are in transition from the experimental stage to the really practical level, why is there still some reluctance to fully embrace CASE technology?

There are five important factors that I believe have limited the acceptance of CASE so far. Hopefully, the second generation of CASE technology will address these factors and therefore accelerate acceptance.

The Need For Integrated Solutions

Most of the CASE tools available until recently have been first-generation implementations of CASE technology. These tools focused only on automating discrete phases of the application development life cycle. Those addressing the front end of the life cycle (sometimes called upper CASE) usually seek to automate some portions of the structured analysis and design techniques developed in the 70s. Since these techniques were conceived long before automated tools, the first-generation front-end CASE products often automate outdated manual techniques.

First-generation back-end CASE tools (sometimes called lower CASE) concentrate on automating the code construction phase. These included such products as code generators or 4GLs. They are capable of speeding up the coding process but because they are limited only to the coding aspects of the application development process, they have no way of ensuring that the code being built will satisfy the end-user's requirements.

While organizations may achieve some incremental benefit by using this first-generation technology, the real payoff for application developers will come when they adopt the next generation of CASE tools. Just as the early attempts to exploit relational technology achieved limited success compared to the current implementations, the early CASE strategies will be superseded by new, integrated, full life-cycle tools.

You are beginning to see this happen already. Recent surveys have shown that the fastest growing CASE vendors are those offering second-generation technology. This survey also indicated that while first-generation CASE products currently dominate the market, the users of those products are looking toward the next generation's integrated products twice as often as the older technology.

These second-generation vendors are focusing on integrated solutions that automate not just the phases of application development but also the transition between phases. These tools truly integrate the front end of the life cycle with the back end. Some generate complete COBOL applications from pictures! In addition, these products are founded on third- and fourth-generation development techniques that have been specifically designed to take advantage of automated tools.

One common characteristic of these second-generation products is an integrated central data store or repository. They define integration as a common, shared database. Information about the enterprise, business requirements and design specifications are stored uniquely and non-redundantly in this database and accessed by all life-cycle tools. This is fundamentally different from the first-generation concept of a dictionary.

The first-generation CASE tools are now experiencing the demand for full life-cycle functionality. Unfortunately, the architecture of these early products was not designed to support the integration of all phases of development. Therefore, they attempt to meet this demand by building interfaces to other products (that is, back-end code generation from front-end tools).

These interfaces allow one tool to pass information collected during the initial phases of the life cycle, to tools on the back end of the life cycle. However, the problem with interfaces is that information must be maintained in at least two separate places and, therefore, quickly becomes corrupted. In contrast, second-generation products allow multiple tools that address each phase of the development life cycle to share information through a common database.

Industry leader IBM's vision of an application development environment mirrors this integration scheme. At the CASExpo conference last May, IBM indicated that a central repository for all business requirements and application definitions was crucial to its Systems Application Architecture (SAA) strategy. IBM also indicated that data sharing is how it expected to achieve CASE tool integration. Application developers that want to prepare for the new world would be well advised to become familiar with second-generation CASE technology.

The Need To Get Real Work Done

Another problem with the way CASE was initially implemented is that most front-end products are rarely used to do real application development. Instead, they have been used primarily for documentation purposes. A survey of the usage patterns of one commonly used CASE tool indicated that the most frequently used feature of the product was the presentation graphics function.

Clearly one benefit of using CASE products is better communication; however, IS management is usually not convinced that real work is going on until they see application source code being produced. I might also add that using a CASE tool to simply produce presentation graphics is like using a hydraulic press to crack walnuts — it probably will work, but it is an inefficient use of the instrument.

Even when the requirements for an application are documented in these CASE tools, the programmers are still free to ignore the specifications and code it any way they see fit. In fact, there is little of the documentation that can be directly used by the programmer, so there is not much incentive to use it.

Efficiency can be lost when information is handed off between phases of the development process. Even when interfaces are used to give the programmers some start on their application construction, so much manual work has to be done on the back end that the benefits of front automation and design engineering are eroded. You cannot take those pretty little pictures, wrap them around the tape drive and make the blue box in the glass house run.

The good news is that IS professionals are beginning to make use of more advanced CASE technology to build real applications. A recent survey was conducted by the Barton Group of some 3,000

CASE users. In the preliminary results, a substantial percentage reported significant improvements in the quality of their business applications due to CASE. However, this group tended to be the more experienced CASE users who are building their applications on common designs and integrated data. They are actively using CASE tools to design and build applications, rather than just document them.

The Need For New Development Techniques

Some users' attempts to implement CASE have focused on the tools by themselves. Minimal emphasis has been given to the methodologies and techniques behind the tools. All CASE software, (artificially intelligent or not!) is pretty stupid. It just sits there and looks at you until you enter something into it. Knowing how to push the right button is not the problem. You must know what questions to ask before you can represent the answers in a CASE tool.

CASE tools may change the way we communicate but they do not automate human interaction. Nor are they a substitute for experience and knowledge. Just as CAD/CAE must rely on the disciplines of engineering, mathematics and physics, CASE must also be combined with a set of engineering disciplines which are fashioned to work with the new automated tools.

Many IS shops do not have an established set of principles, methods, standards and techniques which are consistently enforced. Unfortunately, too many industry experts and consultants today recommend that IS organizations adopt a manual methodology first *before* they consider implementing a CASE tool. They are missing the important opportunity that CASE presents to help IS establish consistent development standards and procedures.

In reality, most IS shops modify their methodologies and development techniques to fit their design and development aids. In turn, most CASE tools are influenced by methodologies and development techniques. Pick any manual methodology and I will guarantee you that you cannot find a CASE tool that *exactly* implements it. Therefore, organizations often put a double barrier in front of themselves — first, adaptation to a new development approach, then, if they adopt a CASE product, adaptation of that development approach to the tool.

Would it make more sense to adopt both new technologies at the same time? Why not choose a development approach designed to work synergistically with automated tools and integrate both into your development environment. Many advanced CASE users have found that CASE tools actually help them learn new development techniques. There is also no better way to ensure that the new guidelines you have adopted are followed than by giving your IS professionals the tools that automate those guidelines.

The Need To Focus On Quality Not Quantity

Many companies have said they will adopt CASE tools only when they can cost justify the expenditures for the technology. Therefore, much of the initial attention IS organizations place on implementing CASE systems has centered around application developer productivity. To these companies, productivity simply means building an application faster, cheaper and with fewer resources. This view hides the real benefit of CASE — the increased quality of the applications being developed. Unfortunately, quality is hard to measure.

Then how do we convince the skeptics that CASE will provide a return on their investment? Take a closer look at productivity. Most experts agree that productivity has two different aspects: efficiency and effectiveness.

Efficiency

Efficiency has to do with completing a task in less time and with fewer resources. Effectiveness is concerned with the quality and applicability of the completed task. In other words, productivity is not just doing the job faster; it is doing the *right* job faster and doing the job correctly the *first* time. Where is the productivity if I can perform the job in one-fourth the time, however, in order to get it right I have to do it five times?

Efficiency is easily measured. You merely have to perform the job once measuring the length of time it takes to complete the job and then perform the same job again, using the productivity aid being evaluated and comparing the two completion times (this, of course, assumes the resources are kept constant).

Effectiveness

Effectiveness is much harder to measure. Quality is often a subjective judgment and, therefore, difficult to quantify. However, it can be argued that effectiveness is by far the most important criteria in determining the ultimate worth of a productivity aid to the organization. A productivity aid that does not do what it was intended to do is worse than worthless.

In software development, quality is usually based on how well the developed application meets the end user's needs (assuming the user knows what his needs are). This would be a reasonable measure of quality except that the end user's needs are rarely static.

It is a fact of life in software development that a program will require modification once it is put into production. Therefore, a software program is effective if the application satisfies a current need *and* if it easily adapts to the changing needs of the business. Put another way, productivity in software development should not be measured over the time it takes to develop the application but over the entire life span of the application.

What does all this have to do with CASE? If you apply the above criteria for measuring productivity to application development using CASE, you can come to some conclusions that may help the skeptics cost justify CASE technology.

There is little question that CASE can increase the efficiency of application development staffs. Many CASE users who have been doing formal requirements analysis, systems and database design, prototyping and so on by manual means have reported improvements of between 60 and 150 percent when using CASE.

The real savings, however, come from CASE's contribution to the effectiveness of the application development staff. Some of the benefits of CASE are:

- A better understanding of the end-user's needs
- A higher quality application
- A shared data and reusable code environment
- Projects delivered on time and within budget
- Lower ongoing support needs
- Code that is consistent with what the end user wanted
- Flexible applications that can incorporate new business needs
- Better communication
- Clear definition of what is important
- Clear and consistent definition of the problem
- Automatic generation of complete applications from specifications

• Documentation that is consistent with the code.

It is, of course, hard to assign dollar values to these benefits. However, consider the American automotive industry in the mid-70s. Early in that decade it began losing market share to the Japanese. The reason was that the Japanese made *higher quality* products.

In order to stem the tide, American automotive manufacturers decided that drastic steps were in order. No longer could they get away with shoddy workmanship and the "we'll fix it later" syndrome. In the mid-70s, the automotive industry made a substantial investment and commitment to improving the quality of its products.

Investments were made in innovative tools such as statistical quality control, robotics, high-tech materials and new management practices. Yet I am sure if the manufacturers had tried to cost justify each of these tools, they would have found it to be just as hard as cost justifying CASE.

Now in the late 1980s with these new technologies having become institutionalized, you can look back and ask, "What is the automotive industries' return on investment?" Well, for one thing it is still in business which for some was questionable 10 years ago. In addition, Ford Motor Company has recorded two of its most profitable years in its history. Was this due to Ford's stand on quality? I am not sure that will ever be known for certain. One thing is sure, Ford is not backing off its commitment to effectiveness.

The Need To Leverage The Current Investment In Existing Applications

One of the major inhibitors to CASE implementation has been the notion that CASE technology is only applicable to new development. Recent studies have shown that nearly 80 percent of the systems development dollar is spent on keeping the existing patchwork quilt of applications up and running. How does CASE technology help with existing systems?

Most of the first-generation CASE products do little to help reengineer the 400 person years of applications most shops have in production. Yet, if you truly expect to be able to get a handle on your information resources, you must find tools that will allow you to migrate your existing applications into a CASE development environment. Fortunately, some of the second-generation CASE offerings are addressing this critical need.

It is important to keep reengineering (or reverse engineering) in perspective. It would take a clever piece of software to reengineer applications that were poorly engineered to begin with. Consider a typical product code: ABX9009Q77789RSV. The first three characters tell you what the next four digits mean, the eight character defines a product type and the last three characters present the product's status. It would take a lot of Artificial Intelligence (AI) to figure out the data model for just this one data element and the real world is full of examples like this.

The best current technologies allow developers to reuse and enhance existing data definitions and program structures by loading them into CASE tools where they can be managed. If the application developer wishes to map them into an implementation independent model (such as an Entity Relationship model), these second-generation tools often provide automated support for that capability.

Obviously, reengineering cannot be undertaken without a considerable amount of human assistance. However, it is important for organizations to realize that implementing a CASE strategy does not mean abandoning their existing investment in business applications. Today's truly effective CASE technologies can use that investment as a starting point.

Summary

These are just a few of the factors organizations must consider now that CASE is in transition from a first-generation to a second-generation technology. To be successful with CASE, IS organizations cannot limit its scope. CASE must address the entire application development process from understanding the mission of the enterprise to understanding what business problems need to be solved, to formulating project plans, to understanding business requirements, to designing technology solutions to those requirements, constructing the designs and testing and implementing those designs.

According to the Barton Group project, organizations that have used CASE tools for three-plus years, completed at least four CASE developed projects and adopted engineering disciplines were the ones that reaped the most substantial rewards from CASE. I suspect that if you conducted a survey of relational database users during the transition from first- to second-generation technology, you would have seen similar results.

One of the choices that executive management faces is whether to start the learning curve now, today or to wait until CASE is *perfected*. In *An Information Systems Manifesto,* James Martin recommends that the stewards of the corporation ask the following question: Who could take my business away from me by using technology more agressively and in new directions? Those organizations that wait for CASE to evolve will find themselves far behind their competitors.

Like all successful managers, IS management must make a commitment to specific objectives in the performance of its charter. Data processing is arguably in the same boat as the automotive industry in the early '70s. The demand for more and more sophisticated software is rising far beyond the current capabilities of IS development to keep up with it. World competition requires that corporations use their information resources as a strategic weapon.

If management is content with its current development practices and technologies, then perhaps CASE has little value and is not worth the cost. However, if today's progressive corporations want to make a commitment to new and better ways of developing computer solutions to mission critical business problems, then they will have to embrace a strategy like the automotive industry's. They will need to make a commitment to application development quality and the tools and techniques that make that possible. If IS management makes that commitment, the costs of implementing CASE technology will be trivial compared to the results that can be achieved with it.

ABOUT THE AUTHOR

Pete B. Privateer is Director of Product Programs for KnowledgeWare, Inc. Privateer's responsibilities include product management functions for all KnowledgeWare product lines and he serves as the focal point for market strategies and for input into the product planning process. He has been involved with the development and implementation of CASE tools and methods for more than eight years. He is a regular presenter at CASE conferences and a contributor to industry publications.

How To Use VM's Essential XEDIT Commands

By Steve Eckols

In this article, you will learn a powerful subset of XEDIT commands. With the commands in this subset, you will be able to do a large portion of your editing work.

In order to use XEDIT effectively, you *must* know the material this article presents. As a result, I encourage you to read it carefully and if a VM terminal is available to you, practice as you read. Take all the time you need to master the XEDIT commands presented here.

The material is divided into eight parts that teach you how to: (1) use XEDIT's default PF key assignments, (2) use common prefix commands, (3) use XEDIT's vertical scrolling commands, (4) work in input mode, (5) save your work, (6) end an editing session, (7) use the on-line help facility and (8) use three XEDIT short cut commands. Because each section builds on the information the preceding ones have presented, I suggest you read the sections in order.

How To Use The Basic XEDIT PF Key Assignments

Terminals in the 3270 family, the kind you are most likely to work at as an XEDIT user, have a set of keys that can be associated with commonly used commands to make terminal work easier. Those keys are called *program function keys* or *PF keys* for short. By default, some common XEDIT commands are associated with function keys one through 12. (If your 3270-family terminal has 24 function keys, the default assignments for PF13 through PF24 duplicate the functions of PF1 through PF12. For example, PF2 and PF14 perform the same function.)

Figure 1 lists the default PF key assignments and briefly describes the function of each. Frankly, a few of the default PF key assignments are for obscure functions. As a result, you are likely to want to change them to perform functions you use more often.

However, three of the default PF key assignments are for functions you will use often; they are the ones shaded in Figure

F I G U R E 1

PF keys	Function
PF1/PF13	Displays a help menu.
PF2/PF14	Adds a single blank line immediately after the line that contains the cursor.
PF3/PF15	Ends XEDIT if no editing changes have been made.
PF4/PF16	Moves the cursor ahead to the next XEDIT tab stop.
PF5/PF17	Used to issue a previously entered command to find or modify text selectively.
PF6/PF18	Redisplays the last command entered.
PF7/PF19	Scrolls the display backward (upward) one screen.
PF8/PF20	Scrolls the display forward (downward) one screen.
PF9/PF21	Reexecutes the last command issued. (This is equivalent to pressing PF6 and enter.)
PF10/PF22	Shifts the display left or right to let you view records that are wider than the display.
PF11/PF23	Splits or joins two adjacent lines of text.
PF12/PF24	Moves the cursor back and forth between its position in the file area and the command line.

Default PF key functions

CIRCLE #80 on Reader Service Card ▲

FIGURE 2 PART 1 and 2

To scroll forward (down), you press the PF8 (or PF20) key . . .

```
LISTMAST COBOL     A1  F 80  TRUNC  72 SIZE  236 LINE   0 COL   1 ALT   0










      * * * TOP OF FILE * * *
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7
             IDENTIFICATION DIVISION
             *
             PROGRAM-ID      LISTMAST
             *
             ENVIRONMENT DIVISION
             *
             INPUT-OUTPUT SECTION
             *
             FILE-CONTROL
====  _
                                                            X E D I T   1 FILE
```

. . . and the line that was at the bottom of the file area is shifted up to the top of the file area.

```
LISTMAST COBOL     A1  F 80  TRUNC  72 SIZE  236 LINE  18 COL   1 ALT   0

             FILE-CONTROL
                 SELECT INVMAST ASSIGN TO SYS020-AS-INVMAST
                 SELECT INVLIST ASSIGN TO SYS006-UR-1403-S
             *
             DATA DIVISION
             *
             FILE SECTION
             *
             FD  INVMAST
                 LABEL RECORDS ARE STANDARD
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7
             RECORD CONTAINS 128 CHARACTERS
             *
             01  INVENTORY-MASTER            PIC X(128)
             *
             FD  INVLIST
                 LABEL RECORDS ARE OMITTED
                 RECORD CONTAINS 132 CHARACTERS
             *
             01  PRINT-AREA                  PIC X(132)
====  _
                                                            X E D I T   1 FILE
```

1. PF7 and PF8 are associated with commands you use to scroll the text forward and backward. PF3 is associated with a command you can use to leave XEDIT.

How To Scroll Up and Down: PF7 and PF8

Even though XEDIT's full-screen features make editing much easier than with a line-oriented editor, most files contain more data than you can see on one screen. As a result, you need to be able to move forward and backward in a file (or, if you prefer, down and up) to find the parts you need to edit. This kind of movement through a file is called *scrolling*.

To scroll foward in a file (downward), you press the PF7 (or PF19) key and to scroll backward in a file (upward), you press the PF8 (or PF20) key. To understand how this works, take a look at Figure 2. From the screen in part 1 of the figure, I pressed the PF8 key to scroll forward (down).

The result is in part 2 of the figure. As you can see, the line that was at the bottom of the screen in part 1 of the figure is at the top of the screen in part 2. The current line is now a different line, although the location of the current line is still in the middle of the screen. (Notice the LINE value in the file identification line is now 18 instead of 0.) Scrolling up with the PF7 keys works in the same way, except data is scrolled backwards toward the beginning of the file.

The PF7 and PF8 keys are actually associated with XEDIT commands that can be entered from the command line to scroll one screen up or down. They are the BACKWARD and FORWARD commands. I will describe them in a moment, along with other XEDIT commands you can use to perform other scrolling operations.

By the way, as you scroll either with the PF keys or by issuing the scrolling commands, you will learn later in this article that it is possible to *wrap around* from one end of the file to the other end. For example, if you are positioned at the end of your file and you scroll down, the top of the file becomes the current line. This can be confusing if you do not recognize what is happening. However, after you are aware of this feature, you can use it to your advantage as you navigate through the files you edit.

How To End XEDIT: PF3

The other default PF key assignment you are likely to use right away is PF3. It lets you leave the editor and return to the CMS environment. PF3 is associated with the XEDIT QUIT command. QUIT causes the editor to end if changes made to the file have been saved. If changes have not been saved, PF3 will not let you leave the editor directly. Later in this article, I will describe how to save your work and different ways to leave the editor. For now, though, you should realize that PF3 provides a quick way back to CMS if you have not made any changes during your editing session or have already saved the changes you made.

How To Use Prefix Commands

After the simple scrolling commands associated with PF keys, the XEDIT commands you are likely to depend on most heavily are prefix commands. Prefix commands let you perform basic operations like opening new lines to add text to a file, deleting lines and moving and copying lines. You should recall that you enter prefix commands in the prefix areas of the lines you want them to affect.

Figure 3 lists the XEDIT prefix commands you should learn right away. The basic prefix commands in Figure 3 fall into five groups: commands to (1) add lines, (2) delete lines, (3) duplicate lines, (4) copy or move lines and (5) set the current line pointer.

Before I present the prefix commands for these functions, you should know that you can issue most of them in two ways. First, you can enter an individual prefix command and specify how many lines it should affect. Second, you can enter paired prefix commands to mark a block of text to be affected by the commands; when you use the prefix commands in this way, they are called *block commands*. It is often desirable to use block commands when you want to operate on a group of lines that spans more than one screen. Even if a group of lines fits entirely on one screen, you are less likely to make a mistake if you use block commands rather than an individual prefix command. That is because you do not have to count the number of lines to be processed when you use block commands. As I present the

FIGURE 3

Add blank lines	
A or I	Add one blank line at the indicated position.
A*n*, *n*A, I*n* or *n*I	Add *n* blank lines at the indicated position.
Delete lines	
D	Delete the indicated line.
D*n* or *n*D	Delete *n* lines from the indicated position.
DD	Marks the beginning and end of a group of lines to be deleted.
Duplicate lines	
''	Duplicate the indicated line one time.
''*n* or *n*''	Duplicate the indicated line *n* times.
''''	Marks the beginning and end of a group of lines to be duplicated one time.
''''*n* or *n*''''	Marks the beginning and end of a group of lines to be duplicated *n* times.
Copy or move lines	
F	Insert the copied or moved lines following the indicated line.
P	Insert the copied or moved lines prior to the indicated line.
C	Copy the indicated line.
C*n* or *n*C	Copy *n* lines, beginning with the indicated line.
CC	Marks the beginning and end of a group of lines to be copied.
M	Move the indicated line.
M*n* or *n*M	Move *n* lines, beginning with the indicated line.
MM	Marks the beginning and end of a group of lines to be moved.
Set the current line pointer	
/	Make the indicated line the current line.

Basic XEDIT prefix commands

Announcing 3090 Central and Expanded Storage from EMC Corporation.

As a 3090 user you now have the choice to make a smart business decision. Our 3090 Central and Expanded Storage upgrades offer you the reliability, performance, local and remote support that you demand at a price 30% lower than that of IBM.

Actually, this good news is old news. For 10 years we have been offering users cost-effective, state-of-the-art storage solutions. Our strong background in mainframe storage solutions has been recognized by Fortune 500 companies worldwide. These companies have realized that we provide a better business decision for their mainframe storage needs. Now, we have built this mainframe experience into our 3090 product offerings.

For additional good news on how EMC Corporation's 3090 storage upgrades can help you reach your business objectives call or write: EMC Corporation, Hopkinton, MA 01748-9103 **1-800-222-EMC2.** (In MA, 508-435-1000.)

EMC² **The System Enhancement Company.**

IBM is a registered trademark of International Business Machines Corporation.

CIRCLE #95 on Reader Service Card ▲

FIGURE 4 PART 1 and 2

The prefix command I6 tells XEDIT to add six blank lines after the indicated line.

```
LISTMAST COBOL    A1  F 80  TRUNC = 72 SIZE = 236 LINE = 18 COL = 1 ALT = 0

           FILE-CONTROL.
               SELECT INVMAST ASSIGN TO SYS020-AS-INVMAST.
               SELECT INVLIST ASSIGN TO SYS006-UR-1403-S.
          *
           DATA DIVISION.
          *
I6_        FILE SECTION.
          *
           FD  INVMAST
               LABEL RECORDS ARE STANDARD
     |...+....1....+....2....+....3....+....4....+....5....+....6....+....7
           RECORD CONTAINS 128 CHARACTERS.
          *
           01  INVENTORY-MASTER           PIC X(128).
          *
           FD  INVLIST
               LABEL RECORDS ARE OMITTED
               RECORD CONTAINS 132 CHARACTERS.
          *
           01  PRINT-AREA                 PIC X(132).
====
                                                            X E D I T  1 FILE
```

Six blank lines are inserted; new text can be keyed into them.

```
LISTMAST COBOL    A1  F 80  TRUNC = 72 SIZE = 242 LINE = 24 COL = 1 ALT = 1

           FILE SECTION.






          *
           FD  INVMAST
               LABEL RECORDS ARE STANDARD
     |...+....1....+....2....+....3....+....4....+....5....+....6....+....7
           RECORD CONTAINS 128 CHARACTERS.
          *
           01  INVENTORY-MASTER           PIC X(128).
          *
           FD  INVLIST
               LABEL RECORDS ARE OMITTED
               RECORD CONTAINS 132 CHARACTERS.
          *
           01  PRINT-AREA                 PIC X(132).
====
                                                            X E D I T  1 FILE
```

prefix commands in this section, I will illustrate both techniques.

How To Add Blank Lines: the A and I Prefix Commands

One way to add lines to a file is by inserting blank lines, then entering the data you want into the lines. To add blank lines to a file, you can use the A (for add) or I (for insert) prefix command. It does not matter which you select; their functions are the same. Both are provided so users with experience using IBM's other mainframe system text editors will find XEDIT easier to work with. (Under the DOS/VSE editor, ICCF, the prefix command you use to add new lines to a file is A; under the ISPF text editor supported by TSO on MVS systems, you use the I command to add new lines to a file.)

To understand how these commands work, consider Figure 4. In part 1 of the figure, I have entered the prefix command

I6

to add six blank lines after the line that contains the text

FILE SECTION

(I could have used 6I, A6 or 6A instead; they all produce the same result.) When I press the enter key, the prefix command is processed and six blank lines are inserted as you can see in part 2. At this point, I can go ahead and key new text into those lines. Notice also in part 2 that the size of the file has increased from 236 lines to 242. Although the same line of text remains as the current line, the current line pointer changed from 18 to 24 to reflect the six lines that were added above it.

If you only want to add a single new line, you do not need to specify the number of lines on the A or I command. For instance, the commands

I

and

I1

are equivalent. If you refer back to Figure 1, you will see that you can also add a single blank line to a file using the PF2 key. To do that, simply position the cursor on the line above the location where you want the new line to be added and press PF2.

Unlike the other prefix commands, there are no block command options available with the A or I commands. That is because it does not make sense to mark a block of existing lines after which a group of new lines will be added; it is only the last line of the block that would be significant.

How To Delete Lines: the D Prefix Command

To delete lines from a file, you use the D (for delete) prefix command. You can enter simply

D

to delete one line or a command like

D6

or

6D

FIGURE 5 PART 1 and 2

The D prefix command tells XEDIT
to delete six lines starting with the indicated line.

```
LISTMAST COBOL    A1  F 80  TRUNC = 72 SIZE = 242 LINE = 24 COL = 1 ALT = 1

           FILE SECTION.
D6_





          *
           FD  INVMAST
               LABEL RECORDS ARE STANDARD
     |...+....1....+....2....+....3....+....4....+....5....+....6....+....7>
           RECORD CONTAINS 128 CHARACTERS.
          *
           01  INVENTORY-MASTER           PIC X(128).
          *
           FD  INVLIST
               LABEL RECORDS ARE OMITTED
               RECORD CONTAINS 132 CHARACTERS.
          *
           01  PRINT-AREA                 PIC X(132).
====
                                                            X E D I T  1 FILE
```

The indicated six lines have been removed from the file.

```
LISTMAST COBOL    A1  F 80  TRUNC = 72 SIZE = 236 LINE = 18 COL = 1 ALT = 2

           FILE-CONTROL.
               SELECT INVMAST ASSIGN TO SYS020-AS-INVMAST.
               SELECT INVLIST ASSIGN TO SYS006-UR-1403-S.
          *
           DATA DIVISION.
          *
           FILE SECTION.
          *
           FD  INVMAST
               LABEL RECORDS ARE STANDARD
     |...+....1....+....2....+....3....+....4....+....5....+....6....+....7>
               RECORD CONTAINS 128 CHARACTERS.
          *
           01  INVENTORY-MASTER           PIC X(128).
          *
           FD  INVLIST
               LABEL RECORDS ARE OMITTED
               RECORD CONTAINS 132 CHARACTERS.
          *
           01  PRINT-AREA                 PIC X(132).
==== _
                                                            X E D I T  1 FILE
```

to delete a specified number of lines beginning with the line where the command is entered. Figure 5 shows you how the D prefix command works. In part 1, I have entered the command to remove the six blank lines I added to the file in Figure 4. When I press the enter key, they are deleted as you can see in part 2.

An alternative to using a single prefix command that specifies a number of lines to delete is to use a pair of DDs as block commands. When you use block prefix commands, you mark the beginning and end of a group of lines to be processed. For instance, to delete a block of lines, you could enter

DD

in the prefix areas of the first and last lines of the group to be deleted. Figure 6 shows how you could use block commands to perform the same delete operation Figure 5 illustrates.

How To Duplicate Lines: the " Prefix Command

If you want to duplicate a single line or a group of lines, you can use the " prefix command either as an individual command or as a block command. Figure 7 shows how you can use the " command to duplicate a single line three times. In this case, I am keying in a COBOL source program and I have reached a part of the program where I need to enter a series of similar lines. Rather than key in all of the characters of each, it is easier to enter the

FIGURE 6 PART 1 and 2

The DD prefix block commands delimit
the group of lines to be removed from the file.

```
LISTMAST COBOL    A1  F 80  TRUNC = 72 SIZE = 242 LINE = 24 COL = 1 ALT = 1

           FILE SECTION.
DD




DD_
          *
           FD  INVMAST
               LABEL RECORDS ARE STANDARD
     |...+....1....+....2....+....3....+....4....+....5....+....6....+....7
           RECORD CONTAINS 128 CHARACTERS.
          *
           01  INVENTORY-MASTER           PIC X(128).
          *
           FD  INVLIST
               LABEL RECORDS ARE OMITTED
               RECORD CONTAINS 132 CHARACTERS.
          *
           01  PRINT-AREA                 PIC X(132).
====
                                                            X E D I T  1 FILE
```

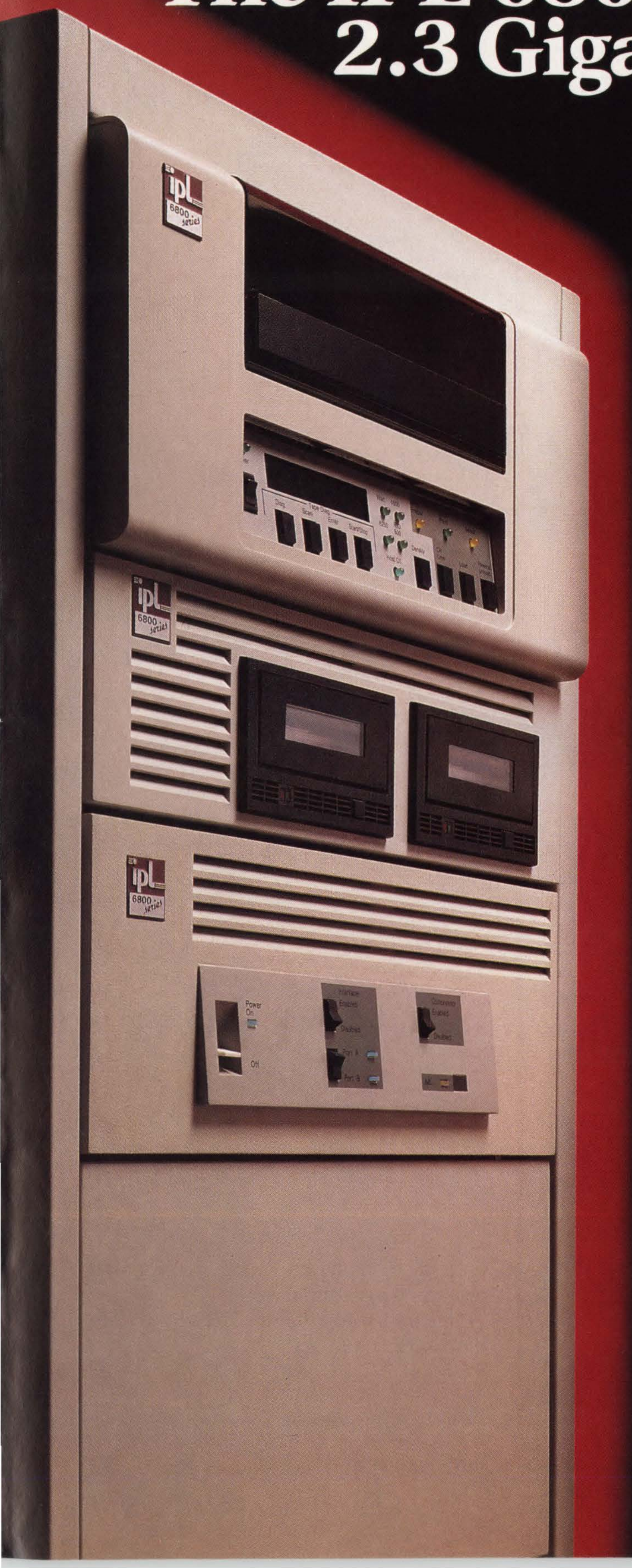
The indicated group of lines has been removed from the file.

```
LISTMAST COBOL    A1  F 80  TRUNC = 72 SIZE = 236 LINE = 18 COL = 1 ALT = 2

           FILE-CONTROL.
               SELECT INVMAST ASSIGN TO SYS020-AS-INVMAST.
               SELECT INVLIST ASSIGN TO SYS006-UR-1403-S.
          *
           DATA DIVISION.
          *
           FILE SECTION.
          *
           FD  INVMAST
               LABEL RECORDS ARE STANDARD
     |...+....1....+....2....+....3....+....4....+....5....+....6....+....7
               RECORD CONTAINS 128 CHARACTERS.
          *
           01' INVENTORY-MASTER           PIC X(128).
          *
           FD  INVLIST
               LABEL RECORDS ARE OMITTED
               RECORD CONTAINS 132 CHARACTERS.
          *
           01  PRINT-AREA                 PIC X(132).
==== _
                                                            X E D I T  1 FILE
```

For System 38, 43XX, 9370 Users

The IPL 6800 Series Turns 2.3 Gigabytes Of Data



Into This

Finally an unattended and revolutionary storage medium for all data processing tasks from IPL.

IPL's 6800 series of tape subsystems boasts a sleek, space-saving, office environment design. These modular, rack-mounted subsystems can easily be configured to meet your specific requirements and connect directly to your IBM system. No hardware or software alterations necessary.

But good looks are just the beginning. For example, the IPL 6860 performs data storage magic . . . using helical scan technology to replace 12 or more reel-to-reel tapes with one 8mm cartridge. You get 2.3 gigabytes of storage with no operator intervention, and proven reliability thanks to IPL's continued commitment to technological innovation.

Cartridges or reel-to-reel. Only IPL lets you choose either one . . . or combine both to fit your applications precisely.

Call IPL today at 1-800-338-8ipl, in MA (617) 890-6620 and cut your data processing tasks down to size with the IPL 6860 8mm, 2.3 gigabyte subsystem.

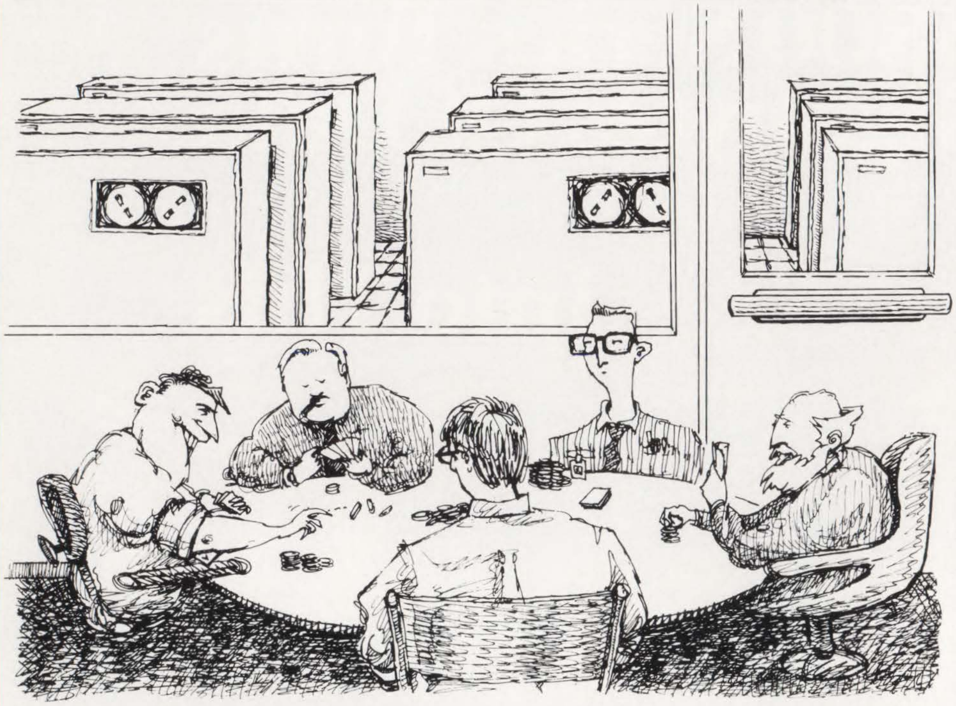
ipl systems, inc.

360 Second Avenue
Waltham, MA 02154

IBM is a registered trademark of International Business Machines Corporation.

CIRCLE #96 on Reader Service Card ▲

Lighter Side



"Is this unattended operations bit great or what!"

FIGURE 7 PART 1 and 2

The " prefix command tells XEDIT to duplicate the indicated line three times.

```
TESTFILE COBOL    A1  F 80  TRUNC = 72 SIZE = 39 LINE = 36 COL = 1 ALT = 31
            01   PRINT-AREA                PIC X(132).
           *
           WORKING-STORAGE SECTION.
           *
           01   SWITCHES.
           *
                05  INVMAST-EOF-SW                          PIC X    VALUE 'N'.
                    88   INVMAST-EOF                                 VALUE 'Y'.
           *
           01   PRINT-FIELDS               COMP-3.
     |...+....1....+....2....+....3....+....4....+....5....+....6....+....7..
           *
                05  SPACE-CONTROL          PIC S9.
 "3_            05  LINES-ON-PAGE          PIC S999 VALUE +55.
  * * * END OF FILE * * *

====
                                                             X E D I T  1 FILE
```

The indicated line has been duplicated three times.

```
TESTFILE COBOL    A1  F 80  TRUNC = 72 SIZE = 42 LINE = 36 COL = 1 ALT = 32
            01   PRINT-AREA                PIC X(132).
           *
           WORKING-STORAGE SECTION.
           *
           01   SWITCHES.
           *
                05  INVMAST-EOF-SW                          PIC X    VALUE 'N'.
                    88   INVMAST-EOF                                 VALUE 'Y'.
           *
           01   PRINT-FIELDS               COMP-3.
     |...+....1....+....2....+....3....+....4....+....5....+....6....+....7..
           *
                05  SPACE-CONTROL          PIC S9.
                05  LINES-ON-PAGE          PIC S999 VALUE +55.
 _              05  LINES-ON-PAGE          PIC S999 VALUE +55.
                05  LINES-ON-PAGE          PIC S999 VALUE +55.
                05  LINES-ON-PAGE          PIC S999 VALUE +55.
  * * * END OF FILE * * *

====
                                                             X E D I T  1 FILE
```

FIGURE 8 PART 1 and 2

The indicated group of lines will be duplicated four times.

```
TESTFILE COBOL    A1  F 80  TRUNC = 72 SIZE = 42 LINE = 36 COL = 1 ALT = 32
            01   PRINT-AREA                PIC X(132).
           *
           WORKING-STORAGE SECTION.
           *
           01   SWITCHES.
           *
 """4           05  INVMAST-EOF-SW                          PIC X    VALUE 'N'.
                    88   INVMAST-EOF                                 VALUE 'Y'.
 """_      *
           01   PRINT-FIELDS               COMP-3.
     |...+....1....+....2....+....3....+....4....+....5....+....6....+....7..
           *
                05  SPACE-CONTROL          PIC S9.
                05  LINES-ON-PAGE          PIC S999 VALUE +55.
                05  LINES-ON-PAGE          PIC S999 VALUE +55.
                05  LINES-ON-PAGE          PIC S999 VALUE +55.
                05  LINES-ON-PAGE          PIC S999 VALUE +55.
  * * * END OF FILE * * *

====
                                                             X E D I T  1 FILE
```

The indicated group of lines was duplicated four times. Because the lines were inserted above thc current line, the text was not scrolled down.

```
TESTFILE COBOL    A1  F 80  TRUNC = 72 SIZE = 54 LINE = 48 COL = 1 ALT = 33
                05  INVMAST-EOF-SW                          PIC X    VALUE 'N'.
                    88   INVMAST-EOF                                 VALUE 'Y'.
           *
                05  INVMAST-EOF-SW                          PIC X    VALUE 'N'.
                    88   INVMAST-EOF                                 VALUE 'Y'.
           *
                05  INVMAST-EOF-SW                          PIC X    VALUE 'N'.
                    88   INVMAST-EOF                                 VALUE 'Y'.
           *
           01   PRINT-FIELDS               COMP-3.
     |...+....1....+....2....+....3....+....4....+....5....+....6....+....7..
           *
                05  SPACE-CONTROL          PIC S9.
                05  LINES-ON-PAGE          PIC S999 VALUE +55.
                05  LINES-ON-PAGE          PIC S999 VALUE +55.
                05  LINES-ON-PAGE          PIC S999 VALUE +55.
                05  LINES-ON-PAGE          PIC S999 VALUE +55.
  * * * END OF FILE * * *

==== _
                                                             X E D I T  1 FILE
```

first, then duplicate it and make minor changes to the duplicates.

In part 1 of the figure, I have keyed in the prefix command

''3

in the prefix area of the last line of the file. (The command 3'' would work as well.) When I press the enter key, the in-

FIGURE 9 *PART 1 and 2*

The MM prefix commands delimit the block of text to be moved and the F prefix command identifies the line after which the block will be inserted.

```
  LISTMAST COBOL    A1  F 80  TRUNC  72 SIZE  236 LINE  52 COL  1 ALT  0

                 01  DATE-AND-TIME-FIELDS
                 .
MM                   05  TODAYS-DATE              PIC X(6)
                     05  TODAYS-DATE-R REDEFINES TODAYS-DATE
                         10  TODAYS-YEAR          PIC XX
                         10  TODAYS-MONTH         PIC XX
MM                       10  TODAYS-DAY           PIC XX
                     05  TODAYS-TIME              PIC X(8)
                     05  TODAYS-TIME-R REDEFINES TODAYS-TIME
                         10  TODAYS-HOURS         PIC S99
      | . . 1 . . . 2 . . . 3 . . . 4 . . . 5 . . . 6 . . . 7
                         10  TODAYS-MINUTES       PIC XX
F                        10  FILLER               PIC X(4)
                 .
                 01  COUNT-FIELDS                 COMP-3
                 .
                     05  RECORD-COUNT             PIC S9(3)        VALUE ZERO
                 .
                 01  INVENTORY-MASTER-RECORD
                 .
====
                                                            X E D I T  1 FILE
```

The block of text has been repositioned.

```
  LISTMAST COBOL    A1  F 80  TRUNC  72 SIZE  236 LINE  47 COL  1 ALT  1

                     05  SPACE-CONTROL            PIC S9
                     05  LINES-ON-PAGE            PIC S999 VALUE - 55
                     05  LINE-COUNT               PIC S999 VALUE - 99
                     05  PAGE-COUNT               PIC S999 VALUE ZERO
                 .
                 01  DATE-AND-TIME-FIELDS
                 .
                     05  TODAYS-TIME              PIC X(8)
                     05  TODAYS-TIME-R REDEFINES TODAYS-TIME
                         10  TODAYS-HOURS         PIC S99
      | . . 1 . . . 2 . . . 3 . . . 4 . . . 5 . . . 6 . . . 7
                         10  TODAYS-MINUTES       PIC XX
                         10  FILLER               PIC X(4)
 _                   05  TODAYS-DATE              PIC X(6)
                     05  TODAYS-DATE-R REDEFINES TODAYS-DATE
                         10  TODAYS-YEAR          PIC XX
                         10  TODAYS-MONTH         PIC XX
                         10  TODAYS-DAY           PIC XX
                 .
                 01  COUNT-FIELDS                 COMP-3
====
                                                            X E D I T  1 FILE
```

dicated line is duplicated three times, as you can see in part 2.

You can also use '' as a block command. To do that, just mark the beginning and end of a block to be duplicated with paired double quotes. The duplicated lines are then inserted right after the original block. With the block form of the '' command, you can also specify the number of times the marked block should be duplicated. To do so, just code the number of duplicates to be made before or after the set of paired double quotes that marks the beginning of the block.

To understand how to duplicate a block of text multiple times, look at Figure 8. In this case, I have entered the prefix commands necessary to make four duplicates of the marked group of three lines. In the prefix area of the first line I wanted to copy, I entered

''''4

and I keyed in

''''

to mark the last line of the group. The paired double quotes identify the operation I want to perform as a block duplication and the number I specified on the first of the prefix commands indicates the number of times I want the operation performed. When I press the enter key, the duplication is performed and the result appears in part 2. Notice that the operation did not affect the current line pointer. The other two groups of lines involved in the duplication operation are in the part of the file that is immediately ''above'' the segment that is displayed.

How To Copy Or Move Lines: the C, M, F and P Prefix Commands

The easiest way to copy lines in a file is to use the '' prefix command to duplicate them. However, if you want the duplicated lines to appear in the file in a position other than immediately after the source lines, you need to use other commands. That is because you need to identify not only the source lines, but also the destination of the lines. In this section, you will learn how to use four prefix commands to perform these functions: C, M, F and P.

The C (for copy) and M (for move) prefix commands let you mark lines that will be copied or moved from one location to another. They work in the same way except the M command causes the original lines to be deleted after the operation is complete while the C command leaves them intact.

When you use either the C or M command, you have to identify the destination for the lines you want to copy or move. You do that with the F or P command. The F command tells XEDIT to insert the lines you are copying or moving after (''F''ollowing) the line you mark with it. In contrast, if you use the P command, XEDIT inserts the lines you are manipulating before (''P''rior to) the line you mark with it.

Figure 9 illustrates how to use these commands together. In the first part of the figure, I have entered a pair of MM commands to mark a block of five lines I want to move. (I also could have entered the individual prefix command

M5

or

5M

on the first line of the block to be moved.) In addition, I entered an F prefix command to indicate the line after which the block should be inserted. When I pressed the enter key, the block was moved as you can see in part 2. The copy command works the same way, except it leaves the marked block in its original position as well as inserting it in the new location.

When you copy and move text, a complete operation requires that you identify the text to be moved and its destination. In part 1 of Figure 9, I was able to meet both requirements by keying in prefix commands on a single screen. Often, however, the block of text you want to move or copy spans two or more screens or the location of the destination point is farther than one screen away from the block you are moving.

That is no problem because XEDIT can keep track of an ''in progress'' copy or move operation. You can mark a line for the start or finish of the block to be copied or for the destination, then scroll and complete the required entries. When you do, XEDIT's status area contains a message that tells you what is happening. For example, if you mark a block of text to be moved but do not identify the destination, the message

'M' PENDING . . .

(if you are using an individual M command) or

'MM' PENDING . . .

(if you are using block commands) appears in the status area. You can then scroll forward or backward to find and mark the insertion point for the text. You can also mark the destination point before you mark the text to be copied or moved.

When you do that, the message

'F' PENDING . . .

or

'P' PENDING . . .

appears in the status area.

How To Set the Current Line Pointer: the / Prefix Command

Another way to change the current line (other than scrolling) is to specifically identify the line that should become the new current line. To do that, just enter the prefix command

/

in the prefix area of the line you want to become the new current line.

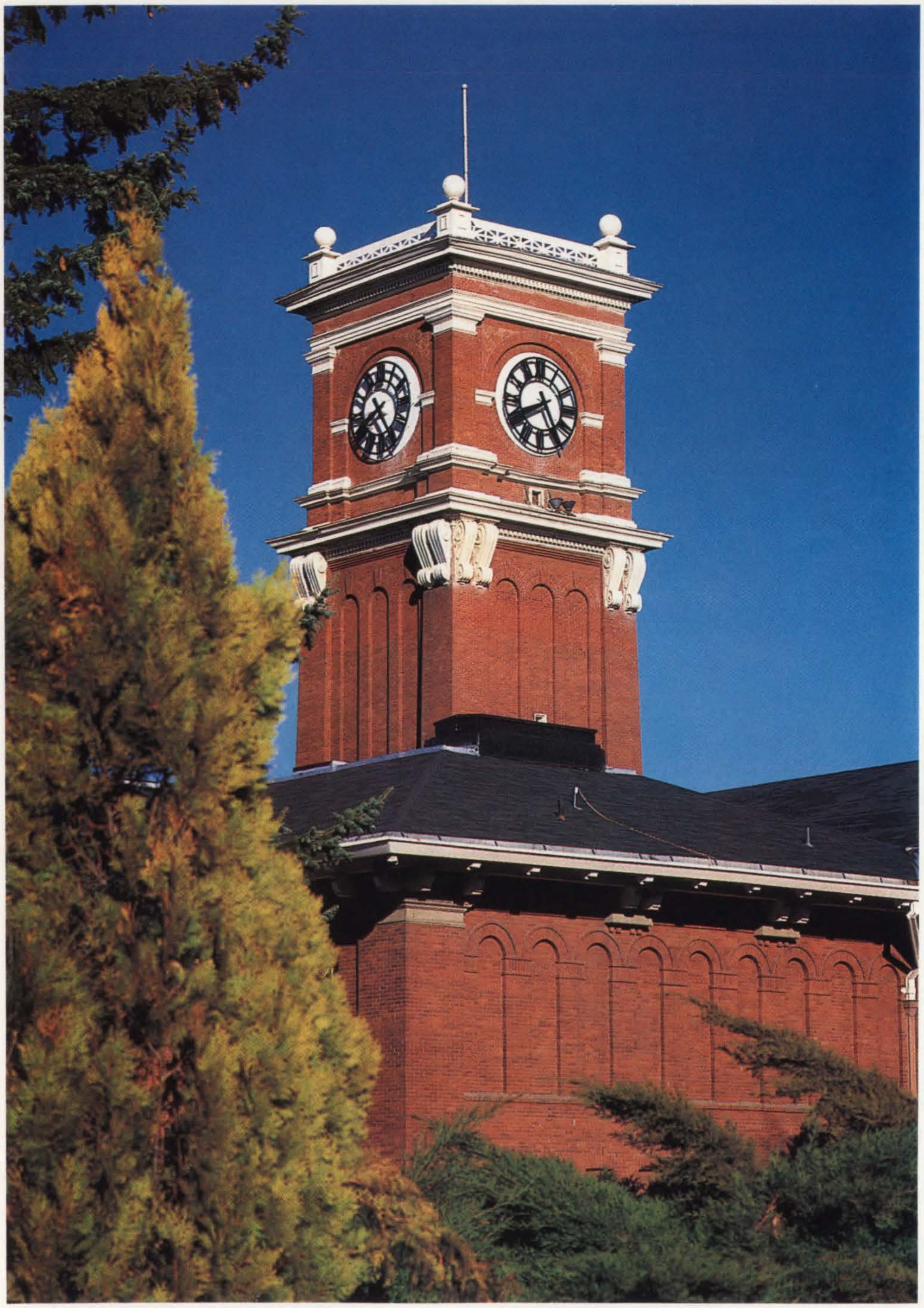
The next article will continue to teach you how to use XEDIT.

This article is an excerpt from Eckols' book titled VM/CMS: XEDIT Commands and Features, *published by Mike Murach & Associates, 4697 W. Jacquelyn Avenue, Fresno, CA 93722, (800) 221-5528; in California (800) 221-5527.*

ABOUT THE AUTHOR

Since 1980, Steve Eckols has written nine books on IBM mainframe subjects ranging from system development to VSE to IMS to VM.

Washington State University Adopts CASE Strategy



By T. Marc Graham

Steve Watson, then assistant director of computing at Washington State University (WSU), remembers well the point in the late 1970s at which rethinking the University's computing operations became priority number one. "It was when an executive vice president of the University made it quite clear that he wanted to be able to ask the same question to multiple offices and get the same answer."

Now, thanks to an integrated, shared database accessible throughout the institution, WSU personnel receive timely, accurate data that makes it easier to meet the academic and business demands of a major university. The school has accomplished this revolution largely through early adoption of a Computer Aided Software Engineering (CASE) computing strategy.

A Major State Computing Center

Located in the small college town of Pullman in rural eastern Washington, Washington State University has a full-time student enrollment of about 16,100 and employs about 5,000 people. A major research and land grant institution, the University also has three branch campus locations in Spokane, Richland and Vancouver and supports a statewide interactive video network called the Washington Higher Education Telecommunications System.

For the past 15 years, the University has also been a state computing center serving multiple state agencies. The Western Library Network is the largest non-WSU agency hosted by the school.

The University's computing needs are met by an IBM 3090-300E mainframe running both MVS and VM. DEC represents the largest minicomputer vendor presence; the University also has several AT&T, Wang, Prime and Hewlett-Packard processors installed. Scattered throughout the University are hundreds of specialty workstations and thousands of IBM, Apple Macintosh and IBM-compatible microcomputers. In total, more than 2,000 workstations reside on the network; the average number of simultaneous users is 400.

Tending to the University's computing needs is the Administrative Computing Department headed by Steve Watson, now WSU's director of administrative and academic computing. The administrative department employs roughly 45 programmers, some on a contract basis. These programmers oversee a main production database that daily receives an average of 4.5 to 6 million calls and processes some 140,000 on-line transactions during the prime shift.

Before 1978, WSU, like most other organizations, relied mostly on batch processing applications. These applications were built independently for the University's various administrative and business units, an approach that eventually became unwieldy and created problems of data redundancy.

''If a student wanted to register a change of address, for example,'' notes Dave Ostrom, WSU's manager of administrative systems, ''he might have had to do so at three or four different places, since the University maintained several different address files.''

The imposition of new state and federal requirements, particularly in the area of student financial aid processing in which new regulations made it necessary for both the University's admissions and financial aid offices to share student files, heightened the need for an integrated database.

First Steps Taken Toward Integrated Database

Administrative Computing began its plans for an integrated database in 1978 by creating a high-level enterprise model, based on an early form of entity-attribute data modeling and data flow diagrams for process modeling from which to sketch a logical information architecture. Even at that early stage, the department knew that the quality of its information architecture would largely determine the success of its future applications.

While the department originally wished to implement the new, integrated database in a single stroke, it soon became apparent that such a strategy could not work. Therefore, to accommodate the ongoing demands of the University, Administrative Computing decided to pursue two parallel courses: it would build applications on an ad hoc basis as it did before to satisfy day-to-day business needs while simultaneously constructing a central data dictionary and the accompanying information architecture.

The department also made the significant decision to develop applications based on their business function within the University, rather than on organizational structure or administrative procedure. Eventually, the department pursued more detailed data modeling through the use of entity-attribute modeling and other techniques. However, because the department's needs had in fact preceded development of appropriate CASE tools, this modeling was accomplished entirely by hand.

WSU took the first step toward an integrated database in 1979 with its decision to implement ADABAS, the database management system from Software AG of North America (Reston, VA). ADABAS was chosen after an evaluation of several products. ADABAS ''represented the best overall combination of flexibility and performance,'' according to Jim Hill, the University's manager of database and on-line systems. Following the purchase of ADABAS, WSU also implemented NATURAL, Software AG's Fourth Generation Language (4GL).

While WSU's programmers first met NATURAL with, in Hill's words, ''some resistance,'' they learned to use it rather quickly — ''some as fast as a week or so.''

NATURAL immediately gave the University an ''interactive development environment,'' Hill says. Doing that greatly enhanced the speed with which the University could develop applications.

''With COBOL, we often looked at weeks, months or even years, for application development. With NATURAL, programmers could prototype quickly and easily, so we could actually talk about development in terms of days or weeks.''

While ADABAS and NATURAL formed the cornerstones of WSU's computing strategy — the University today has more than 5,300 NATURAL programs — they are but two of the more than 20 Software AG products currently used by the University to construct and deliver an integrated database on-line to users throughout the University. In fact, the ability of Software AG products to allow Administrative Computing to create on-line, real-time applications has been central to their popularity at the University, where information such as up-to-the-minute class enrollment and current department expenditures is often in immediate demand.

In 1985, Administrative Computing followed up the integrated database strategy with a strategy of allowing users to download data directly to their PCs. To accomplish this goal, the department made innovative use of Software AG's NATURAL CONNECTION, a mainframe-to-micro link, as a transparent download feature of on-line applications. Currently, more than 90 workstations download material, via a menu interface, from all aspects of the University: department budget statements, salary expense data, class lists, student records and a maintenance department job inventory.

Early Adoption Of CASE Principles And Practices

By the end of 1982, WSU had demonstrated a rather strong adherence to CASE principles even though the CASE acronym had not as yet been coined.

WSU was engineering a CASE environment by designing a data-driven architecture and conducting substantial business analysis of its operation prior to implementation of the database. By planning for enterprise-wide information and applying the ''best available'' engineering principles and concepts, WSU had built a ''homegrown'' system to facilitate front-end modeling and database design before the CASE tools became available from software vendors. The first vendor-supplied CASE tool adopted by the University was a process modeling tool from Optima, Inc. (Schaumburg, IL) called Brackets.

Brackets functions as a word processor for Warnier-Orr diagrams which outline the logic of processes. The tool proved useful in many design and analysis situations. It also aided application development as WSU discovered that end users do understand the diagrams and could help detect errors in the logic of specifications.

In its pursuit of a CASE environment, WSU has followed two key strategies: support of a central dictionary and emphasis on placing as much development activity as possible on the PC while still maintaining a ''single system image.''

WSU first creates dictionary definitions on a PC platform and not the central data dictionary for several reasons. First is the University's strategy for doing as much development as possible with the PC. Second is the greater freedom provided by the PC's dictionary building facility during the early stages of analysis. Dictionary definitions are ultimately uploaded to the University's primary corporate dictionary, from Manager Software Products' (Lexington, MA) DATAMANAGER.

Once data entities in the DATAMAN-

See CASE Strategy *page 92*

CMF Analysis

Management Concerns Addressed With CICS' Monitoring Facility

By Dale Doolittle

Most CICS shops tend to be volatile and dynamic environments, ever changing. Conversions occur; releases and upgrades are constant. Decisions have to be made regarding vendor monitoring and collection systems. Management must also determine the structure of its on-line systems. I believe strongly in providing one *picture* to the end user. Therefore, in my particular shop, all regions interact. This interface has to handle terminal, application and database regions as they are added, changed and deleted. New applications are implemented and existing applications are constantly being modified. PTFs, APARs and *fixes* are never-ending. As all these changes occur, management naturally has concerns. A performance analyst will find it necessary to analyze CICS transaction data at the detail level to answer these concerns and with each analysis a different view of CICS activity will be revealed. Detail data, such as found in the CICS Monitoring Facility (CMF), is necessary in these analyses.

This article will discuss concerns expressed by management over the last two years and the data and variables in CMF records that helped address those concerns. Many of the issues involved response time, MRO/ISC considerations, Logical Unit type 6.2 (LU 6.2) problems, line traffic, wait times, total transaction tracking and user information.

Response Time

By far the most common request for information by management involved response time. One must distinguish response times as either network (end-user) response or internal (host) response. CMF addresses only the host internal response. However, CMF data can be used to calculate an estimated end-user response using character counts and known line speeds. The next two topics, MRO/ISC considerations and LU 6.1/LU 6.2 also reveal insight into response time issues.

The best measurement of end-user response time would probably be the difference between:

- The time the terminal user hits ENTER to let the system know input is ready to be processed
- The time at which results of the processed transaction are displayed to the end user.

End-user response time includes the time to send a transaction and receive the data as well as the host response. Host response can be defined as the time that CICS is first aware of the transaction until the time it has completed processing and data is ready to be shipped back to the user.

SMF110 records do not carry a field called host response time. It must be calculated. The equation for host response seems a simple one:

RESPTME = ENDTIME − START

This is the elapsed time of the transaction; the time that the transaction existed.

It must be remembered that for non-conversational and pseudo-conversational transactions (those that do not issue terminal read requests), the elapsed time is a valid measurement of host response time.

Conversational transactions *do* issue terminal read requests and thus they are *suspended* and put in a terminal wait state as they wait for a response from the terminal user. Therefore, the elapsed time is not a true reflection of host response. The Terminal Control Wait (TCWAIT) must be removed to reflect accurate response time. This could be classified as a third definition of response time and IBM does just that, calling this IRESP.

To show the *best case* of computing end-user response time using CMF, you would calculate the host response time using either method described above, then acquire an estimated line time for characters in and out and add that time to the host response. For example, I transmit at 4,800 baud or 4,800 bits per second. If a transaction transmitted 2,000 bytes in and out or 16,000 bits divide 16,000 by 4,800 to arrive at a transmission time of 3.3 seconds. More information on character counts is described later in the Line Traffic section.

Management should understand the differences between transaction types and their response times. You must balance effective utilization of resources (network configuration, CPUs, DASD) with response time necessary for users to be productive. Tuning both the network and the host is necessary to provide the optimum response to the end user.

MRO/ISC Considerations

InterRegion Communication (IRC)

covers two ways in which CICS can communicate with other systems: MRO and ISC. CICS MultiRegion Operation (MRO) allows communication between CICS systems that run on the same host processor. InterSystem Communication (ISC) requires a link through VTAM. It connects CICS systems on different CPUs and allows them to communicate.

When calculating response time in an MRO/ISC environment, the equation gets complicated. True response time depends on several factors. Is the transaction conversational? MRO or ISC? Are you using LU 6.1 or LU 6.2? Your CICS systems programmer can help answer these questions using the TCT tables internal to CICS. As mentioned above, conversational transactions are placed both in a terminal wait condition and also on a CICS suspend chain. In order to determine the actual time it took to process the transaction, subtract the lesser of the suspend time and terminal wait time from the elapsed time. Thus, the formula for response would be:

RESPTME = ELAPSE − min(SUSPTME,TCWAIT)

A shop should be careful in setting up its transactions. It will need to decide which transactions will be considered conversational and which will be classified as MRO or ISC. A method of tracking MRO and ISC transactions from region to region is described in a later section.

LU 6.1/LU 6.2

CICS uses two SNA protocols for processing ISC transactions: LU 6.1 and LU 6.2 better known as Advanced Program to Program Communication or Advanced Peer to Peer Communication (APPC). Both can be used for data communication between processing systems.

One major difference between these two is that under LU 6.2 a terminal that is owned by one CICS system can run transactions owned by other systems. APPC is also an architecture that sees a terminal as a whole CICS system; it is a device-level concept in which the terminal supports a single APPC session.

Our CICS shop is not running under LU 6.2 due to outstanding problems, although we are aware of other shops that are using it. IBM feels you must use LU 6.2 for cross-system communication. However, we have found LU 6.1 to be sufficient when cross-system communication has been needed.

One problem when going to LU 6.2 was that the response times looked almost *too* good. The suspend wait field included time that was not really suspend time. The performance database system was designed to use the equation as mentioned above: that is, subtracting the lesser of the suspend time and the wait time from the response time. This was incorrect in our situation because we really had no conversational transactions.

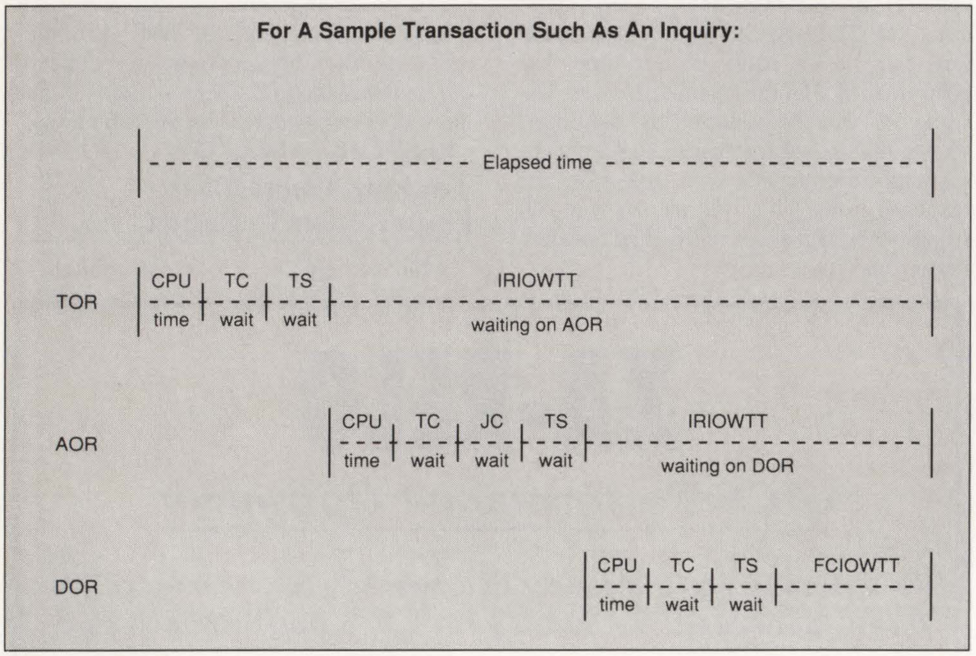
In order to obtain valid host response, we had to change not only our own SAS programs which read the SMF 110 records, but also the vendor source code of our performance database system to use the simple calculation as mentioned above:

RESPTME = ENDTIME − START

Tracking Transactions From Region To Region

Management feels it is not enough to

FIGURE 1



know response time for a transaction from just one region to the next. One must understand the components of response time. Terminal regions will give you complete response time. In order to understand the full host internal response time, it is necessary to track the transaction as it began in the Terminal Owning Region (TOR), routed to an Application Owning Region (AOR), then accessed to a Data Owning Region (DOR) and the return trip back to the TOR. This is needed to define the processing requirements of the transaction such as CPU, number of database calls and access method counts.

CICS 1.7 created two new fields in CMF which allow you to perform this valuable feat. They are NETNAME and UOWID. Unit-Of-Work-ID (UOWID) is a value that is time derived. NETNAME, the VTAM ID, is the name by which the CICS system is known to VTAM. As mentioned before, under LU 6.2 (APPC) a terminal is seen as a whole CICS system.

These two fields make up a unique *key* to connect all routings which actually make up one transaction. You can link all transaction activity to the originating transaction and consider it as one Unit-Of-Work (UOW). In attempting to link transactions using these two fields, another interesting problem was found. NETNAME contained only the APPLID, not the VTAM ID or unique terminal ID. IBM released an APAR on March 30, 1988 to correct the problem. The APAR # is AL22115; the PTF to fix the APAR is # UL27326.

Wait Times

When management sees some of the response times as discussed in the previous section, it demands to know just where all that time was spent. CMF gives eight different breakdowns of times exclusive of the start and end times:

USRDISPT	User Dispatched Time
USRCPUT	CPU Processor Time
TCIOWTT	Terminal Control I/O Wait Time
FCIOWTT	File Control I/O Wait Time
JCIOWTT	Job Control I/O Wait Time
TSIOWTT	Temporary Storage I/O Wait Time
SUSPTIME	Suspend Chain Time
IRIOWTT	InterRegion Wait Time

By using these fields, a graphic presentation can easily show where a transaction spends its time. Looking at Figure 1 also might help to understand these wait times.

Line Traffic

CMF provides data on message counts and sizes that help analyze changing networks for line traffic demands and trends. The workload might remain consistent and just grow in size. Then new workloads could be added. The number of on-line terminals increases. Your shop upgrades its CPU or gets a new system. Screens become larger and denser. Line compression is added. More information is passed. On-line report viewing is needed.

There are eight fields relating to line counts in the SMF 110 record shown in Figure 2.

Since the source code of our database system added the principal and alternate fields together for their summarized databases, we did the same in our detail programs. However, in comparing our line traffic numbers to NLDM statistics, our sizes were just too large and the network people said there was no way the lines were handling that kind of traffic.

F I G U R E 2

TCMSGIN1	TC Input Message Count — Principal Terminal Facility
TCMSGIN2	TC Input Message Count — Alternate Terminal Facility
TCMSGOU1	TC Output Message Count — Principal Terminal Facility
TCMSGOU2	TC Output Message Count — Alternate Terminal Facility
TCCHRIN1	TC Input Character Count — Principal Terminal Facility
TCCHRIN2	TC Input Character Count — Alternate Terminal Facility
TCCHROU1	TC Output Character Count — Principal Terminal Facility
TCCHROU2	TC Output Character Count — Alternate Terminal Facility

The answer comes in understanding the *principal* and *alternate* fields.

- Principal fields represent the actual characters transmitted from the terminal to the host and then the characters received back from the host at the conclusion of a transaction.
- Alternate fields are larger as they include the principal count plus control characters required by IRC to pass the transaction from region to region. Just how much larger is not consistent because various factors, such as buffer size, come into play.

Both are necessary and important fields. For actual line traffic purposes in the user network, the principal is all that is needed. When analyzing system-to-system sizing you should also consider the alternate. For example, when your regions are remote, the IRC characters passed back and forth should be counted as line traffic. We have since learned that our performance database vendor plans to create new character and message count fields and separate the principal and alternate fields.

Another consideration in dealing with line traffic is understanding the effects of line compression. Several products are available which allow optimal use of network lines. These optimizers transmit only the characters needed. CMF will not reflect this compression. You can, however, estimate the compression factor and then the CMF data can still be valuable.

Storage Fields

Storage is always a concern to management. Questions arise. How much are we using? Do I need to break up this region? What are my limits? How many more terminals can I put in this region? How many programs, and what are their sizes? Can I add more programs to a region? What size storage *cushion* do I need to maintain? Do I need storage compaction?

CICS 1.7 CMF records have three fields that help measure storage:

SCUSRHWM	Maximum amount of storage allocated to the user-task. This is in bytes and reflects the high-water mark that this application took in storage.
PCSTGHWM	Maximum amount of program storage in use by the user-task. This is in bytes and reflects the storage needed to hold the application source code.
SCUSRSTG	This is defined in (6) as *the storage occupancy of the user-task*. It reflects an average amount of storage that the user-task consumed over its elapsed time.

It has been helpful to review these storage statistics and surprising at times. Some application region programs can be immense and when you consider many applications running at one time, the total storage use is a major concern especially with the limited private area. The new ESA architecture should alleviate some of this problem but will it be in time?

USERID Information

The significant value of a new field, USERID, depends on your company's standards policy. By the implementation and strict enforcement of standards that control how the USERID is assigned and managed, CMF can provide you with a field that can represent location, cost centers and groups for analysis and billing purposes to mention only a few. The information in USERID can go beyond the standard fields provided to identify the source or user such as Terminal ID and Transaction ID.

It is a fair assumption that many companies have user information they would like to relate to the detail CMF records. In our case, we wanted to relate the transaction mix and resource consumption data to the banks and branches we support.

CICS 1.7 automatically provides us with data we had to maintain in user fields under CICS 1.6.1. USERID is an eight character field that can be extremely beneficial to specific user concerns: it uniquely identifies the user.

Summary

There is a vast amount of information in the SMF110 record that can answer management concerns. This article has only begun the exploration. For example,

See CMF Analysis *page 92*

SQL/DS In Full Production

By Roberto Bernik, Robert L. Herbst, Joseph A. Hendrickson and Augusto Ramirez

Predictions of ease-of-use systems that would require more memory and disk storage to operate have come true concurrently with the lower cost per unit of hardware. These ease-of-use systems first appeared as the third-generation network and hierarchical database management systems. However, it was not until the fourth generation, led by the birth of the relational model, that the promise of productivity was fulfilled. The mystery software that would take advantage of the new hardware is here today in the IBM Program Product SQL/DS.

The success and acceptance of SQL/DS is not surprising. Personnel costs, in what is a labor intensive process, are rising. The demand for senior technicians is exceeding the supply. The United States Department of Labor statistics confirms this fact already perceived in the data processing industry on a daily basis.

This article concentrates on the SQL/DS product installed in a production capacity and on the degree of software fine-tuning required to achieve performance objectives. *Full production,* for the purpose of this article, is defined as SQL/DS being used as the primary organization-wide database for daily operations in both on-line and batch processing.

Even though the larger IBM 9370 and 4300 processors together with faster disk storage devices offer improved processing, a significant degree of software fine-tuning is required to achieve throughput expectations.

The objective of this article is to review the fundamental requirements for perfecting an SQL/DS environment ready for efficient, full-production operation. The conclusion is that certain upfront efforts are required to maximize the performance and benefits of SQL/DS.

SQL/DS is discussed from the vantage points of Operating Environment, Database Organization, SQL/DS Parameters, Programming Considerations, Database Administration, Operations and Hardware.

Operating Environment

It is important to carefully assess system software plans in advance before investing costly systems engineering efforts on trial-and-error alternatives. Full production is the wrong place to rethink system software strategies.

Since SQL/DS is memory intensive and virtual memory is a limited resource, virtual memory utilization must be carefully understood when planning the operating environment.

Ideally, to minimize software overhead SQL/DS should be placed in the same address space as CICS/VS. However, due to the requirements for virtual storage, this is not practical. Instead, SQL/DS can be combined in the same address space with batch SQL/DS partitions. It is desirable to keep the partitions that will use SQL/DS in the same address space to reduce the overhead involved in crossing address spaces. The use of POWER and XPC (Cross Partition Communication) involves overhead and adds to delays.

System files should be monitored frequently to avoid file contention. The placement of page datasets, POWER files, label areas and the lock file are critical.

The *fast CCW translation* option for SQL/DS should be turned off at the job level since all DASD addresses are already pre-mapped into the SQL/DS directory.

Interactive SQL/DS (ISQL) is a tremendous productivity tool. However, it is important to keep the number of ISQL users under control. The fact that ISQL is interpretive and its use of *agents* determines that it requires substantial resources and therefore makes it extremely susceptible to abuses in terms of database accesses that can be generated from improper queries.

The SIZE parameter of the EXEC

statement should be utilized when compiling programs with embedded SQL/DS code. However cumbersome, it is more efficient to use POWER facilities (DISP = I on the PUNCH statement) instead of employing system files (SYSPCH and SYSIPT) on disk.

The deactivation of partitions that use SQL/DS facilities must be avoided because SQL/DS locks are held while the partition is deactivated. A helpful facility for monitoring the status of resource utilization on a system-wide basis is the system status display under the Interactive Interface of VSE/SP.

Database Organization

All addressing under SQL/DS is handled at the page level. A *page* is defined as a 4K area of disk or memory space. This is consistent with both VSE/SP and VM/SP virtual memory mapping structures, so storage fragmentation is not an issue.

A particularly strong feature of the SQL/DS product is its locking conventions. By providing sound locking mechanisms, both integrity and synchronization are more than adequately provided for.

The level of locking granularity (row, page, DBSPACE) is defined at the DBSPACE level and applies to all tables residing in that DBSPACE. This decision affects the lock escalation process and overhead and should be defined based on application and data integrity needs. Incorrect application design and locking level decisions can cause locking escalation and adversely affect data access.

The *lock table* is a reserved area of main memory within SQL/DS that handles all read/write locks. It should be kept as small as possible since the Dispatcher scans this table frequently. The dispatching algorithm is one of the most frequently used pieces of SQL/DS code. It decides the order of the tasks to be performed based on prerequisite and corequisite resource demands.

Concurrency is another factor that can greatly reduce the overhead of SQL/DS pages. The placement of tables within a DBSPACE is an important factor. Multiple tables sharing the same pages by interleaving data can drastically save I/O time, especially when two tables are frequently referenced together as with JOIN operations. This can be accomplished by using the free space parameter to force this condition at table load time. A sound understanding of the data and how it will be used is required to accomplish this successfully. Caution must be exercised since the results of misusing this feature can be much worse than the best benefit.

The most dramatic single performance aspect of SQL/DS is the proper use of indexes. All tables should have at least one index and should be physically clustered by one index. If there is more than one index, the first index created becomes the governing index used for physical organization of the data on disk.

For read-only tables, several indexes may be defined since no index maintenance is involved. Conversely, data with frequent UPDATE, DELETE and INSERT activity should have as few indexes as possible to minimize the overhead of index maintenance.

Temporary indexes can be created only for the duration of a specific job when the existing index or indexes do not satisfy the requirements of the search. Creating such an index on a permanent basis does not justify the overhead of index maintenance. This is particularly useful for infrequently executed jobs and can be accomplished by creating a temporary index, reprepping the affected programs and dropping the index after execution.

Permanent indexes should represent the columns most frequently used in WHERE, JOIN and ORDER BY clauses. Preferably, columns that are modified frequently should not be part of an index. One reason for this is that SQL/DS will not use an index if it contains columns to be updated.

Random inserts for tables require that free space be left during table loading to preserve clustering. Conversely, if it is known that inserts will not be done, no free space need be allocated when the DBSPACE containing such tables is created. Users should be cautioned that the percent free space specified turns out to be only approximate, since SQL/DS uses a free class table in deciding how table pages should be loaded and the resulting number of rows per physical page can be quite different than anticipated. For example, if a table with an average row length of 100 characters is loaded with the percent free defined as 50, each page will contain only one row as opposed to the expected 20.

To minimize contention, user data should not be placed in DBSPACEs that contain *system catalogs*. Additionally, internal DBSPACEs should reside in their own storage pool to avoid short-on-storage conditions.

Logical Units of Work (LUW) in the applications are also important. LUWs should be synchronized, consistent with the processing objective and kept as short as possible.

DASD balancing should take into consideration the distribution of the database onto different physical devices to enhance SQL/DS performance and also to reduce contention with the DASD resources used by other Program Products or application systems.

SQL/DS Parameters

There are several options available at SQL/DS startup time. This section discusses these various options and their impact on performance.

Application programs request data from SQL/DS through the Resource Manager. In turn, the Resource Manager dispatches an *agent* to process the data access request.

The Number of Concurrent Users parameter (NCUSERS) determines the number of agents allocated to SQL/DS. If this number is too low, users will wait longer, response time will increase and concurrency will be reduced. Concurrency can be increased, of course, by increasing the number of agents. However, overallocation of agents can cause increased storage demands, excessive locks and buffer contention within the SQL/DS partition. The number of agents should take into consideration the desired degree of concurrency versus the associated overhead involved. A good approach is to start with a low number of agents and then increase them judiciously.

The Dispatcher Bias Algorithm (DISPBIAS) ranges from one to 10, where one services agents on a round robin basis and 10 favors short LUWs over long ones.

The Number of Lock Request Blocks (NLRBs) should be set based on the worst transaction, since it influences the size of the *lock table*. Blocks are counted in both per-user and in global terms. Usually, two lock request blocks are required for each lock request. Since SQL/DS issues many lock requests, locking done at the row level can translate into a large number of blocks, making the Lock Request Table unnecessarily large. By specifying locking at the page level (default), the size of the *lock table* can be controlled. If too many lock request blocks are allocated, the internal structures will grow and may adversely affect paging.

Buffers are an area of virtual memory used to cache SQL/DS data accesses. Data and directory information (NPAGBUF and

NDIRBUF) are buffered so that subsequent requests for the same data will trade an I/O to disk storage for an access to memory. The ratio between disk I/O and the number of buffers accessed can be monitored with the use of the SQL/DS COUNTER command. Unfortunately, there is no formula to determine the optimum number of page and directory buffers. Again, it is best to start low and work up.

SQL/DS provides for logging modifications made to the database and for archiving the log for emergency rollback situations. Logging can be accomplished with one log file or with two identical copies (DUALLOG) of the same log file. Dual logmode provides an extra safety factor in the event either log is destroyed. However, there is additional overhead involved, since every entry must be written on both log files.

There are several modes of log operation. The first mode (LOG = A) requires a log file large enough to hold all the database activity for any given day. When the log file is almost full, SQL/DS will suspend processing while it performs an archive on tape of the database which would be detrimental during on-line activity. This mode allows for forward recovery of the database. The second mode (LOG = Y) requires a substantially smaller log file. In this mode, the log file is reusable and no forward recovery is possible. An extremely small log file may not be sufficient to accommodate long LUWs, typical of batch programs containing only one COMMIT WORK statement at the end of a job. The third mode (LOG = L) of operation automatically archives the log when a predetermined threshold is reached. The last mode (LOG = N), only valid in *single user mode,* deactivates the log entirely.

A *checkpoint* is a marker on the log file identifying a period of database activity that can be recovered in the event of failure and functions as a mechanism to release shadow pages. The Checkpoint Interval (CHKINTVL) parameter controls the frequency with which checkpoints are taken. If the CHKINTVL is low, frequent checkpoints will be taken slowing down the processing. Too high a value in the CHKINTVL can slow down *warm restart* recovery procedures, a somewhat minor inconvenience.

Archives provide for saving all pages in the database changed since the last archive was taken and are used to recover from failures. Each installation has its own unique requirements relative to the logging, checkpointing and archiving methods that are desirable.

Programming Considerations

The application programmer needs to be properly briefed on certain SQL/DS features to correctly design and code SQL/DS modules. Poor coding at this level can severely impact overall SQL/DS system performance.

In order to protect the integrity of the data, it is necessary to understand the effective isolation level of the data to be accessed. This isolation level is determined when the lock level of the DBSPACE is defined. However, an expected isolation level can be nullified by data accesses that ignore the indexes and perform a DBSCAN.

The price for excessive isolation is a lesser degree of concurrency in which users will have to wait to access data. *Isolation level* is a preprocessor-time consideration. The isolation level options are Cursor Stability (CS) and Repeatable Read (RR). CS only isolates the target row for the I/O, regardless of the LUW. RR holds all pages of target rows until the LUW is completed and provides for reread contingency by the application, guaranteeing that no other processing will update the same data between the first and subsequent reads.

The *block fetch* feature is a preprocessor-time option that provides for interim memory storage of numerous data requests which are transferred in block to the application program. By using this feature, resources are saved in passing data to the application. This feature is only applicable in *multiple user mode*.

The programmer and the database administrator must coordinate their efforts to ensure limited updating of columns that are part of an index. It is more effective to delete and then add the target row when columns to be updated are part of an index.

The SUM verb provides for summing data at retrieval time. If the data is properly indexed, the SUM feature reduces coding. However, improper use of this feature can yield poor performance and it is sometimes more efficient to retrieve target rows and do the summing of data via user coding.

Proper design of LUWs, which should be kept as short as possible, will have a positive impact on SQL/DS performance. COMMIT WORK causes the release of resources and the physical modification of SQL/DS data on disk storage and signals the end of the LUW.

With on-line transactions, the tendency had been to keep each module small and link together as many modules as required to define the logical transaction. It was previously taught that CICS modules should be limited in size to reduce virtual memory resources and increase throughput. Given the way SQL/DS handles agents, access modules, locks, pages, binding and attachment processes, it may be more desirable to group all SQL/DS modules pertaining to one task together in one larger module. The resultant tradeoff in terms of memory can be better than the overhead of agents, log file I/Os, transfer of control and the resultant LUW from module linking. LUWs spanning several SQL/DS modules increase restart/rollback overhead and negatively impact overall SQL/DS performance. In general, longer LUWs increase the potential for *deadlocks*.

A deadlock is the way SQL/DS resolves a deadly embrace situation. When this occurs the youngest request for data will be backed out to release its resources, which are being wasted, since this task is stalled pending a particular locked resource.

Deadlocks can occur due to poor application design. When designing LUWs, it is important to evaluate concurrency and availability of resources. This will help to limit the number of deadlocks in production.

SQL/DS commands can be *static* or *dynamic*. Static commands are commands embedded in an SQL/DS program in which variables are defined but their values are unknown. For static commands, optimization is done at preprocessing time. It is important to note that the Optimizer favors a DBSCAN because SQL/DS assumes a 25-30 percent hit ratio on a DBSCAN. The dynamic approach provides a method of building SQL/DS commands, including values, and presenting them to SQL/DS at execution time. In this case, optimization is done at execution time; however, the execution time will bear the overhead of both Optimizer and Preprocessor, since preprocessing in this method happens each and every time the SQL/DS command is executed.

Preprocessing an SQL/DS module requires exclusive locks on the SQL/DS system tables such as System Access (SYSACCESS), Program Authorization (SYSPROGAUTH), Table Authorization (SYSTABAUTH), Usage Authorization

(SYSUSAGE) and the *access module tables* in SYS000n. If there is an excessive number of SQL/DS compilations, consideration should be given to developing a fast compile partition to reduce the exclusive lock time of these control areas.

Explicit locks of a table or DBSPACE should normally be avoided. In general, the locking of resources should be left to SQL/DS. When an explicit exclusive lock is employed, the transaction issuing it will wait until such a request can be granted, and once obtained, all other transactions seeking to access the same data will wait in turn until the exclusive transaction is completed.

The EXPLAIN command provides a method of studying the structure and performance of SQL/DS commands. Through this method, SQL/DS statements can be prototyped. Vital information about the execution of the statement is stored in four work tables: PLAN, COST, STRUCTURE and REFERENCE. By using ISQL and the EXPLAIN command, prototyping can resolve performance issues during program design.

JOINs can be one of the more resource intensive commands in SQL/DS. Incorrect JOINs can cause DBSCANs of large volumes of data unnecessarily. The EXPLAIN facility is particularly useful for avoiding this circumstance.

Database Administration

The database should be monitored regularly to avoid out-of-space conditions. This can be accomplished by using the Database Services Utility (DBSU) or the SHOW command.

Since SQL/DS utilizes the same storage conventions for its own system data as for user data, DBSPACEs that contain SQL/DS system data can be easily monitored.

The constant adding, deleting and replacing of *access modules* will eventually exhaust the maximum number of entries allowed in access module DBSPACEs. Periodic deletion of unwanted access modules can be performed to relieve this situation.

Due to the manner in which SQL/DS maps data areas, it is impossible to modify DBSPACEs after they have been acquired. It is therefore necessary to plan in advance when acquiring DBSPACEs because they cannot be made larger nor combined to form a larger DBSPACE.

The SQL/DS Optimizer relieves the programmer of the responsibility of physically navigating through the database for accessing data. However, to perform efficiently, the Optimizer requires up-to-date information about the database structures, current status and index component characteristics. The UPDATE STATISTICS function is used to reflect the current state of the SQL/DS database. Since UPDATE STATISTICS causes DBSCANs of the affected DBSPACEs, UPDATE STATISTICS should be performed selectively or on a scheduled basis to reduce database contention. UPDATE STATISTICS at the table level should be performed after major changes to a table and UPDATE STATISTICS at the DBSPACE level after changes to multiple tables in the same DBSPACE.

Reorganization of unclustered tables and/or spaces is extremely important and must be done regularly. Various alternatives, from manual to automatic determination of reorganization needs, are possible. An UPDATE STATISTICS run is recommended before any reorganization procedure, so that valid reorganization decisions can be made. The UPDATE STATISTICS function is automatically performed by SQL/DS when an index is created after a table is loaded.

When loading a table with the PURGE option, SQL/DS deletes each row, employing the *logging* function while retaining all table and index definitions. By DROPping and reCREATING the table, SQL/DS simply manipulates the *catalog* entries in a fraction of the time without the use of the logging function. Since it takes longer to LOAD with the PURGE option, it is better to DROP, CREATE and LOAD the table with the NEW option. In the first case, the index will be automatically rebuilt, whereas in the second case the index must be created explicitly. In order to control excessive interleaving of physical data on 4K pages in data spaces containing multiple tables, it may be better to DROP the entire DBSPACE, reACQUIRE the DBSPACE, and CREATE and reLOAD all the tables contained therein.

It is beneficial to place a large table in a DBSPACE by itself. If this is not done, DBSCANs of other tables in the same DBSPACE will include the scanning of the large table as well.

When utilizing ISQL, *routines* are a better alternative than *stored queries,* since they are easier to modify and are more completely displayed.

Operations

Batch jobs executed during periods of critical on-line activity may affect on-line response time due to contention for data access. The advantage of executing batch jobs in *single user mode* is that there is no interference with on-line activity and no cross partition overhead. In addition, if logmode is off, there is no logging overhead and no danger of exhausting the log file. The tradeoff is that concurrency is negated in single user mode which may, in fact, be acceptable or even required for certain batch programs, especially those with unusually long LUWs.

Backups of the database are critical and can be performed at the table and DBSPACE level via the UNLOAD command. There are various provisions for preserving data in the event of failures. Database backup, database archive and log archiving are different strategies which can be employed depending on the recovery objectives of each installation.

The COUNTER command displays SQL/DS system-wide statistics and can be invoked through the operator console or via ISQL. The SHOW command provides additional statistical information about current status of DBSPACEs, DBEXTENTs, LOGs, LOCKs and other SQL/DS components. For instance, SHOW LOCK is helpful to obtain more detail when investigating locking problems. The system catalogs are another helpful source of information that can be queried via the SELECT command.

Each catalog is a table containing detailed information about the SQL/DS environment such as Tables, Columns, DBSPACEs, Indexes and Program Usage Profiles.

For instance, the SYSCATALOG table contains information on all tables in the database such as CLUSTERTYPE, number of ROWS, number of PAGES containing ROWS and percentage of active pages. The SYSCOLUMNS table contains information on columns such as LENGTH, NAME and TYPE. The information provided in the SYSDBSPACEs table refers to the number of TABLES, number of usable PAGES, number of PAGES set aside for Indexes, percent free per data page and Locking Granularity. The SYSINDEXES table indicates the Index TYPE (unique or non-unique), percent free per Index Page, CLUSTERED indicator and LOCKMODE. The information provided in the SYSUSAGE table shows program and table dependencies. This information is maintained on both application data and SQL/DS system data.

Certain systems functions such as UP-

DATE STATISTICS, CREATE INDEX and DROP TABLE may require extended execution times because of DBSCANs involved.

Accounting information collects the utilization of resources on a user basis during SQL/DS execution such as initialization records, operator checkpoint records and termination records. User resources accounted for include the number of buffers referenced, CPU time and active time. The accounting option requires a sequential DASD or tape file and can be tailored based on initialization parameters. Accounting information is not meant for performance monitoring.

There are other helpful features available for managing the SQL/DS environment. The TRACE facility is a console command used to determine specific problems. However, it is somewhat difficult to read and it does affect SQL/DS performance. Relational Design Tool (RDT) is a separate IBM Program offering that can be used to recommend indexes for planned or existing applications. The CICS transaction CIRD displays transactions waiting to establish a link to SQL/DS, transactions holding a link and accessing SQL/DS, transactions holding a link but not using it and transactions that previously held a link.

Hardware

One major consideration regarding the development of an SQL/DS environment is a conscientious hardware plan. Raw processor power, I/O capacity and memory are key elements for a successful production SQL/DS facility. However, it should be stressed that hardware upgrades by themselves only provide a temporary solution to performance problems. Application system growth will definitely bring again to the surface the problems caused by a poorly tuned SQL/DS system. SQL/DS tuning efforts yield longer lasting results than hardware enhancements alone.

The history of data processing has proven that any product as flexible as SQL/DS bears a certain cost in terms of system resources. Today's database management systems generate thousands of instructions per logical data I/O request to DASD. Given such a high cost per I/O, it is certainly beneficial to reduce the number of I/Os. This can be accomplished through careful and prudent database organization, sound programming standards and an effective hardware configuration. There are many alternatives with regard to DASD, controllers, processor models and system software configurations that must be seriously reviewed in the planning phase.

Each facility must recognize its own optimal configuration unique to its environment based on business objectives, software profile, budget and software architectural parameters. Because of these diversities, there is no generic hardware configuration that can be recommended for any facility.

Conclusion

SQL/DS is a full production database capable of handling significant, real time updates while providing excellent user terminal response time.

The tuning of SQL/DS is the single most important factor in achieving acceptable performance objectives. This process is somewhat complicated by the number of parameters involved, their numerous rules and roles and the time consuming nature of the process itself. However, a good portion of this work is performed only once during the development cycle.

Tuning SQL/DS has been typically accomplished through basic trial and error. When using this approach, it is important to record all changes made in order to be able to recognize and understand the impact of each one of them, singularly and in combination. Sometimes changes that would have been beneficial may not return demonstrable gains because of other underlying problems. A detailed understanding of the various options available is required in order to meet performance objectives.

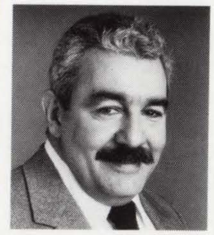
With personnel costs escalating and the unit cost of hardware declining, SQL/DS is an excellent way to lower application development costs through increased programmer productivity.

The non-procedural, non-physical and logical nature of relational technology and the soundness of SQL/DS offers an excellent foundation upon which to base future investments in information systems.

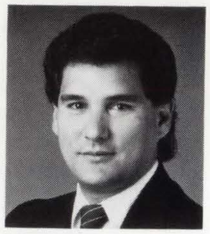
Any facility with a commitment to technical staff and application development should consider application development products that are flexible, promote programmer productivity and reduce personnel costs.

With a detailed understanding of SQL/DS, proper tuning and a realistic hardware plan, real time and production application systems with satisfactory response time can be accomplished using SQL/DS.

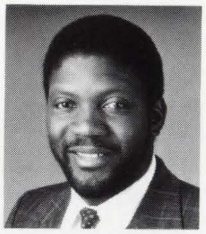
ABOUT THE AUTHORS



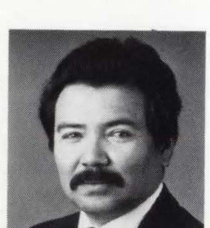
Roberto Bernik has 17 years experience in systems software installation, fine tuning, conversions, planning, maintenance and many IBM VM/VSE Program Products.



Robert L. Herbst has 17 years experience in application systems development including planning, business requirement analysis, design, development and database administration.



Joseph A. Hendrickson has 15 years experience in systems engineering including hardware configuration development, site planning, installations, conversions, performance tuning, telecommunications, transaction processing systems and database administration.



Augusto Ramirez has 12 years experience in application system development including design, programming, implementation, transaction processing systems and database administration.

The four authors are principal partners in the firm, George Tuccio & Company, Inc. GTCO is a full service technical contracting firm providing systems engineering and application development services to IBM mainframe users and specializing in IBM software such as MVS, VM, VSE, CICS, VTAM, SQL/DS, DB2 and CSP. GTCO, 322 W. 57th St., New York, NY 10019, (212) 957-9471.

Expert Systems For Computer Performance Evaluation

MetaRules And Knowledge Engineering

By Bernard Domanski, Ph.D., and Sid W. Soberman

The objective of this article is to explore knowledge representation and knowledge engineering techniques that can be applied when creating knowledge bases for Computer Performance Evaluation (CPE) domains. The concept of MetaRules, rules encompassing other rules, are used to help simplify the complex and cumbersome task of building knowledge bases. Examples are given for an expert system whose domain is performance problems for IBM's MVS/XA operating system.

A *domain* is a clear and definable extent of knowledge about specific subject matter. As the discipline of Computer Science associates data structures with types of algorithms, a *rule-based system* is a representation of knowledge (that is, the data structure) for a particular domain. The rules describe the relationships between facts in the domain. Rules are usually expressed in an IF-THEN form. The *knowledge engineer* must gain knowledge about a particular domain and must represent that knowledge in usable form, for example, a rule-based system. Knowledge engineers build expert systems.

For CPE, knowledge acquisition and knowledge engineering are complex and cumbersome tasks. During the development of a rule-based system for the CPE domain, a knowledge representation was initially formulated. This representation was designed to support the standard expert system *consultation paradigm*. While in the knowledge engineering phase of the development, *changes* had to be made to the knowledge representation to accommodate the peculiarities of the domain knowledge.

Knowledge Representation

Expert systems fall into several different classes. Systems within the same class use the same general type of prototype or *paradigm*. Diagnostic expert systems use the *consultation paradigm*. Users are asked to describe observable characteristics or *symptoms* so the expert system can determine which of several possible problems exists.

In a production system using the consultation paradigm, rules can be viewed in the following form:

```
IF     symptom(A) &
       symptom(B) &
       ...
THEN diagnosis(X) &
       treatment(Y)
```

The fundamental strategy for making diagnoses and suggesting treatments is to ask a user a precise set of questions in a consultation session. The replies are stored as *facts* and these facts are used to fire *rules* in the format above. Rules express the relationships between the facts (that is, the symptoms) and the conclusions (that is, diagnoses and treatments). Inevitably, the expert system reaches a conclusion(s) based on the information available.

In the area of MVS performance tuning, an example of such a rule might be

```
IF    I/O response time > 40 msec
      PEND time > 0.10 *
        (Connect time + Disconnect time)
      IOSQ time > (Pend time +
        Connect time + Disconnect time)
THEN
     diagnosis:
          I/O operations are queued in IOS
          because the device is too busy servicing
          other I/Os from this system. The number
          of I/Os needs to be reduced.
     treatment:
          The number of I/Os can be reduced by
          increasing blocksizes and/or increasing
          the number of buffers.
```

A key aspect in the systems' design is the knowledge representation used; that is, the selection of *objects* and their associated *attributes*. The effectiveness of the system is a function of the type and *order* of the rules used. Of course, a simple yet robust design will aid in the systems' development.

At the first attempt at knowledge representation for the CPE domain, I devised the following simple scheme for objects and attributes (see Figure 1).

The expert system looks for the existence of problems; the *condition* primitive is used to define the set of questions *(set_of_keywords)* that the expert system must ask first. Each *keyword* is associated with a *question_to_ask* using the *prompt* primitive. To diagnose a problem, the *boolean_expression_of_keywords* (a conjunction of positives or negatives) in the *diagnosis* primitive is evaluated. If the expression is true, the expert system assumes that the *problem* exists. The *diagmsg* primitive associates the problem with actual *diagnostic_text* and the *treatmsg* primitive associates *treatment_keys* with the actual *treatment_text* that the expert system displays.

As an example, convert the previous IOSQ time rule into this format (see Figure 2). Note in the example the *boolean_expression_of_keywords* in the *diagnosis* primitive was the conjunction of resp, pend and iosq. This can be read as *resp & pend & iosq*. To represent negation, a hyphen (-) is placed in front of a keyword. The boolean expression in

```
diagnosis   ("problem2", ["symptom_A",
             "-symptom_B"], "treatment_2")
```

is read as *(symptom_A) &-(sympton_B)*.

Another aspect of the basic strategy is that the manner of the dialog is symptomatic. The questions are about symptoms: observations that the user can make. This is true for CPE as well as other domains. There is a limitation in the scheme for knowledge representation. Consider the following compound rule:

```
IF    I/O response time > 40 msec &
      - (VIO is being used for small
        temporary datasets)
                    OR
      I/O response time > 40 msec &
      VIO is being used for sort work files
                    OR
      I/O response time > 40 msec &
      - (cylinder boundary allocation
        being used for VIO datasets)
THEN
      diagnosis:
            I/O operations are queued in IOS
            because the device is too busy servicing
            other I/Os from this system. The number
            of I/Os needs to be reduced.
      treatment:
            negate the VIO symptom(s) found.
```

During knowledge acquisition, the knowledge engineer for this domain will closely examine each potential rule. What if VIO were not being used at all? The given scheme for knowledge representation would force the three VIO questions to be asked (see Figure 3).

Driven by the need to formulate KBAs that are concise while being complete, a new primitive is needed by the knowledge engineer; *depend* allows a question to be asked only if it makes sense in the current context. In the example, first ask, "Is VIO being used?" The expert system will ask the three related VIO sub-questions if the reply is *positive*. However, if the reply is *negative,* the expert system needs to *bypass* asking the three sub-questions (see Figure 4). Thus, *depend* conditionally asks about the presence of symptoms in which the condition is a *boolean expression* of questions previously asked.

FIGURE 1

```
prompt     (keyword, question_to_ask)
condition  (problem, [set_of_keywords])
diagnosis  (problem, [boolean_expression_of_keywords], treatment_key)
diagmsg    (problem, diagnostic_text)
treatmsg   (treatment_key, treatment_text)
```

FIGURE 2

```
prompt     ("resp", "I/O response time > 40 msec")
prompt     ("pend", "Pend time > 0.10 * (Connect time + Disconnect time)")
prompt     ("iosq", "IOSQ time > (Pend time + Connect time + Disconnect time)")

condition  ("IOSQ problem", ["resp","pend","iosq"])

diagnosis  ("IOSQ problem", ["resp","pend","iosq"], "iosq treatment")

diagmsg    ("IOSQ problem", "I/O operations are queued in IOS because the device")
diagmsg    ("IOSQ problem", "is too busy servicing other I/Os from this system. The")
diagmsg    ("IOSQ problem", "number of I/Os needs to be reduced.")

treatmsg   ("iosq treatment", "The number of I/Os can be reduced by increasing")
treatmsg   ("iosq treatment", "blocksizes and/or increasing the number of buffers.")
```

```
depend     (problem,
  [boolean_expression_of_keywords], keyword)
```

Next turn to yet another basic strategy of the dialog process — help. Classify help for expert systems as follows: first, a user may need help in how to navigate through the expert system. For example, on-line help may be available that instructs the user in the use of the function keys or the cursor control keys. This class of help is necessary but has little to do with the knowledge domain (that is, CPE).

The second class of help is fundamental to the knowledge itself. If the user of the system does not understand the question being asked, the expert system should provide a clear, concise explanation of what is being asked. For example, for "Is I/O response time > 40 msec," help would consist of a brief explanation of the components of I/O response time. In addition, help should also direct the user to the measurement source (that is, an RMF report) so (s)he can find the answer to the question. This class of help is equivalent to a *data dictionary* and is a central component of not only expert systems, but also of Info Center applications as well. The *help* primitive is defined

```
help  (keyword, help_message_text)
```

Thus, for the above example, the help primitive might be (see Figure 5).

Finally, the last aspect of the basic strategy allows the user to ask the expert system about its reasoning approach. While analyzing the domain, the knowledge engineer naturally questions all the knowledge gained. The KBA must be carefully constructed and tested for accuracy. As the KBA grows in size, it becomes increasingly more difficult for the knowledge engineer to remember all the rules (s)he has constructed. During the testing phase of the development cycle, the expert system often reaches a diagnosis that the knowledge engineer does not understand. At first, the knowledge engineer may presume there is an error in the KBA. On closer examination, the knowledge engineer realizes that what has happened is that (s)he has forgotten the rule that has fired!

What is constructed for the knowledge engineer's *debugging* purposes is a *replay* function. Here, when the expert system makes a diagnosis, the knowledge engineer asks *why* that diagnosis is being made. The expert system then recounts the rule(s) that fired in this instance. Though the replay function was first developed to assist the knowledge engineer, it is a *critical* function for the expert system user. As a user would ask a human expert how a particular diagnosis is reached, so the user would ask the expert system. The net effect is to increase the users' confidence in the expert systems' knowledge and reasoning process.

FIGURE 3

```
prompt    ("svio", "VIO is being used for small temporary datasets")
prompt    ("wvio", "VIO is being used for sort work files")
prompt    ("cvio", "cylinder boundary allocation being used for VIO datasets")
condition ("IOSQ problem", ["resp","svio","wvio","cvio"])
  ...
```

FIGURE 4

```
prompt    ("vio", "VIO being used")
condition ("IOSQ problem", ["resp","vio"])
depend    ("IOSQ problem", ["vio"], "svio")
depend    ("IOSQ problem", ["vio"], "wvio")
depend    ("IOSQ problem", ["vio"], "cvio")
```

Should a new primitive be created for the replay function? If a traditional procedural approach were used by the expert system inference engine, the replay function would most likely be implemented using a *stack*. There, as each symptom is affirmed or denied, it is pushed onto a stack where it is popped for the replay function.

In the given knowledge representation, all the knowledge necessary for the replay is available. Consider the diagnosis primitive:

```
diagnosis    ("IOSQ problem",
  ["resp","pend","iosq"], "iosq treatment")
```

The diagnosis for *IOSQ problem* is given by matching the string *IOSQ problem* in a diagmsg primitive. The treatment is given by looking up *iosq treatment* in treatmsg primitives. Each of the *keywords* in the boolean expression ([''resp'',''pend'',''iosq'']) is looked up in prompt primitives to produce the replay message. Note, if one of the keywords is negated in the boolean expression, the replay is negated accordingly.

```
Diagnosis for IOSQ Problem is being
   made because
      (I/O response time > 40 msec) &
      (Pend time > 0.10 *
       (Connect time + Disconnect time)) &
      (IOSQ time > (Pend time +
       Connect time + Disconnect time))
```

To summarize, the consultation paradigm has several components in its basic strategy. The dialog must consist of a precise set of questions; these questions are driven symptomatically and are conditionally asked. Help must be available so the user can understand what is being asked; and the replay function is necessary for the user to understand the reasoning process and to increase the level of confidence in the expert system. In CPE, the original scheme posed was modified to accommodate the inherent nature of the knowledge as well as the needs of the knowledge engineer and the user.

Knowledge Engineering

To build a knowledge base for CPE, goals must be clearly defined at the outset. For MVS, there are many overlapping areas to be addressed: I/O, page/swap configurations and SRM settings. In addition, you must consider the type(s) of workloads that execute; the higher priority workloads often dictate the underlying hardware configuration so that critical performance can be achieved and maintained. Yet, putting too much emphasis on the higher priority work can cause poor performance for the lower priority workloads. A delicate balance must be found between the workloads while keeping hardware and software costs down.

Literature is filled with papers outlining different CPE studies, each providing some summarized guidance or *rule-of-thumb*. Manuals published by hardware vendors also contain tuning guidelines. Thus, you are literally surrounded with *data;* what separates this from *knowledge* is focusing the data on a set of problems. By defining a set of *goals* to be achieved, you can begin to identify *facts* that can help prove each goal. For each case, enough knowledge must be supplied so each hypothesis can be proved or disproved. Yet, superfluous facts can lead to confusion or wrong conclusions; thus, care must be taken.

Using this goal directed approach, consider what is involved in generating a knowledge base for addressing page/swap configuration problems in MVS. To order the procedure, create a *cause/effect table* where the causes (rows) are the possible configurations and the effects (columns) are the symptoms (see Table 1).

Notice the following characteristics of forming such a cause/effect table. Facts that are missing and facts that confuse the goal can be easily identified. In some cases there is an either/or situation; one set of symptoms or another may be related to a particular configuration. The cause/effect table reveals common symptoms between various configurations; these are used to

FIGURE 5

```
prompt  ("resp", "I/O response time > 40 msec")
help    ("resp", "I/O response time is the total amount of time it takes")
help    ("resp", "to perform an I/O. In MVS/XA, an I/O operation consists")
help    ("resp", "of IOSQ time, PEND time, and SERVICE (connect + disconnect)")
help    ("resp", "time. A common indicator of I/O bottlenecks is a high value here.")
help    ("resp", "Refer to an RMF report for these data items.")
```

TABLE 1

Configuration	Symptoms: S_1	S_2	S_3	S_4	S_5	...
C_1						
C_2						
C_3						
.						
.						

where ...

Configurations	Symptoms
dedicated swap	path util
non-dedicated swap	swap device util
demand locals with swap	local device util
dedicated demand local	# actuators on path
non-ded demand local/data	swap rate
dedicated locals	vio rate
non-ded locals/data	demand rate
locals w swap w data	# local actuators
	swap-in transfer time
	demand-in transfer time
	using > 1 page datasets on one volume
	non-paging vol w/dev util
	non-paging data on local vol
	# swap & local actuators on path

define subgoals in the reasoning process. Finally, the cause/effect table is valuable for testing and documentation purposes.

From this cause/effect table, the next step is to create a decision tree that will form the core of the knowledge base. The strategy is to try to start by asking a question that will eliminate a large percentage of the search space; for example, are swap datasets being used? The questions should go from the general (are non-paging datasets on a local actuator) to the specific (is the average swap-in transfer time > 80 msec). Each node in the decision tree formed represents a question in the expert system and the links represent answers to the questions. Multichoice questions can have many branches, while yes/no questions will have only two (see Figure 6).

The eventual diagram may look more like a network than a tree as there will probably be several paths to the same conclusion or subgoal. The knowledge representation scheme can support this.

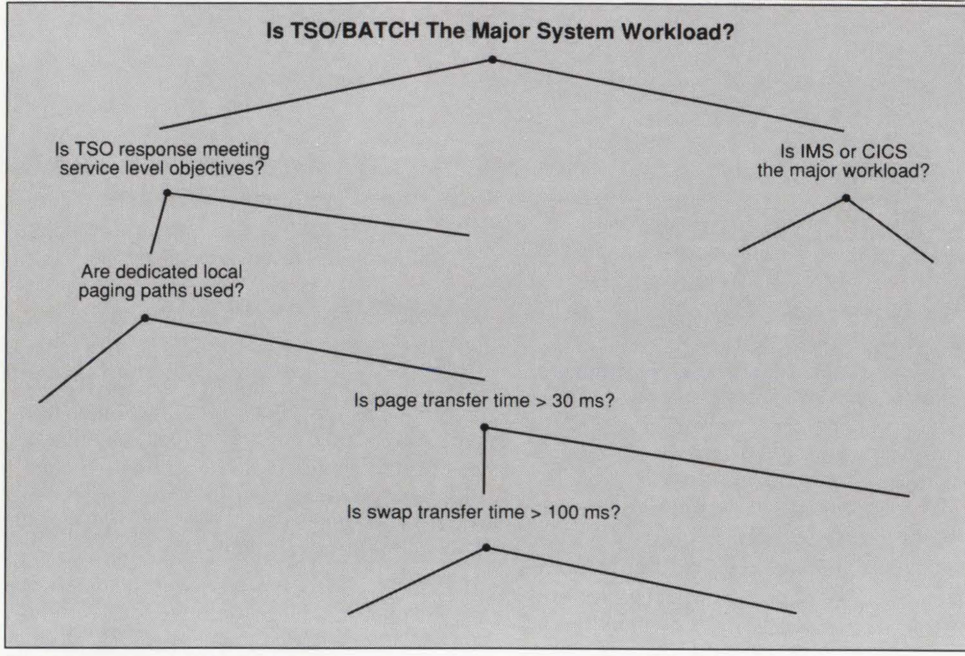
The actual design process is quite cyclic in nature. You can go back and add more dimensions to the cause/effect table; for CPE, a third dimension could be workload (TSO, batch, DB/DC and so on), a fourth could be the physical resources available (for example, expanded storage, solid-state disk and so on). The decision tree is modified as more possible issues are added.

It is possible that even with a multitude of facts, no obvious structure emerges. Domain experts often cannot agree on the approach to apply to a problem. The decision tree is a *heuristic* that the knowledge engineer applies to the system to simplify the consultation process. There is no real guarantee that it will be the best heuristic; in addition, the expert system will still work if no structure is used. Questions can be asked and conclusions can be reached independent of other questions and conclusions in the knowledge base.

An important aspect of knowledge engineering is the notion of *conciseness*. For a knowledge base it means having rules with minimal boolean expressions so as to capture the principle behind the rule. For example, different papers in the published literature might have similar yet conflicting rules:

```
IF    the major workload is TSO/BATCH                &
      Dedicated LOCAL paging path(s)                 &
      - TSO response is meeting objectives           &
      page transfer time > 30 msec                   &
      - swap transfer time > 100 msec
THEN
      There is high page-in response time on a
      dedicated LOCAL path.
```

F I G U R E 6




```
IF    the major workload is TSO/BATCH                &
      Dedicated LOCAL paging path(s)                 &
      - TSO response is meeting objectives           &
      - page transfer time > 30 msec                 &
      swap transfer time > 100 msec
THEN
      There is high page-in response time on a
      dedicated LOCAL path.

IF    the major workload is TSO/BATCH                &
      Dedicated LOCAL paging path(s)                 &
      - TSO response is meeting objectives           &
      page transfer time > 30 msec                   &
      swap transfer time > 100 msec
THEN
      There is high page-in response time on a
      dedicated LOCAL path.
```

On close examination, you see the great similarity between these rules; in fact, if you apply some of the basic laws of switching (boolean) algebras, you can simplify and minimize these rules to:

```
IF    the major workload is TSO/BATCH                &
      Dedicated LOCAL paging path(s)                 &
      - TSO response is meeting objectives           &
      page transfer time > 30 msec
THEN
      There is high page-in response time on a
      dedicated LOCAL path.

IF    the major workload is TSO/BATCH                &
      Dedicated LOCAL paging path(s)                 &
      - TSO response is meeting objectives           &
      swap transfer time > 100 msec
THEN
      There is high page-in response time on a
      dedicated LOCAL path.
```

If you let A = (page transfer time > 30 msec) and B = (swap transfer time > 100 msec), the differences in the first three rules are expressed as: (where + is the notation for OR)

$$(A)(-B) + (-A)(B) + (A)(B)$$

The simplification follows:

$$(A)(B + -B) + (-A)(B)$$
$$(A)(1) + (-A)(B)$$
$$(A) + (-A)(B)$$
$$(A) + (B)$$

Though such simplifications are not always possible, an examination of the KBA while it is being constructed will greatly aid in preserving conciseness. It also helps the knowledge engineer analyze potentially confusing rules-of-thumb.

The cause/effect table(s) created is the best beginning for testing the KBA. As knowledge engineering is a cyclic process, rules will be rewritten implying the cause/effect table gets updated. The *replay* interface described previously is critical when testing the KBA.

To summarize the knowledge engineering process:

1) Define the goals of the KBA
2) Begin acquiring knowledge and form the cause/effect table
3) Define a decision tree
4) Build and test a small prototype
5) Analyze, redesign and test as necessary
6) Expand the KBA adding more rules.

MetaRules

An expert system that can explain its own reasoning process is said to demonstrate *MetaKnowledge* (knowledge about its own knowledge). As an illustration, the *replay* function uses the knowledge contained in the prompt and diagnosis primitives to display MetaKnowledge. In a rule-based system, MetaKnowledge will also include the origin of knowledge (that is, I/O response time can be found on an RMF report).

F I G U R E 7

```
prompt    (keyword, question_to_ask)
diagnosis (problem, [boolean_expression_of_keywords], treatment_key)
```

F I G U R E 8

```
prompt("cputil", "CPU utilization >= 101%")
prompt("pagefaul", "Observed page-fault rate > RCCPTRT")
prompt("uic", "uic < 2")
prompt("outrdy", "AVG OUT-and-READY Queue > # of processors")
prompt("meta000", "System appears to be over-utilized — MPL being reduced")

diag("gen000",["cputil", "pagefaul", "uic"],"meta000")
diag("gen001",["meta000","outrdy"],"tgen001")
```

MetaKnowledge can also be represented by rules on which other rules take precedence. Consider the following rules:

Rule 1

IF cpu utilization > 101% &
page fault rate > RCCPTRT &
UIC < 2

THEN
the SRM MPL adjustment algorithm will reduce the system-wide Multiprogramming Level (MPL)

Rule 2

IF the system-wide MPL is being reduced &
AVG OUT-and-READY Queue > # of processors

THEN
the system is over-initiated. The primary workload cannot be brought into main storage because of a lack of resources.

This pair of rules illustrates a dependency of one rule on another. We have a rule of the form IF A THEN B and another of the form IF B THEN C. The second rule is called a *MetaRule;* the rule being fired depends on whether a different rule has been fired. MetaRules can be used to symbolize *subgoals* within a knowledge base. In Rule 1 of the example, *the system-wide MPL is being reduced* is a subgoal and is displayed as part of a diagnosis. The diagnostic message can also include an indication of where the consultation dialog will proceed:

The SRM MPL adjustment algorithm will be reducing the system-wide MPL.

The dialog will now proceed to ask about your IPS and your physical paging and I/O configuration.

Note that IPS (software) changes will, in general, be easier to carry out than configuration (hardware) changes.

MetaRules help the user understand the reasoning process behind the dialog and help the knowledge engineer order the consultation process. Once a subgoal has been identified, it can be used as part of other MetaRules as well:

IF the system-wide MPL is being reduced &
– TSO is meeting service level objectives &
Number UNILATERAL SWAPS >
25% total PHYSICAL SWAP OUT

THEN
TSO users are being unilaterally swapped out in a system that appears to be over-utilized.

How can MetaRules be *coded* in the knowledge representation? Recall the prompt and diagnosis primitives (see Figure 7). If you are given Rule 1 in the form *IF A THEN B,* the diagnosis primitive is coded as

diagnosis (problem1,[A],B)

Having the expert system assert the *treatment_key* (B) in its working memory allows it to be used in other rules. Thus for Rule 2 in the form *IF B THEN C,* the diagnosis primitive is coded

diagnosis (problem2,[B],C)

To support the replay function, the *prompt* primitive is used to associate text with the *treatment_key* keyword. As a final example, I illustrate how the rules involving the system-wide MPL are coded (see Figure 8).

The concept of MetaRules goes beyond a single knowledge base. For example, you may be working in a domain that concentrates on the use of the SRM and discover that paging problems are present on the mainframe system. Under these conditions, the treatment message in the SRM KBA should direct the user to the paging knowledge base for a detailed analysis of the potential problem. Here, the expert system can create a file containing those facts that the paging knowledge base would require; this would streamline the consultation process when the paging KBA is used. Taking the concept even further, an initialization program could query the user *one* time about his/her specific mainframe configuration: type of processor, type/number of DASD, quantity of main/expanded storage and so on. Other knowledge bases can then be created that presume that these configuration facts exist in a readable format. This use of MetaRules will help to minimize redundancy across knowledge bases.

Summary

Building knowledge bases for any unstructured domain is no simple accomplishment. The objectives were to outline a simple yet rich representation for knowledge in the MVS tuning area. Knowledge engineering techniques vary from interviewing human experts to running simulation models; as CPE rules-of-thumb have been published in many papers, I believe the cause/effect table assists the knowledge engineer in establishing a structure for the knowledge. In addition, the use of the fundamental laws of switching algebras can help to establish concise knowledge bases. Finally, I have defined the concept of MetaKnowledge and have tried to show how the use of MetaRules can improve the quality of knowledge bases. It is my hope that these observations can further the study of knowledge in the CPE problem domain.

ABOUT THE AUTHORS

Bernard Domanski, Ph.D., has more than 17 years experience in the data processing industry with 13 of these years directly in the computer performance management and capacity planning area. He holds a PhD in Computer Science from the City University of New York where he is an associate professor in the Computer Science Department at the College of Staten Island. As principal of Domanski Sciences, he developed the first commercially available Expert System. He has been a member of CMG since 1977 and a member of the Program Committee since 1983. He has published more than 35 papers and has lectured extensively in the United States.

Sid W. Soberman is Senior Vice President of Domanski Sciences where he develops knowledge bases for MVS and CICS systems. He has been involved in commercial data processing for IBM environments since 1972. His areas of expertise include performance management and capacity planning for MVS, CICS, IMS and DB2-based systems and he is experienced in the use of CK-ROM technology. He has been a member of CMG since 1984 and has acted as a referee of technical papers. Domanski Sciences, 16 Colonial Court, Howell, NY 07731, (201) 367-0257.

CS Monitor Eliminates Common Service Area Creep

By John Kador

MVS users who are concerned about Common Service Area (CSA) monitoring could do well to read the current bestseller, *All I Really Need to Know I Learned in Kindergarten,* by Robert Fulgham. In that book, the author reminds us that most of what we really need to know about how to behave we first learned in kindergarten. These rules are fundamental, the author claims, to living harmoniously with each other. They apply equally as well to living with CSA:

- Share everything
- Put it back where you found it
- Clean up your own mess
- Do not take things that are not yours
- Say you are sorry when you hurt somebody.

The alternative is to install a product that can help you enforce some of these rules. Boole & Babbage's Common Storage Monitor enforces most of the rules mentioned previously to optimize CSA utilization for MVS users.

CSA monitoring has been a problem that has plagued MVS users since MVS was introduced. IBM program products, packages from independent software vendors and internal systems *all* use and often misuse CSA and the related System Queue Area (SQA). MVS users have always operated in the blind in this critical area.

Most data processing centers experience some form of "CSA creep," the slow increase of allocated CSA until the system is severely degraded or brought to its knees. CSA creep is often the result of unreleased or misused CSA, resulting in fragmentation, throughput problems and a steady decline in user service levels.

Originally developed as a part of RESOLVE PLUS, Boole & Babbage's MVS real-time problem detection, diagnosis and solution tool, CS Monitor provides an in-depth investigation tool to monitor users of CSA and SQA on MVS/XA. It provides tools to identify and recover "wasted" CSA held by terminated tasks. It can reduce IPLs resulting from CSA creep and keeps vendor software and internal application use of CSA and SQA in check.

Extra Demands

Monitoring CSA and SQA is technically challenging. First, common and private storage compete with each other for allocation of the total, available virtual storage on an MVS system. Unlike private area storage, pageable CSA does not get paged out when an address space is swapped out. Instead, it must wait for a system pageout request. All of this contributes to putting extra demands on the paging subsystem.

Second, allocated (GETMAINED) CSA must be explicitly released (FREEMAINED) when it is no longer required. It is not the responsibility of the operating system. The user must do it. Failure of the user to release common storage that is no longer required leads to shortages of CSA that degrade system performance.

Third, CSA is not user-oriented. It is non-accountable storage. This means that MVS does not keep track of what address spaces have been allocated to which storage locations. As a consequence, shortages of CSA and SQA routinely crop up and have been historically hard to anticipate, prevent or diagnose. Indeed, it is often a mystery as to which system components, user applications, ASIDs and jobs obtain common storage and why. It has been equally difficult to determine when the allocation was made and whether or not the common storage had been released.

MVS/XA was supposed to eliminate CSA creep. However, as many XA sites have discovered, certain systems, application and vendor products will not run above the 16MB line. In addition, several new products (such as DB2) consume huge amounts of ECSA. As a result, common storage both above and below the 16MB line remains a constrained resource in many areas.

Common Storage Shortage

CS Monitor helps to diagnose the causes of common storage shortage by tracking requests for CSA and SQA and maintaining a table of allocations by JOBNAME and ASID. The system time stamps requests and accumulates them by subpool, storage key and location (above or below the line).

CS Monitor can display CSA and SQA allocation in any of several formats. Detailed data, including storage contents, date and time of allocation, length, subpool and storage key can be displayed for the entire system or by specific JOBNAME. Allocations of common storage without corresponding active tasks are flagged by CS Monitor to help identify potentially misused and/or wasted common storage. In this way, it provides a link between the common storage allocation and the address spaces actually doing the allocation.

Using ISPF dialogues, CS Monitor assists users in gathering information and navigating through the system's many options. Full, high-resolution color charts provide displays of common storage allocation.

CSA Hurts Hertz

CSA monitoring has been a frustration for the Hertz Corporation since it introduced MVS at its sprawling Oklahoma City data processing center. "We've always operated in the blind in this area. We couldn't be sure which users were using CSA and who didn't clean up after themselves properly," says Jerry Sevier, manager of MIS.

When Hertz developed a mission-critical on-line Rating System to support its international car rental business, the CSA utilization problem threatened to become unmanageable. CSA utilization often increased to 100 percent during the day. As a result, the system bogged down, frustrating customers at hundreds of rental locations and occasionally forced an IPL outage that brought the entire system down. In the cutthroat world of car rentals when the competition is right next to you on both sides, such outages are completely unacceptable. Hertz operates an IBM 3090-400E under MVS/ESA.

Hertz does not use CS Monitor to actively free CSA. Rather, it employs it offline as a preventive measure usually to alert vendors and users to code that contributes to the problem.

Hertz would like to make the monitoring functions of CS Monitor available to more users. However, the company is also concerned that some features of the system are too powerful for wide distribution. It can take advantage of six levels of password protection to restrict specific uses of the product. Users may be required to have unique passwords to start CS Monitor, to stop it, to access it, to browse common storage allocations, to alter common storage and to free common storage.

"We now have control of the CSA problem. Hertz now operates in a more controlled environment rather than in constant panic mode. Anytime you have a scenario in which you can take preventive action, it's better than fixing things after they're broken," Sevier explains.

CSA Fragmentation At USAA

Any data center with significant processing volumes that does not take control of CSA creep will inevitably become the victim of unscheduled IPLs. This was the case for the United Services Automobile Association (USAA). The San Antonio, TX-based insurance company, processing over five million transactions per day on its four 3090-class computers, found that CSA fragmented so rapidly that it could run the system only for one day before risking a failure. The company was forced to IPL nightly.

"CSA fragmentation problems were appearing every normal production day," recalls Roy Kimberly, an MVS performance specialist and 10-year USAA veteran. "We had to IPL three or four times a week to resolve these problems — without knowing their source."

When USAA management introduced a new service level objective of 24-hour availability, the data center knew it had to stop merely reacting to CSA problems and get ahead of the problem. The time spent bringing the system down, draining it and re-IPLing proved too costly.

A task group studying the problem determined that during the business day the growth of CSA showed steep variations. As the system approached its maximum capacity during the day, paging increased, system response degraded and 300 or more users were frustrated with unacceptable response times. The system limped along until it could be reinitialized later that night. While fine tuning using utilities supplied by IBM gave some relief, the problems grew progressively worse. To diagnose its CSA problems properly, it needed new sources of information.

After installing CS Monitor, USAA quickly learned some things it did not know before. It saw which users and applications received the lion's share of CSA. It identified bugs in some systems, MVS design limitations and application coding errors. Armed with this information, the data center cleaned up bugs, released storage, defragmented the CSA and pointed out design problems to software vendors.

The single biggest finding concerned IBM's paging technique to allocate memory. The system used solid-state devices for its page and swap datasets which had to be added to the system using the MVS PAGEADD command. A total of 16 datasets were being added in this manner after every IPL. Using CS Monitor, they discovered that this process was unnecessarily allocating over 256KB of storage below the 16MB line. Eliminating this problem took a lot of the pressure off. As necessary, USAA uses the FREE service of CS Monitor to delete that storage. USAA contacted IBM which opened APAR #OY13984. IBM has since delivered a solution and the problem is closed.

"No vendor has the time, inclination or energy to go through and analyze the code line-by-line for CSA usage," Kimberly concedes. "So it is understandable that there will be problems." He suggests that vendors run their products through CS Monitor. Then, working backwards, they can determine where the software did not release storage.

CS Monitor is an ISPF dialogue providing detailed information about the use of Common Storage for MVS/XA and ESA systems. It operates in either full-screen ISPF mode or in RESOLVE PLUS command mode.

System requirements for CS Monitor are as follows: MVS/XA or ESA and ISPF Version 2.2.0 or higher. GDDM Version 1, Release 2.0 or higher is recommended but not strictly required. A perpetual license for CS Monitor on a stand-alone basis is $15,000. At press time, the stand-alone version was not available. Until it is, the vendor is offering Resolve Plus at no additional cost. For further information, contact Boole & Babbage, 510 Oakmead Parkway, Sunnyvale, CA 94086, (408) 735-9550, (800) 222-6653.

CMF Analysis

CMF Analysis *from page 79*

not addressed are access method counts and file gets and puts. Global transaction records and accounting and exception records were not covered. Your CICS systems programmers can help determine what variables your particular shop needs to maintain. The Dictionary records show you the format for all variables available in CMF. The field connectors tell you which ones your particular shop is recording. Under CICS 1.7 you have the option of not recording all variables, using an EXCLUDE statement.

Management should encourage some members of the staff to study these complex records. The only way to understand your company's SMF110 data is for someone to spend time with your raw records. Then as management raises its questions, knowledge and a valuable tool are at hand.

Editor's Note: This article was written before Release 2.1 of CICS. In preliminary research, the author has found that the CMF data described under CICS 1.7 appears approximately comparable in 2.1, although further testing will be performed.

It is difficult to process and read complex SMF110 records and a challenge to understand how they all fit together. Dale has created several appendixes to this article which can provide you with initial tools to access and interpret this information. If you would like a copy of these appendixes, send a self-addressed envelope (at least 9" x 12") with $.65 in postage stamps included to "Dale's SMF110 Appendixes," MAINFRAME JOURNAL, *10935 Estate Lane, Suite 375, Dallas, TX 75238.*

ABOUT THE AUTHOR

Dale Doolittle is a performance analyst in Technical Systems for Barnett Computing Company, Barnett Banks of Florida. She has worked in data processing for 20 years with previous experience with NASA and Boeing in the space program and for IBM as a systems engineer. Ms. Doolittle presented this material at CMG '88 in Dallas as well as various regional CMGs. Barnett Computing Co., 4800 Spring Park Rd., Jacksonville, FL 32207, (904) 464-6232.

VSE

VSE *from page 60*

For transients in the LTA or CRT transient areas, the SDL entry is used to load the phase from the library into the shared phase library and then subsequently move the transient into its respective area. These directory entries for these phases are designated by the move operand on the SET SDL statement, for example $$BOCRTA, MOVE.

And finally, a directory entry can be loaded into the SDL using the SET SDL statement *phasename* with no operand, for example $JOBCTLA. Although the phase is not ever loaded to the shared phase library, this will improve performance by bypassing the core-image library directory search at fetch time. This method, however, should only be used with system core-image library phases.

The system getvis area is used widely by the VSE system to dynamically acquire storage for various tables such as the lock table or the DASD label areas. Additionally, this area is used frequently by software vendors for their own execution and working storage. The techniques used to manage system getvis are relatively the same as the techniques used to manage partition getvis except with extra operands on the macro invocations, for example SVA = YES on the GETVIS macro.

Conclusion

From control blocks, to task selection, to storage management, you are building your knowledge base of *VSE under the covers* to the point where you can approach problem situations with confidence. In the next peek, I will begin addressing debugging situations using this base to provide the skills necessary to manage today's complex VSE environment.

ABOUT THE AUTHOR

Eric L. Vaughan is President of Smartech Systems, Inc., a systems software firm specializing in the VSE market. Eric has more than nine years of technical experience with the VSE and VM operating systems and has worked extensively in the field of telecommunications. Smartech Systems, Inc., 10015 W. Technology Blvd., Dallas, TX 75220, (800) 537-6278.

CASE Strategy

CASE Strategy *from page 75*

AGER dictionary have been logically connected, however, it is difficult to change definitions. It has also been difficult to translate logical database definitions into physical schema definitions. Until recently, the procedure was entirely a manual one, involving duplication and transfer of entries from DATAMANAGER to Software AG's data dictionary, PREDICT. To automate much of this process, Administrative Computing is currently beta testing PREDICT CASE, a tool for the planning, development and documentation of an application system.

According to David Wells, WSU's development manager, one of the main attractions of PREDICT CASE is its full integration with the University's complete computing environment. Wells indicates that WSU eventually hopes to integrate PREDICT CASE with NATURAL EXPERT (which will be beta tested at the University later this summer). PREDICT CASE serves as a repository and directory of reusable components and also provides the program composer. NATURAL EXPERT will be used to build a knowledge-based software configuration system.

"By coupling expert systems technology from NATURAL EXPERT with the repository and program composer feature of PREDICT CASE, we expect to produce high quality applications quickly with a minimum amount of custom programming," Wells adds.

The University currently accomplishes process and data decomposition through the use of NATURAL ARCHITECT which the school plans to supplement with PREDICT CASE. The conversion should involve no loss of data from NATURAL ARCHITECT due to an import facility in PREDICT CASE.

Wells indicates that he expects the University to make as solid a commitment to PREDICT CASE as it did to NATURAL and ADABAS seven or eight years ago.

"There's a strong parallel between the two situations, since CASE is as immature today as 4GLs were seven or eight years ago when we first adopted ADABAS and NATURAL. So we're well aware that we'll face a lot of hurdles and challenges along the way; however, we believe the end result will be a better environment."

ABOUT THE AUTHOR

T. Marc Graham is a free-lance writer from Boston, MA specializing in high technology.

ADABAS 5: more than more performance.

ADABAS 5's leading position as a high-performance DBMS has been verified once again by a series of standard, fully scaled benchmarks. Each was conducted on a National Advanced Systems AS/EX™ 100 (equivalent in power to an IBM 3090 500S). And audited by Coopers & Lybrand.

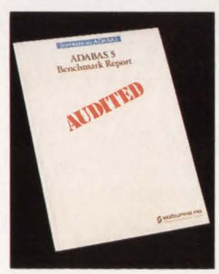
In the **standardized TP1 debit-credit benchmark,** ADABAS 5 worked with a data base containing **1 million accounts.** The results: a **record-breaking 388 transactions per second** (tps) – 99.3% serviced in under one second!

For the first time, an authentic **ET1 debit-credit benchmark** was conducted with ADABAS 5 managing **10 million accounts.** The results: **167 tps** (from terminal in/to terminal out) – 99.5% serviced in under one second!

These figures represent major savings in time and money. But they're not surprising – at least not to the thousands of organizations which already use ADABAS 5.

They expect more performance. Plus the benefits that come from 18 years' experience in DBMS technology: – relational functionality which allows adaptable data structures and fast information retrieval based on multikey criteria – document management and free text retrieval – 24 hour processing in a fully integrated DB/DC environment – portable applications across various hardware and operating systems – distributed processing – entity relationship data bases with recursive retrieval functions for knowledge-based systems.

Discover how much more profitable your DP organization can be. **Conduct your own ADABAS 5 benchmark,** using your own data and application profile in your own production environment. The facts will speak for themselves.



Demand the performance and functionality only ADABAS 5 can offer.

To order your free copy of the ADABAS 5 Benchmark Report, call toll-free: 1-800-843-9534 today.

software AG
Programming Business Success

™ AS/EX is a trademark of National Advanced Systems Corporation.

CIRCLE #75 on Reader Service Card ▲

Q *To what degree can system overhead be affected by using an IPS designed for a 3081 on a 3090-200E with 64MB real and 64MB expanded? I can't seem to convince management of the importance of the IPS!*

A I would need to know what your configuration and software levels were prior to and after the upgrade before discussing whether or not system overhead has indeed been affected. The variables involved are numerous and each affects the system differently.

The function of the IPS is to regulate the workflow through your system. To keep this concern in perspective, ask if your performance objectives are being met. If they are, remember that MVS operates more efficiently under heavier utilization and if the performance begins to suffer you should review your IPS then. Your new processor may have enough extra power not to be driven to high utilization by your existing workload.

If management is not interested in the IPS, you should make them aware that it affects almost every aspect of how your system functions. You may want them to advise you on their priorities and tailor the IPS based on their input.

Answer provided by Leroy Shea, Davis, Thomas & Associates, Minneapolis, MN.

Q *I am having a problem using the VSE macro GETFLD. The GETFLD macro is one of those magical invocations only whispered about in the VSE SUPERVISOR logic manual and we all know apprentice wizards are not privy to understanding them. I would like to create a program that will display the start IOS for each partition, much like the ICCF program IESDA. The way I am using this macro is:*

```
GETSTAT   DS       OH
          GETFLD   FIELD=COMRGPTR,PART=XX
          .
          .
          .
XX        DS       H
```

For every partition I wish to display, I load the corresponding PIK into XX (depending of course on how many partitions are in the system). Once I know the address of the COMREG area, I can get the address of the accounting table at offset +116. The rest is a simple matter of adding all the SIO entries in the Accounting Table. This is a class A FASTSVC service and should be unrestricted but when I try to execute this code I get the following error:

ILLEGAL SVC — HEX LOCATION B29146 — SVC CODE HEX 6B

This error message might as well read:

OOPS — SOMETHING WENT WRONG — YOU GUESS WHAT.

Can you tell me why this doesn't work or what I may be doing wrong?

A You are correct, the invocation that you describe *is* unrestricted and through my reproduction of the description of your problem, I found several facts. I suspect you have a "bug" in your program in setting up the "PART=XX" operand of the macro. In my testing, if I specified a correct PIK in that macro, I achieved the desired results and obtained the partition's communication region pointer. If, however, I specified an invalid PIK with the macro call, I received the same error exit from the supervisor or "Illegal SVC." Perhaps I could suggest the error message is a misnomer of sorts.

The PIK value in the macro should be half-word aligned and in hexadecimal format, e.g. x'0010', x'0020' etc. I would suggest this is your problem.

Use of the macros *is suggested* as it makes coding easier; however, I could suggest a different approach that achieves the same result *without* an SVC:

```
GETSTAT   DS       OH
          LH       R2,XX           Load desired PIK in R2
          A        R2,APIBTAB      Index to PIB2 table from PSA
          USING    PIBADR,R2       Address PIB dsect
          LH       R1,PIBCOMRA     Load part's comreg address
          DROP     R2
```

The address of the Partition Information Block Index Table or PIBTAB can be found within the PSA at address X'248' beginning with VSE/SP 2.1.

Answer provided by Eric Vaughan, President of Smartech Systems, Inc., Dallas, TX.

Q *From a batch partition in MVS/XA, are there any system macros that can determine if an ICF dataset is currently enqueued by any job in the system? Are there any macros that can deallocate/force this dataset from the job it is enqueued to and later reallocate that dataset back to a specific job (not the job running this enqueue test)? Could you show coding examples? The specific use is that CICS journals (systems and user) must be deallocated from the CICS job (after a journal switch) before a batch job can dump and format that journal.*

A IBM has supplied a transaction that is the answer to your problem. The transaction is ADYN and it does allocation and deallocation of datasets. It is documented in the *CICS/OS/VS Customization Guide* (SC33-0239). Also in the same manual in Chapter 6.2 is a sample program that does dynamic allocation. The manual explains how this works fairly well.

By using the DYNALLOC function, a terminal operator can dynamically allocate or deallocate any file that CICS can open and close. With suitable operating disciplines and CEMT commands, extra-partition transient data files, journals, dumps, traces and statistical files can be included.

A VSAM file in the FCT is not opened until the first read is issued. The dynamic allocation program can also allocate and deallocate File Control Table datasets (VSAM files) but you would not normally need this program for datasets. You can use the dynamic allocation and deallocation facility which is part of CICS 1.7. If you have IMS/VS 2.1 you can use its dynamic allocation and deallocation support.

There are also several non-IBM products on the market which will perform batch allocation and deallocation of files, DL/1 databases and CICS queues.

Answer provided by Dennis Bertrand, Davis, Thomas & Associates, Minneapolis, MN.

Q *We have recently upgraded our MVS/SP 2.1.7 system to MVS/SP 2.2.0. Prior to the upgrade our paging environment was relatively stable averaging 10 to 12 pages per second. This upgrade included DPP 2.3. Any ideas?*

A We tried to contact the person who asked this question but found he was no longer with the company. Based on the limited amount of information we have, we thought that the company could be using the default IPS with MVS 2.2.0. It is also possible that the MVS/SP 2.1.7 system had a tailored IPS for the company's needs. Of course, there are other possibilities but anything at this point is just speculation.

Answer provided by Michael O'Donnell, Davis, Thomas & Associates, Minneapolis, MN.

VM *from page 19*

other change, one that also improves the ease of editing. Find the line in PROFNOTE XEDIT which reads:

COMMAND SET PREFIX OFF

and replace it with:

COMMAND SET PREFIX ON

The PREFIX command turns the leftmost five bytes on the screen into a command area. Now you have available all of XEDIT's line commands for adding (A), copying (C or CC), moving (M or MM) and deleting (D or DD) text. The reformatting capabilities still fall below the norm for professional word processing packages; however, the prefix commands and the PF11 split/join command let you manipulate words and phrases without retyping. Ragged margins matter little compared to the clarity of the message.

In addition to configuring the software, your company needs to train the users. Beginners in electronic mail often make the mistake of writing the same memo on the screen as they would on paper. Paper memos tend to fill one or more pages and frequently cover several aspects of a topic or even several topics. Their relatively slow transit time encourages including as much as possible.

Long memos in electronic mail are difficult to read. One screen holds less than half of what is on a page, so that readers must change screens on a long memo two or three times as often as they flip pages in the paper form. Also, a memo with multiple topics makes them go through it several times while answering each part. The problem is that one topic may have an immediate answer while another requires research. With paper memos, they may well wait until a complete answer is ready. For electronic mail, the wait is self-defeating.

Project teams should follow three basic rules when using electronic mail:

Rule 1: A memo that fills more than two screens should be more than one memo.

Rule 2: A memo that has more than one topic should be more than one memo.

Rule 3: Electronic mail deserves rapid response.

The idea of the first two rules is to make the third possible. If team members answer immediately even just half the questions they get, the lag should be only a couple of hours on the average, instead of a couple of days via departmental mail.

Timeliness is important. Unanswered questions can lead to false assumptions and a slow response may arrive too late to alter a line of thinking. A person who has spent two days working on an idea may not abandon it willingly.

Notebooks

A project team that uses VM's electronic mail will have a machine readable record of the discussion. The NOTE command automatically saves every memo into a NOTEBOOK file, that, with proper editing, can become part of the final, formal requirements definition.

Keeping the project related NOTEBOOKs clean of extraneous memos is important. Each NICKNAME entry has only one NOTEBOOK associated with it. If project team member Joan writes to project team member Michael, every memo will go automatically to the PROJECT NOTEBOOK she assigned to his NICKNAME. This includes any collegial gossip or invitations to lunch. She can segregate such private memos by specifying a keyword qualifier. Instead of a simple:

```
NOTE MICHAEL
```

she could enter:

```
NOTE MICHAEL (NOTEBOOK LUNCHES
```

If Joan and Michael serve jointly on two project teams, she can either make up two NICKNAMEs for him in the address book or use a shell EXEC called MICHAEL:

```
/* REXX EXEC to send notes to Michael */
ARG PROJECT /* Specify which project notebook.*/
/* Establish "FINREC" as the default. */
IF PROJECT = '' THEN PROJECT = FINREC
EXEC NOTE MICHAEL '(' NOTEBOOK PROJECT
EXIT
```

Then if she wants to write to him about the financial records project (FINREC), she need only enter:

MICHAEL

If she is responding to a question about the PAYROLL project, she would add:

MICHAEL PAYROLL

Likewise if she is asking him to lunch (on business, of course), she can also save keystrokes:

MICHAEL LUNCHES

The overhead of such a shell EXEC is minimal compared to the convenience. IBM uses the technique frequently in its own code.

Joan will want to save notes that come to her reader queue. Again VM makes this simple. When she reads her mail, she can use the PF9 key in PEEK or from the RDRLIST display to RECEIVE the message. It will go into whichever NOTEBOOK she specified for the sender in her address book. Again this may be a problem if she and Michael are working on two projects at once. Another shell EXEC could be useful:

```
/* REXX EXEC for receiving notes into the
     PAYROLL NOTEBOOK
The filename of this EXEC should be PAYROLL  */
ARG FILENUM
RECEIVE FILENUM '(' NOTEBOOK PAYROLL
EXIT
```

To save a note correctly now, she need only type PAYROLL over the desired entry on the RDRLIST display.

Joan can examine her project notebook while writing a reply. VM provides two ways to do this. First, she could go to the command line of her current XEDIT session and enter:

```
X FINREC NOTEBOOK
```

This will replace her note with a full screen display of the financial records project notebook. When she has seen everything she wants, she could QUIT using PF3 and VM would return to what she had been writing. She can also keep the FINREC NOTEBOOK open by entering:

```
SCREEN 2
```

to split the screen. If at some point she wants a single screen again, SCREEN 1 will change it back. She can also define any of her PF keys to split and un-split screens:

```
SET PF19 SCREEN 2
SET PF20 SCREEN 1
```

These settings are useful enough that you should consider including them in an updated PROFNOTE XEDIT on the Y disk.

These techniques for using NOTEBOOKs let project team members keep a paperless *paper trail* of all questions and decisions, filed chronologically and backed up on a regular schedule. In a future article, learn how to turn these NOTEBOOKs into project documentation. Good record keeping cannot substitute for a well researched requirements study; however, it can help the team avoid reworking issues already resolved.

This article is an excerpt from Chapter Four of VM Applications Handbook *(McGraw-Hill, 1989) Gary McClain, editor.*

ABOUT THE AUTHOR

Michael Seadle, Ph.D., is head of user services at Eastern Michigan University, Ypsilanti, MI. He has worked as an applications developer, data-base administrator and VM systems programmer. Also, he helped establish and is an active member of the VM Enthusiasts of Michigan, a VM users' group.

VIEWPOINT

Reengineering: Buzzword Or Business Advantage?

By Kent Petzold


Kent Petzold, President and CEO, VIASOFT, Inc.

Computer Aided Software Engineering (CASE) was originally thought of as the front-end software tools that facilitate the development of new systems to meet new market requirements. By this definition, CASE development typically consumes less than 20 percent of MIS programming resources. I would like to propose a wider, more representative definition of the term, adding all the programming activity that goes into enhancing and maintaining existing systems. This activity, called CASE for Existing Systems or *reengineering*, is a top priority for corporate America. According to the Gartner Group, reengineering consumes up to 80 percent of MIS resources and costs corporations $30 billion annually.

History Of Reengineering

One of the most frequently discussed results of the technological revolution in American business has been the realization that information is a corporate asset. Two decades' worth of information gathering and software development have created a huge portfolio of applications that are running businesses and providing unique assets that deliver a competitive advantage. It should come as no surprise, then, that maintaining and enhancing these assets to respond to changing market conditions is a top priority. The surprise is that it is a time-consuming and error-prone process that has remained largely untouched by the technological advancements that created it.

The problem of enhancing existing systems has many roots. Most of the world's mainframe applications are written in COBOL, many of them large and complex systems. It is the standard mainframe programming language with more than 80 billion lines of code in production. However, COBOL was designed for development flexibility and power, not for easy maintenance and enhancement. These billions of lines of production code have gone through decades of change at the hands of many different authors and much of it is older than the programmers working on it. Finally, the few products that have been brought to market are severely limited, so that this process remains primarily manual and resource-intensive.

Studies such as those undertaken by the Gartner Group reveal that reengineering will grow to consume 90 percent of MIS resources by 1990. IBM conducted a detailed study of the reengineering cycle and found that almost half of a programmer's time is spent not in the action of design, code and test but in the cognitive challenge of trying to understand how the program works. This means that 50 percent of the reengineering cycle is bogged down in analysis. These studies define the problem as a crisis in corporate America and have propelled companies to begin developing reengineering solutions.

Phases In The Reengineering Cycle

The steps in the reengineering process or cycle are well-defined:

1. Identify a need for enhancement, repair or code improvement
2. Review the system and develop a plan (specification and design)
3. Code and compile
4. Test and debug
5. Documentation and production turnover.

In reality, the process rarely proceeds so simply. As a programmer learns more about the program throughout the cycle, midstream adjustments are made. Findings in a test session often require coding changes, a design update or even a specification change, then more testing. It generally takes an average of seven to 12 attempts just to get the program through the compile step. The cycle becomes an iterative process resulting in delays and growing backlogs.

Current Solutions To The Reengineering Crisis

Despite the significance of reengineering and its impact on corporate MIS resources, system software firms have been slow to offer effective solutions. Instead they have been focusing on the easier option of marketing software for new applications development. Most of the techniques that are being described as reengineering solutions are characterized by misplaced focus; they automate the steps that take the least amount of time and they work around or eliminate existing code rather than leverage it. They fall into three categories: rewriting, restructuring and maintenance.

- Code generators assume that the best approach to reengineering is to always rewrite the code. However, development of new code is impractical and difficult to cost-justify if the present system is still functional; IBM estimates that only seven percent of existing applications need to be rewritten.
- Restructuring products rearrange the structure of a program, creating code which is easier to work with. However, restructured code does not reveal the program logic and data relationships, so the critical, resource-intensive process of analysis remains manual.
- Maintenance utilities assist in repairing old problems. However, they add nothing to the process of improving or enhancing code.

Features Of True Reengineering Products

There has now emerged a new category of reengineering products which are truly effective solutions. These products can:

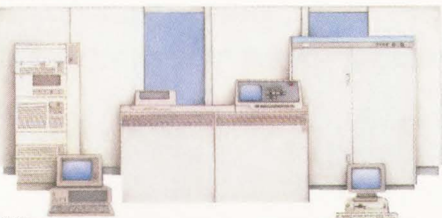
- Read and understand COBOL programs
- Capture comprehensive logic and data path information
- Create logical modules by identifying and isolating related business functions
- Build an on-line repository of information about the application that is available at all times
- Make the information flexible enough to be used in each of the reengineering tasks from redesign to bug-fixing
- Integrate program knowledge with tools that support all the phases of the reengineering cycle from design and analysis to coding, testing and debugging
- Provide an integrated technology platform and tool set, automating the transition and flow of information between reengineering tasks
- Provide a platform that supports evolution toward reverse engineering.

This new class of products is offered as an integrated suite by vendors, thereby offering a comprehensive option to buyers of CASE tools for existing systems. These full life-cycle tools bring automation and intelligence to the challenge of reengineering and will go a long way to addressing one of the biggest concerns of MIS today.

Our laser printers give new meaning to "sharing"

Until now, sharing has been pretty one-sided: several workstations had to share one laser printer. But Printer Systems has redefined the idea of sharing. Now a laser printer can share across several dimensions.

Different platforms

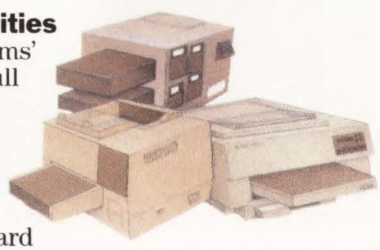


Printer Systems laser printers can be shared by several different kinds of computers. Like IBM. And DEC. And, of course, PCs. All at the same time. Coax, twinax, parallel, and serial connector ports are provided on all our laser printers.

Different applications

And you can get some pretty fancy things out of these different computers, without a lot of programming. Like barcodes. And FlexFonts from BitStream, with access to over 2,000 fonts. So now your mainframe can do what you thought you needed a PC for.

Different personalities



With Printer Systems' new laser printers, full emulation of the IBM 3812, the DEC LN03 Plus, and the HP LaserJet II is a given. In fact, our printers not only come standard with all the same fonts as IBM's, DEC's, and HP's laser printers, but our font matrices are precisely identical to theirs. So ours print exactly the same as theirs.

Real compatibility

Complete compatibility is a given, too, right down to the resolution. Like the IBM 3812, you can have 240 dpi for IBM applications. Or 300 dpi like the DEC LN03 Plus and the HP LaserJet II. In the same machine. Switched–not scaled–by the machine itself depending upon the application.

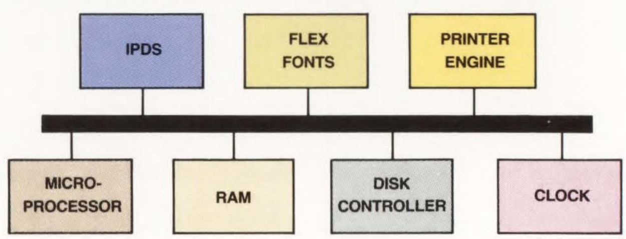
Introducing the Intelliprint 218®

Printer Systems' new Intelliprint 218 permits all this sharing without giving up performance. Its 32-bit RISC technology means that its real throughtput is the rated speed of 18 pages per minute. And its peak duty cycle of 50,000 pages per month means it can handle however much you demand of it.

And the Intelliprint 106®

Sometimes, you can have too much of a good thing. So Printer Systems developed the Intelliprint 106. It, too, is based on 32-bit RISC technology, so its 6 page per minute rated speed is its real throughput. And its duty cycle of 3,000 pages per month is ideal for lower-demand situations.

Open architecture on a PC bus



Printer Systems has also redefined "sharing" by putting a PC AT inside the Intelliprint 218. That provides all the functionality of a dedicated workstation. And, like any other workstation, it affords the flexibility to add options, like a LAN interface, for instance.

Wait, there's more

In fact, there's a lot more. Printer Systems has more up its sleeve, like spooling. And real-time, automatic switching between interfaces. And different graphics emulations. But we can't tell you about all of it yet. At least not here. Why not give us a call at 1-800-638-4041? We'd be delighted to fill you in. So you can start sharing the way sharing was meant to be.

Compatible in every way

Printer Systems Corporation, Corporate Headquarters: 9055 Comprint Court, Gaithersburg, Maryland 20877
(301) 258-5060, (800) 638-4041, Fax: (301) 926-7333, Telex II: 710 828 9621

CIRCLE #50 on Reader Service Card ▲