VM AND THE VM COMMUNITY: Past, Present, and Future

Melinda Varian

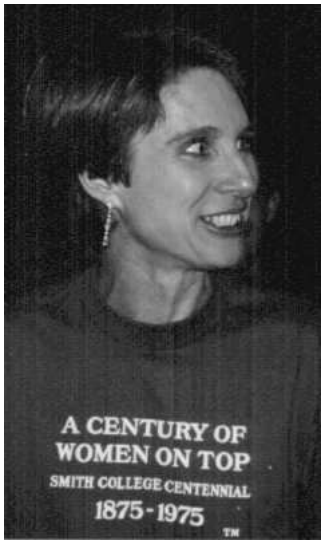
Office of Computing and Information Technology
Princeton University
87 Prospect Avenue
Princeton, NJ 08544 USA
—.—
BITNET: MAINT@PUCC
Internet: maint@pucc.princeton.edu
Telephone: (609) 258-6016

April, 1991

I. INTRODUCTION

I will be talking today about the past, present, and future of VM, with an emphasis on the influence of the VM community on the growth of the VM product.

This paper was originally presented at Australasian SHARE/GUIDE in Melbourne in 1989. My husband Lee and I had a delightful time at ASG and are most grateful to ASG for being our host in Australia and to SHARE for giving us the opportunity to represent it there.



Cecilia Cowles

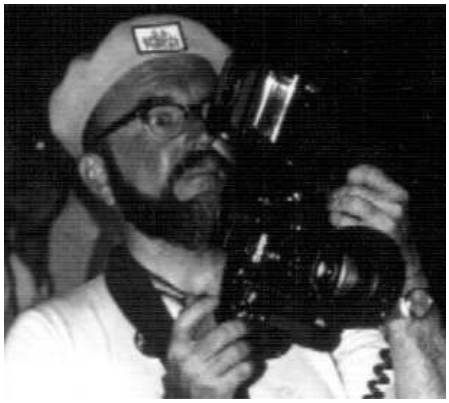
When I spoke at ASG, I began by conveying greetings from the President of SHARE, Cecilia Cowles. I will do that again today, because the pictures are too good not to use again.

In the past, when I've spoken at SHARE and SEAS, my talks have been strictly technical. This talk was the first time I'd been asked to give my opinions, so you may find that you get more opinion than you wanted. Certainly, I should make sure you understand that my views are not necessarily those of my management (and are sometimes not those of SHARE management either).

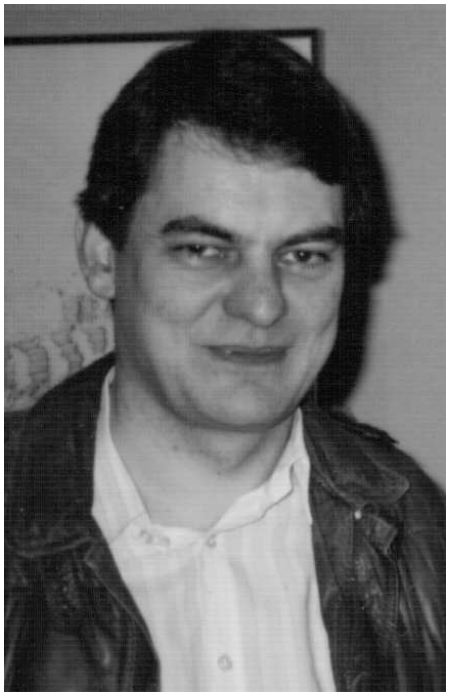
I must also ask you in advance to forgive me my ethnocentricity. Though I speak of "*the* VM community", I realize that there are actually several overlapping communities of VM people, located in different parts of the world, both inside and outside of IBM. For the most part, I will be speaking of the community of which I'm a long-time member, whose center is the VMSHARE

electronic conference. This community overlaps heavily with SHARE and SEAS, with the annual VM Workshops in North America, and with various regional VM user groups. It includes many participants from IBM as well.

I'll be showing you pictures of some members of this community, but because there's not nearly enough time to show all the people who have made outstanding contributions to VM and to the VM community, my choice of who to show was semi-random, depending a lot on which pictures I was able to get. I owe thanks to many photographers who lent me their pictures, but especially to Joe Morris of SHARE and Stuart McRae of SEAS.[1] I am also indebted to Sandra Hassenplug and John Hartmann for their assistance in preparing slides, as well as to several of my colleagues at Princeton.



Joe Morris



Stuart McRae

Sandra Hassenplug

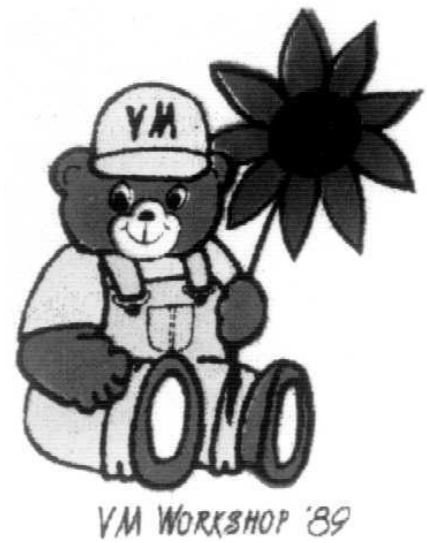
[1] I am grateful to the many people who succumbed good-naturedly when I badgered them for photographs. I wish particularly to thank Bob Creasy, Adenah DeAngelis, Jerry DePass, Walt Doherty, Lyn Hadley, Ed Hendricks, Peter and Carol Jobusch, Ted Johnston, Ken Holt, John Shaw, Dave Tuttle, Lee Varian, John Wagner, Lynn Wheeler, Rich Wiggins, and Joan Winters.

John Hartmann

Princeton Interactive Computer Graphics Laboratory

I should probably also explain the iconography I'll be using. For many years, the SHARE VM Group lamented the fact that VM had no symbol, no totem. A couple of attempts were made to select one, but they fell flat, because, of course, such things can't be mandated. Meanwhile, the MVS Group had the turkey (which they chose of their own volition), and they went around wearing turkey hats and putting turkey stickers on elevator doors, and so on. The legend is that the MVS Performance Project began using the turkey as a symbol in the early days when MVS performance was definitely a turkey, and the symbol soon got extrapolated to the whole MVS Group.

With VM's amazing growth, the VM Group in SHARE has always had a problem making newcomers feel at home, simply because they always outnumber the oldtimers. In 1983, the Group was going through yet another attempt to overcome this problem, and it was decided that at SHARE 60 we would hand out little square yellow stickers to newcomers to the VM Group and little square blue stickers to oldtimers, with the idea that if they all put the stickers on their badges, the oldtimers could identify the newcomers and help make them feel at home. The problem with that, of course, was that nobody could remember which sticker was which, so it didn't work out at all. A couple of days into that week, however, Carol Jobusch bought a few hundred teddy bear stickers, with the idea of affixing them to the cuddlier of the oldtimers so that the newcomers would know that here was a warm cuddly person who ran *the* warm cuddly system and who could be counted on to be friendly if approached. Within hours, the teddy bear had become the *de facto* symbol for VM, and everybody in the VM Group, old or new, cuddly or prickly, was wearing a teddy bear on his badge. (The Jobusches subsequently got a 50-KB roll of stickers, to keep SHARE well supplied.)



Carol Jobusch

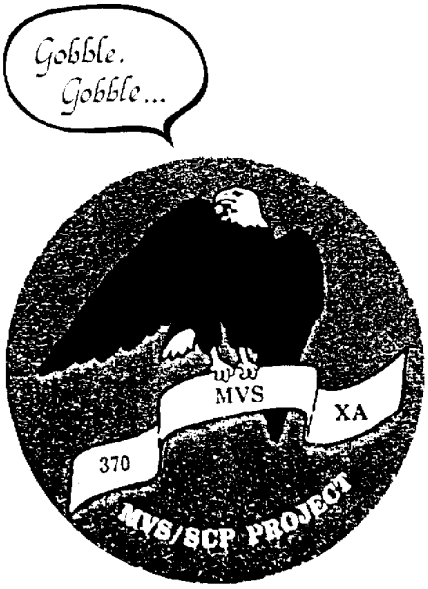
One rather strange result of all this has been that the offices of many hard-bitten system programmers are now full of teddy bears.

Typical VM system programmer's office

However, even without being reminded of it by the MVS Group, we would have been careful not to let our arctophilia degenerate into icky sweetness.

Not surprisingly, soon after VM adopted the teddy bear, the MVS Group decided that the turkey was no longer an appropriate symbol for MVS, and opted instead to use the eagle.

But, of course, such things can't be mandated.

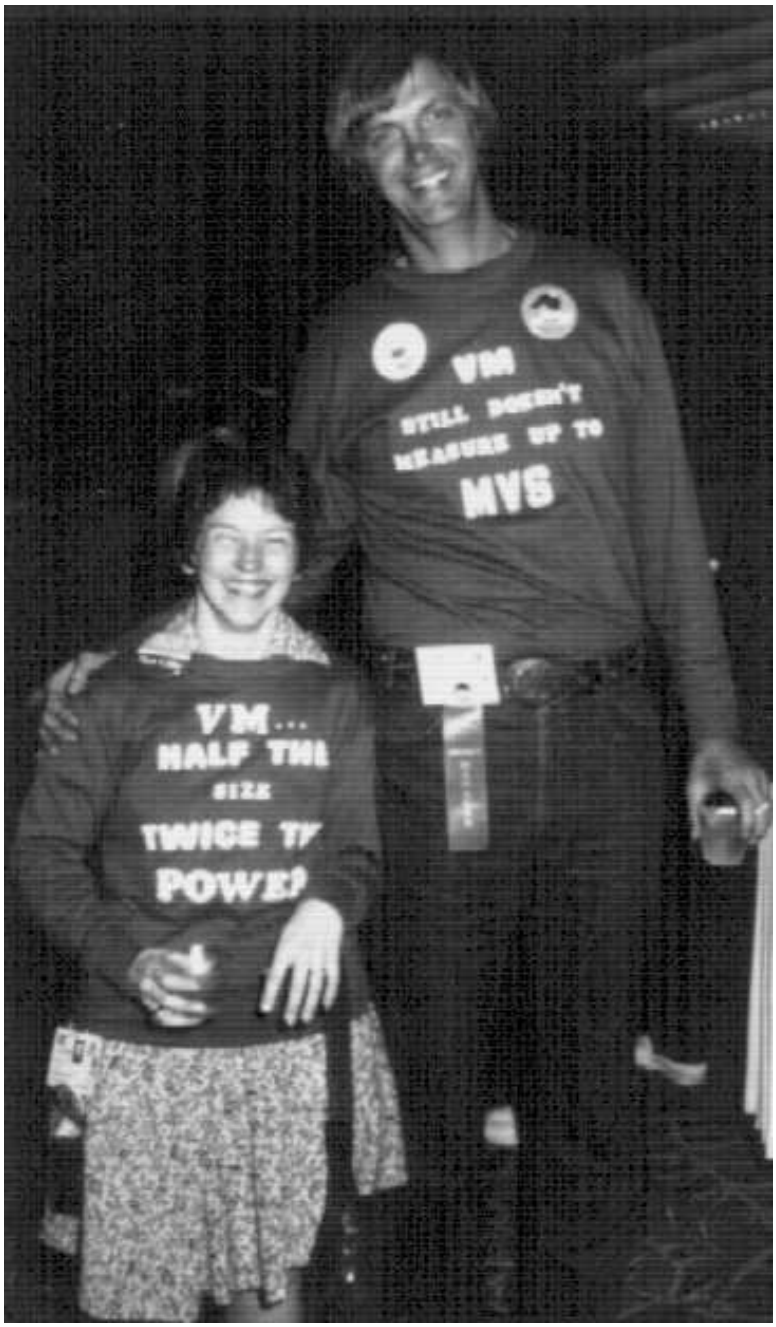


I should also warn you that you may notice in my presentation a few slides that indicate a certain rivalry between the VM and MVS Groups.

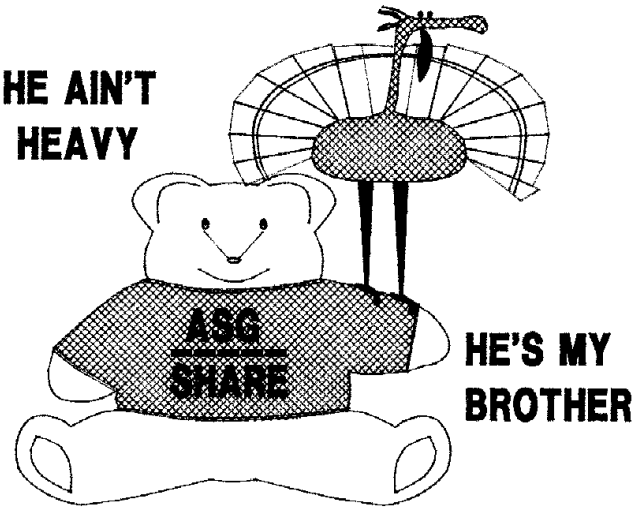
"Is this any way to treat a guest?"

"The light at the end of the tunnel"

I hope that none of you will take offense at our banter, for I assure you that the rivalry is a good-natured one and only skin deep.



Donna Walker: "VM—half the size, twice the power"
Jimmy Pittman: "VM still doesn't measure up to MVS"



"He ain't heavy—he's my brother"

In fact, most installations in the SHARE VM Group run MVS, and most Group members use MVS every day, although, of course, very few of us use TSO.

Running TSO

is like kicking

a dead whale

along the beach.

Horace: "Ars Poetica"

II. DIGRESSION: ON WHERE REALLY GOOD SOFTWARE COMES FROM

Before getting into the history of VM, I'd like to discuss briefly some observations I've made over the years on where really good software comes from. I want to talk about this topic a bit because I think it may serve as a theme for what follows.

First of all, let me define my terms. To me, "really good software" is simply software that greatly enhances the ability of people to use computers, software that lets people use computers to do easily things they couldn't do at all before. Perhaps the best test of whether a program or a system is really good is whether people fall in love with it, whether their eyes light up when they talk about it.

The reason that it's important to understand where such software comes from is that it is the source of the real growth of our systems. Really good software enhances the productivity of our end users, makes our systems grow, and expands IBM's markets. So, everyone of us in this room benefits when IBM or its customers find ways to create more really good software.

In case it's not already clear, let me add that I'm not talking about software with low APAR rates. Really good software often has extremely high APAR rates, especially when it is new. Indeed, I'm almost, but not quite, willing to assert that an extremely high APAR rate in a new product should be cause to give the author an award. Certainly, a very high APAR rate in a new product is often an indication of high product acceptance. If I may give a couple of examples:

- First, let's consider the early days of Xedit,[2] although I always hate to use editors as examples because they're so controversial. I can't remember now who it was who said, "Editors are like religions, except that people don't have such strong feelings about their religions."

  There can be no question that by releasing Xedit in 1980, IBM gave CMS a new lease on life. Nevertheless, when Xedit first came out, its APAR rate was so high that an IBMer whom I greatly respect asked me whether releasing it had been a mistake. In fact, the APAR rate was quite high, but the problems that were being reported were almost all very small problems. It became clear that people were using Xedit so enthusiastically and so creatively that they were stretching it to its limits, and in the process running into little errors that hadn't been discovered in IBM's testing. While the errors that were being reported were genuine and needed to be fixed, the remarkable thing about many of them was that people cared enough

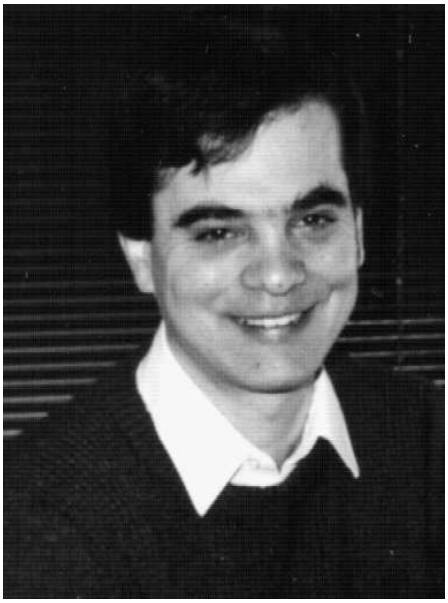
---

[2] When asked what other editors influenced the design of Xedit, Xavier de Lamberterie graciously replied with the following note:

> Well, Xedit comes from a long way. It has been influenced by the editor from CP-67, then some other editors that were developed locally at the Grenoble University (including editors with macro capabilities, which were probably the first ones to have such a concept), and certainly from Ned that we had a long time ago (some Xedit target commands are inspired from Ned). Then later on, when full-screen displays were available, Xedit took some ideas from Edgar and ISPF (features like prefix line commands).
>
> But I guess the major feature of Xedit was to keep the "heart" relatively small and allow users to redefine and/or extend the existing commands *via* EXECs or REXX macros. This was one of the major successes of Xedit.

about the product to invest the time needed to get such a minor problem fixed. That they were willing to do that was a tribute to Xedit's author, Xavier de Lamberterie. (If you've ever wondered where the "X" in "Xedit" came from, now you know—it was Xavier here.)

Xavier de Lamberterie

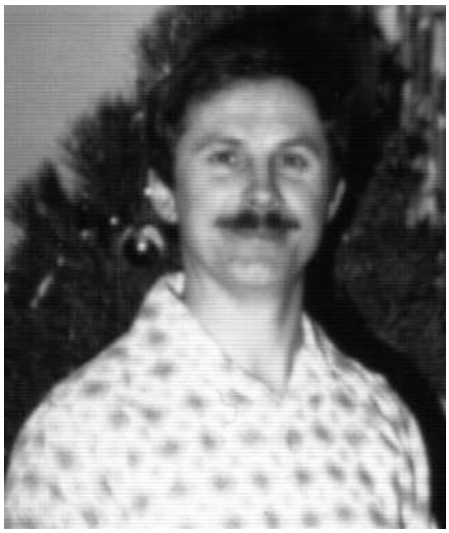


Noah Mendelsohn

- Second, let's look at the early days of Pass-thru. The original author of the system that became Pass-thru was Noah Mendelsohn. In 1974, Noah was given the task of inventing a way to allow the PSRs and Change Team people using a system in Mohansic to get access to the RETAIN system in Raleigh. His solution was "V6", a server that provided the basic Pass-thru function in an elegant and extensible form. Bill Anzick, who had advised Noah on V6 from its beginning, took over the project in 1977 and expanded it into the product that was announced in 1980 as Pass-thru.

  Although I don't believe that SHARE had asked for something like Pass-thru, as soon as we saw it, we wanted it badly. As soon as the tapes arrived, they were rushed to our computer rooms, and the product was installed right away, all over the world.

  Pass-thru had been used extensively inside IBM before it was released, so it had already been fairly well debugged. Many of us who had moderate-sized Pass-thru networks never saw a failure. However, within weeks of its release, Pass-thru started being used on quite large networks, with many more concurrent users than it had ever had before. Inevitably, it was found that some algorithms that worked perfectly well in smaller networks didn't work so well in large networks. That was not surprising. What was surprising was the way the problem was handled. Anzick and Don Ariola, of Field Engineering, spent a year heroically expanding and strengthening Pass-thru to support the very large networks that customers were building. Their changes came out as APARs, so again we saw a new product with a very high APAR rate, and again that APAR rate meant that the customers were extremely happy with the product.



Bill Anzick

Don Ariola

Of course, I am not disappointed to find a piece of really good software that is also bug-free. The example that comes to mind is REXX, which was essentially flawless by the time it was released. This means that the author of REXX, Mike Cowlishaw, had managed to write a large piece of code that was coherent enough that it could actually be debugged, but it is also the result of his code having been exercised very thoroughly inside IBM before it was released.

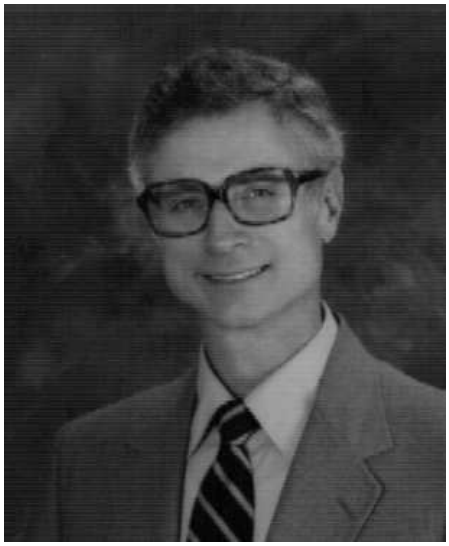
Mike Cowlishaw

These three examples and almost all other examples I know of really good software have common characteristics that I've come to believe are necessary in producing software that allows users to do wonderful new things with computers:

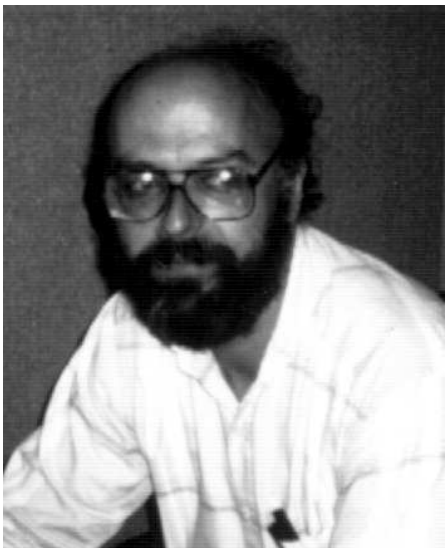
1. **Really good software can be written only by very small groups of very skilled programmers.**

   A group size of one is not too small; twelve is probably too large. The group must be small enough that communication is not a problem and small enough that everyone involved feels responsible for the end result. The programmers themselves must be highly skilled, though perhaps we dissolve into a tautology if we say that really good programmers are the ones who write really good programs. Nevertheless, average programmers never write really good software. On a large project, using very good programmers is the only way to keep the size of the group small enough to maintain communications. But even on a small project, software produced by very good programmers is qualitatively different from that produced by the less talented, and no amount of procedure or process or standards can compensate for this difference.

   VM itself is the perfect example of what can be achieved by a very small number of very good programmers. During CMS's formative years, 1965-1971, there were never more than half a dozen people working on it at any one time, including designing, developing, and documenting it. There were no more than eight people at a time working to develop the System/370 version of CMS. The original versions of the Control Program, CP-40 and CP-67, were also the work of very small groups. Most of the work on Release 1 of the VM/370 Control Program was done by just twelve people.



Chuck Tesler



Serge Goldstein

I've already mentioned Xedit, Pass-thru, and REXX, which had one or two primary authors each. The same was true for Smart, which was written by Dick Jensen, and VMAP, which

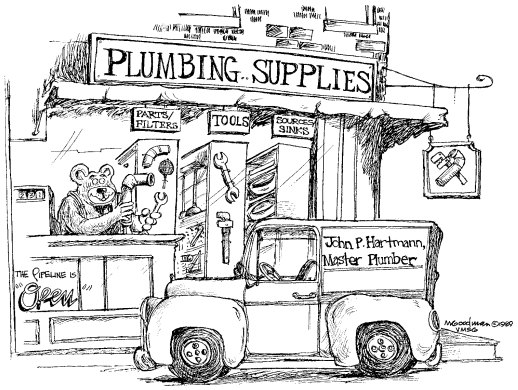
was written by Chuck Tesler.[3] Another good example is Track, the splendid system programmer tool written by my colleague at Princeton, Serge Goldstein.

2. Really good software is almost always a labor of love.

By this I mean that the programmer must really want to create this particular piece of software. A case in point is *CMS Pipelines*. John Hartmann, of IBM Denmark, the author of *CMS Pipelines*, has described its origin:

> I passed through Peter Capek's office one day. We can't really remember when it was—probably sometime late '80 or early '81. He had a box of the *Bell Systems Technical Journal* issue on UNIX[4] under his table. I saw him slip a copy to someone, so I said gimme! Having read it (and ignoring their remarks about structured data), I ran off shouting from the rooftops and then began coding with both hands and my bare feet.[5]

John Hartmann



"The Pipeline *is open"* [6]

It's not too far-fetched to say that the best programs are the ones written when the programmer is supposed to be working on something else. REXX is an obvious example of that.[7] Since

---

[3] Then with IBM, now with ProSoft.

[4] UNIX is a trademark of UNIX System Laboratories, Inc.

[5] J.P. Hartmann, private communication, 1990.

[6] This drawing by Michael Goodman is copyrighted (1989) by the VM Systems Group and is used with permission.

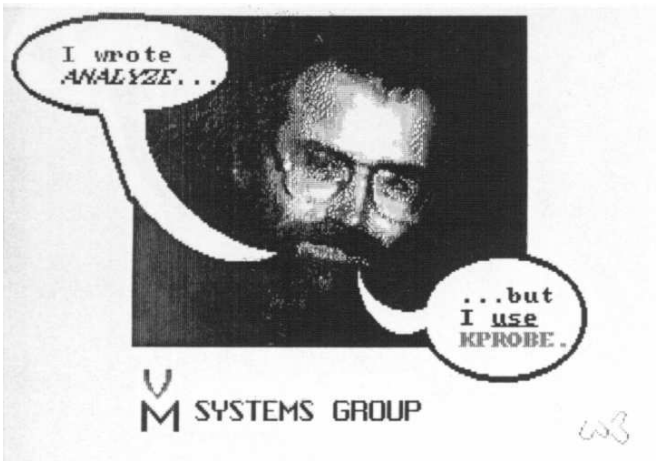
[7] "It was my good fortune to be in charge of the IBM Hursley Laboratory during the period when Mike Cowlishaw, who at the time was a member of IBM's System Assurance Laboratory, created REXX. I should hasten to say that I claim no credit whatsoever; REXX is the product of a dedicated individual committed to the solution of a problem. He did so not

Serge isn't here, I can say that Track is another good example of that, but it's true of almost every really good program. Very good things happen when management is enlightened enough to appreciate the importance of allowing programmers some free time for projects of this sort.

The programmer must also be allowed a sense of ownership, of personal responsibility. This is not "egoless programming". Exciting software, the kind that expands the uses of computers, is not produced by people who feel themselves to be small cogs in vast machines.

3. **Really good software is almost always begun because the author himself needs it; it makes his computer do something that he really wants it to do.**

CP-40 was written because the folks in Cambridge needed a way to share their one small computer so that they could do all the things they wanted to do. The first communications component of VM was written so that people at the Cambridge Scientific Center could communicate with people at the Los Angeles Scientific Center. Smart and Track and *CMS Pipelines* were all written to assist their authors in their work. Dick Jensen, who had been a CP developer in the Burlington days, was a fire-fighting SE at the time he wrote Smart, just as Serge is a fire-fighting system programmer today.



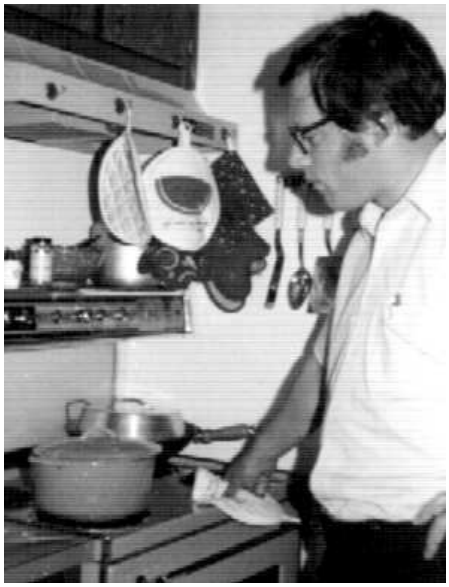
Wilt Byrum (self-portrait)

Another good example here is Amdahl's Analyze dump viewer. Analyze began as a massive set of local modifications to IPCS Dumpscan. The author was Wilt Byrum. Wilt was a system programmer being deluged by dumps, so he needed a better and faster dump reader badly enough to write it for himself. After Wilt left Amdahl, Analyze was adopted by John

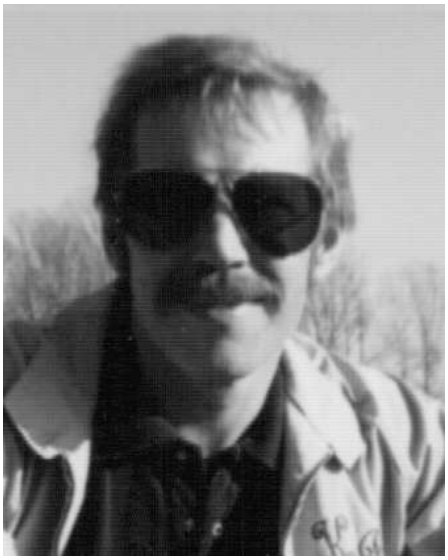
---

because he was asked to; not because he was expected to; not even because there was any job-related requirement for him to do so, because he was supposed to be wholly dedicated to the evaluation of products developed by others; but he did so because Mike Cowlishaw saw the need and the solution, and got on with it, almost *despite* his management rather than *because* of them." (Sir John Fairclough, in *The REXX Handbook*, Gabriel Goldberg and Philip H. Smith III, eds., McGraw-Hill, 1992, p. *xi*.)

Alvord,[8] who massively extended and restructured Wilt's massive extensions. John needed an even better and faster dump reader because he was reading dumps for lots of customers. After John left Amdahl, Keith Philip adopted Analyze and greatly extended it, out of a system programmer's love for a great tool (even though he officially works on something else).



John Alvord



Keith Philip

4. Really good software is never finished; it continues to grow. If it doesn't grow, it decays.

If a program or a system is good enough for users to exercise to its limits, they will always run up against those limits and will want them removed. They will always have ideas for making it more powerful and more useful. If a program or system doesn't grow to fulfill users' evolving requirements, someone will write a replacement that will.

You may have been surprised that I would cite Smart and VMAP as examples of really good software, since today there are clearly superior tools available. Yet, when they were first released, VMAP and Smart greatly enhanced our ability to understand and support our systems. Unfortunately, they have been allowed to stagnate since then.

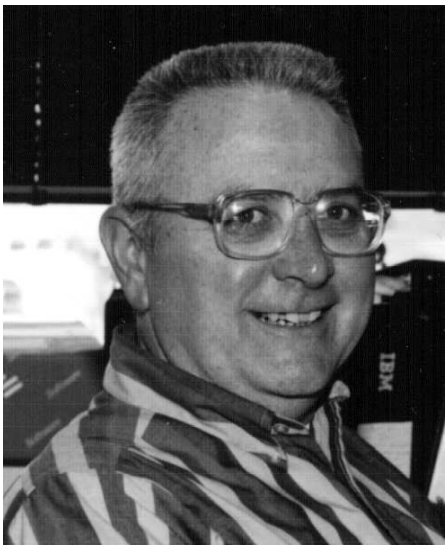
An even more extreme example is IPCS Dumpscan. We've all been scoffing at Dumpscan for so many years that it is difficult now to remember how revolutionary it was when it first came out, how tremendously it increased our productivity. Yet, I remember the day I learned to read dumps with Dumpscan as clearly as I remember the day I learned to read, and I remember feeling the same sort of triumph on both those days. Dumpscan was the brainchild of the late Dick Seymour. His original intent was not the creation of a tool to be used by customers or even by the Change Teams; Seymour was exploring the possibilities for automatic analysis of system failures. Dumpscan was an unexpected byproduct of that study.

---

[8] Now with Candle Corporation.

The primary author of Dumpscan was John Shaw.[9] Larry Estelle, a Regional VM Specialist who had been a CP developer, demonstrated Dumpscan's potential as a service tool by dialing into a customer system and using Dumpscan to shoot dumps. Ultimately, Dumpscan was made available to customers, whereupon it quickly became the object of more customer enhancements than almost any other part of VM. Once people saw the usefulness of such a tool, other, far superior dump viewers soon appeared, while IBM allowed Dumpscan to waste away of neglect.

Dick Seymour



Larry Estelle

Tom Pattison and John Shaw

[9] Mike Ness and Tom Pattison also worked on Dumpscan and other parts of IPCS.

Software that we all recognize as good continues to grow. REXX is still evolving. One hears tales of Xavier's having attempted to sneak new features into Xedit after the cutoff date for its first release. Hardly a week goes by without at least one new feature being added to Track and to *CMS Pipelines*.

5. **To evolve into really good software, a program or system must be "hacked at".**

   There's no point in telling designers and programmers to "do it right the first time". In fact, one learns what a program really should do only by writing it and using it. I think it's safe to say that the first few designs and at least the first implementation should always be discarded, once the lessons they can teach have been learned. If an organization wishes to create really good software, software that will greatly expand the uses of its computers, then the programmers must be free to rework the design, the structure, and the code over and over until they are almost as good as they can possibly be.

6. **To evolve into really good software and remain good, a program or system must develop an intelligent, adventurous, passionate user community with enough influence to be able to guide its further development.**

   No matter how good a piece of software is, in order to succeed it must have enthusiastic users who will proselytize for it and help teach others to use it. If it fails to win such support, it will die. If it does attract such support, then no matter how brilliant and complete its design, the users will always want to use it to do things the author never imagined. If the author is unable or unwilling to get and use feedback from the users to guide his further development of the software, then much of his effort will be wasted in adding features the users don't care about. When software is good enough that users are concerned about its future and willing to contribute their time and ideas to provide direction, *and* the author or developer is willing to accept their direction, then the result can be a positive feedback loop, with the software becoming ever more successful, ever more widely used.

   VM itself is the ultimate example of this. The periods of its greatest improvement have been those when there were close ties between the developers and the users and when the developers were highly responsive to the users' needs and concerns. VM's only real failures have come in times when those ties were less close and when the developers appeared to be paying less attention to the users.

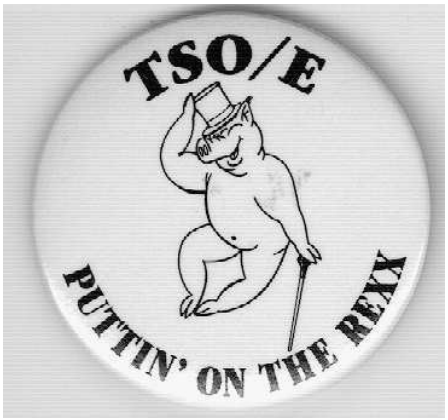
   Throughout VM's history, real end users have played an indispensable role in teaching others to use CMS. An interesting recent example of this is seen in the spread of *CMS Pipelines* within IBM. As it matured, *CMS Pipelines* was distributed throughout IBM *via* the VNET network. Experienced *CMS Pipelines* users (known as "master plumbers") soon developed a self-study course to help others learn to use *CMS Pipelines*.[10] This self-study course later evolved into the excellent *CMS Pipelines Tutorial* (GG66-3158).

   Another very successful example of the role knowledgeable users play is seen in the evolution of REXX. Mike Cowlishaw made the decision to write a new CMS executor on March 20, 1979. Two months later, he began circulating the first implementation of the new language,

[10] "Larry Kraines wrote the first eight lessons of the course originally. John Lynn wrote one when Larry got busy elsewhere. Many other people contributed improvements after that." (J.P. Hartmann, private communication, 1990.)

which was then called "REX". Once Mike made REX available over VNET,[11] users spontaneously formed the REX Language Committee, which Mike consulted before making further enhancements to the language. He was deluged with feedback from REX users, to the extent of about 350 mail files a day. By consulting with the Committee to decide which of the suggestions should be implemented and which pieces of contributed code should be incorporated, he rather quickly created a monumentally successful piece of software. When REXX celebrated its tenth birthday recently, it was still spreading to new systems and delighting new users.[12]

"/ Best Language of the Year */"*



"TSO/E—Puttin' on the REXX"

Frequently, people who use a really good program or system will care so much about it that they will add desired new function themselves, if there are facilities for their doing so. If the author of the software makes a point of learning about the most successful of these new functions, he can gain much guidance in developing his product further.

If you will keep this list in mind as I outline the history of VM, I think you may notice other cases where these factors resulted in innovations that accelerated the growth of our systems.

---

[11] "The most important influence on the development of the REXX language was the IBM internal electronic network, VNET. Without the network (and the people who keep it running), there would have been little incentive to start a task of this magnitude; and without the constant flow of ideas and feedback from people using the network REXX would have been a much poorer language. Much credit for the effectiveness of VNET as a communication medium for this sort of work is due to Peter Capek who created the *VM Newsletter* (1973-1983). Today, REXX language design is carried out over the same network almost entirely with the aid of the Tools computer conference system—appropriately enough, a system written in REXX." (M.F. Cowlishaw, *The REXX Language: A Practical Approach to Programming*, Prentice Hall, second edition, 1990, p. *x.)*

[12] Cowlishaw was made an IBM Fellow in 1990.

III. A BRIEF HISTORY OF VM

I owe many kind VMers thanks for having shared their memories and their memorabilia with me so that I could share them with you.[13] I regret that this account will leave out many people who have made good and lasting contributions to VM, but this is inevitable, given our time constraints and my ignorance of much of what has happened inside IBM during the past twenty-five years. I would be delighted to hear any corrections or additions you may have to story I'm about to tell.

A. CTSS

In the beginning was CTSS, the "Compatible Time-Sharing System". CTSS was written by a small group of programmers[14] at the Massachusetts Institute of Technology (MIT) in Cambridge, Massachusetts, under the leadership of Professor Fernando Corbató. One of the CTSS programmers was Robert Creasy, who was later to become the leader of the CP-40 project.

Papers discussing the idea of a time-sharing system began being published about 1959.[15] There followed a period of experimentation at MIT and other institutions. An early version of CTSS was demonstrated on an IBM 709 at MIT in November, 1961. From that beginning, CTSS evolved rapidly over the next several years and taught the world how to do time-sharing.[16]

Fernando Corbató

[13] I've managed to locate most of the people mentioned in this chapter. Without exception, they have been extremely generous with their time and assistance, regaling me with delightful tales of their days in VM and patiently enduring my endless questions. I am grateful to them all. I am also indebted to the people who have searched out physical evidence for me: the system programmers at Brown University, who donated an intact PID shipment of CP-67 Version 3; Bill Frantz, Scott Tyree, and Walt Hutchens, who sent me stacks of early manuals; Chuck Rodenberger, who sought out dozens of old "blue letters"; Alan Greenberg and Dewayne Hendricks, who sent me their archives from SHARE and GUIDE activities in the CP-67 and early VM/370 days; David Walker and Jacques Myon, who unearthed some amazing artifacts; Bob Cox, Gabe Goldberg, and Chuck Morse, who sent slide sets from VM demonstrations and announcements; Dave Tuttle, who wrote out his memoirs and also lent me an astonishing collection of VM relics; Fernando Corbató, Les Comeau, and Don Wagler, who made me videotapes of rare old films; and Stu Madnick, who sent me the contents of his attic.

[14] Marjorie Merwin-Daggett, Robert Daley, Robert Creasy, Jessica Hellwig, Richard Orenstein, and Lyndalee Korn.

[15] C. Stratchey, "Time Sharing in Large Fast Computers," *Proceedings of the International Conference on Information Processing*, Paper B.2.19, UNESCO, New York, June, 1959.

CTSS was developed on a series of IBM processors. In the 1950's, IBM's president, T.J. Watson, Jr., had very shrewdly given MIT an IBM 704 for use by MIT and other New England schools.[17] Then, each time IBM built a newer, bigger processor, it upgraded the system at MIT.[18] The 704 was followed by a 709, then by a 7090, and finally by a 7094. IBM also gave MIT the services of some highly skilled Systems Engineers and Customer Engineers, who formed its MIT Liaison Office, which was housed at the MIT Computation Center.

---

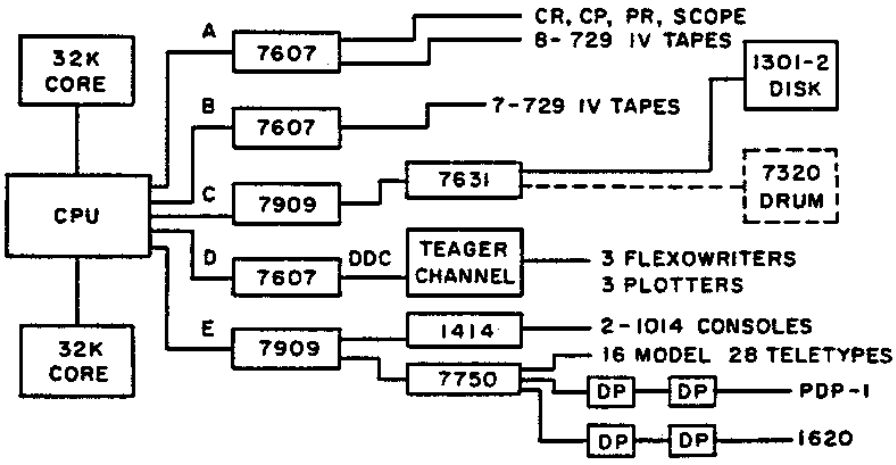
[16] From the preface to the "candy-striped manual" (F.J. Corbató, M.M. Daggett, R.C. Daley, R.J. Creasy, J.D. Hellwig, R.H. Orenstein, and L.K. Korn, *The Compatible Time-Sharing System: A Programmer's Guide*, The MIT Press, Cambridge, Mass., 1963):

> The only other general purpose time-sharing system known to be operating presently, that of the Bolt, Beranek and Newman Corporation for the PDP-1 computer, was recently described by Professor John McCarthy at the 1963 Spring Joint Computer Conference. Other time-sharing developments are being made at the Carnegie Institute of Technology with a G20 computer, at the University of California at Berkeley with a 7090, at the Rand Corporation with Johnniac, and at MIT (by Professor Dennis) with a PDP-1. Several systems resemble our own in their logical organization; they include the independently developed BBN system for the PDP-1, the recently initiated work at IBM (by A. Kinslow) on the 7090 computer, and the plans of the System Development Corporation with the Q32 computer.
>
> To establish the context of the present work, it is informative to trace the development of time-sharing at MIT. Shortly after the first paper on time-shared computers, by C. Strachey at the June 1959 UNESCO Information Processing Conference, H.M. Teager and J. McCarthy at MIT delivered an unpublished paper *Time-Shared Program Testing* at the August 1959 ACM Meeting. Evolving from this start, much of the time-sharing philosophy embodied in the CTSS system has been developed in conjunction with an MIT preliminary study committee (initiated in 1960), and a subsequent working committee. The work of the former committee resulted, in April 1961, in an unpublished (but widely circulated) internal report. Time-sharing was advocated by J. McCarthy in his lecture, given at MIT, contained in *Management and the Computer of the Future* (MIT, 1962). Further study of the design and implementation of man-computer interaction systems is being continued by a recently organized institution-wide project under the direction of Professor Robert M. Fano.

[17] "Our customers often complained that the most difficult thing about having a computer was hiring somebody who could run it. They'd ask for help, we couldn't provide all those technicians ourselves, and there was not a single university with a computer curriculum. Sometimes we even found ourselves in a position where we had to hold back from taking a customer's order. So I went up to MIT in 1955 and urged them to start training computer scientists. We made a gift of a large computer and the money to run it, and they shared that machine with ten other schools in the Northeast. For the 650, we adapted a very aggressive college discount program that existed for our punch-card machines: you could get 40 percent off for setting up a course in either business data processing or scientific computing, and 60 percent off for setting up courses in both. I put these education policies near the top of the list of IBM's key moves, because within five years there was a whole new generation of computer scientists who made it possible for the market to boom." (T.J. Watson, Jr., *Father, Son, and Co.: My Life at IBM and Beyond*, Bantam Books, New York, 1990, pp. 244-5.)

[18] It appears that (without a clear directive from Corporate management) IBM's Cambridge Branch Office decided to interpret Watson's original grant to MIT as authorization for them to upgrade the system at MIT whenever IBM produced a more powerful computer.

As CTSS evolved, Professor Corbató and his students and colleagues began to encounter problems that they knew were better addressed by hardware than by software, so they asked IBM for modifications to their processor. The IBMers in the Liaison Office had the job of finding engineers in Poughkeepsie to build the hardware extensions that MIT had determined were necessary to do time-sharing properly. By the time CTSS was in full production in 1963, the 7090 at MIT had been modified to have a second memory bank (32K words), an address relocation register, and memory protection. With these extensions to the hardware, Corbató's group was able to build CTSS into the system that became the exemplar for time-sharing systems.



MIT's Modified 7090/94 System, 1963 [19]

A CMS user would find the log of a CTSS session fairly easy to follow. Commands were composed of 6-character, blank-delimited tokens. Files were referenced by their file name and file class. File attributes, such as permanent, read-only, etc., were determined by the file mode. Some commands, such as "START", "LOAD", "RENAME", and "LISTF", would be quite familiar. The system typed "READY" when it completed the processing of a command. (It also typed "WAIT" when it started the processing of a command, so response time was obviously not what it is today.)

Nor would the system internals (as described in the 1963 CTSS user's guide) be entirely strange:

> The consoles of CTSS form the foreground system, with computation being performed for the active console users in variable-length bursts, on a rotation basis, according to a scheduling algorithm. The background system is a conventional programming system (slightly edited for the time-sharing version) which, at the least, operates whenever the foreground system is inactive, but which may also be scheduled for a greater portion of the computer time. The entire operation of the computer is under the control of a supervisor program which remains permanently in the 32,768 word A-bank of core memory; all user programs are either kept in the 32,768 word B-bank of core memory, or swapped in and out of the disk (or drum) memory as needed.[20]

---

[19] Corbató *et al*, *Ibid.*, p. 3.

> The supervisor program ... functions include: handling of all input and output for the consoles; scheduling of console-initiated (foreground) and offline-initiated (background) jobs; temporary storage and recovery of programs during scheduled swapping; monitoring of all input and output from the disk, as well as input and output performed by the background system; and performing the general role of monitor for all foreground jobs. These tasks can be carried out by virtue of the supervisor's direct control of all trap interrupts, the most crucial of which is the one associated with the Interval Timer Clock.[21]

By trapping interrupts, the CTSS supervisor controlled and isolated users in a manner very similar to the way the VM Control Program does this same thing today. CTSS users could request supervisor services by causing a protection exception, in much the same way that we use the CMS SVC and CP DIAGNOSE instructions today.

B. The Births of System/360, Project MAC, and the Cambridge Scientific Center

While CTSS was being developed in Cambridge, in Poughkeepsie IBM was designing the new family of computers on which it had staked its future, System/360. MIT by then was committed to time-sharing and was providing CTSS services to several other New England universities as well as to its own users. At MIT, it was "no longer a question of the feasibility of a time-sharing system, but rather a question of how useful a system [could] be produced".[22] The IBMers in the MIT Liaison Office and the Cambridge Branch Office, being well aware of what was happening at MIT, had become strong proponents of time-sharing and were making sure that the System/360 designers knew about the work that was being done at MIT and understood the purpose of the modifications to the 7090. They arranged for several of the leading System/360 architects to visit MIT and talk with Professor Corbató. However, inside IBM at that time there was a strong belief that time-sharing would never amount to anything and that what the world needed was faster batch processing. MIT and other leading-edge customers were dismayed, and even angered, on April 7, 1964, when IBM announced System/360 *without* address relocation capability.

The previous Fall, MIT had founded Project MAC[23] to design and build an even more useful time-sharing system based on the CTSS prototype. Within Project MAC, Corbató and others were to draw on the lessons they had learned from CTSS to build the Multics system. The basic goal of the Multics project "was to develop a working prototype for a computer utility embracing the whole complex of hardware, software, and users that would provide a desirable, as well as feasible, model for other system designers to study."[24] At the outset, Project MAC purchased a second modified 7094 on which to run CTSS while developing Multics. It then requested bids

---

[20] Corbató *et al*, *Ibid.*, p. 2.

[21] Corbató *et al*, *Ibid.*, p. 8.

[22] F.J. Corbató, in P.A. Crisman, ed., *The Compatible Time-Sharing System: A Programmer's Guide*, second edition, MIT Press, Cambridge, Mass., 1965, p. 1.

[23] "MAC" may have been an acronym for "Machine-Aided Cognition" or "Multiple-Access Computing" or "Man and Computer".

[24] F.J. Corbató, in E.I. Organick, *The Multics System: An Examination of Its Structure*, The MIT Press, Cambridge, Mass., 1972, p. *ix.*

for the processor on which Multics would run.

In February of 1964, IBM had sent Norm Rasmussen[25] to Cambridge to establish what became the Cambridge Scientific Center (CSC). Cambridge in the 1960's was an exciting place, full of ferment. It was a congenial place for Rasmussen, who was very much a man of that era, and it was a congenial assignment, as well, because Rasmussen was eager to demonstrate that science has a role to play in the building of good software.

545 Tech Square, Cambridge

Norm Rasmussen

Rasmussen arranged for space for the Scientific Center in the same building as Project MAC, 545 Technology Square.[26] (For many years after that, the Scientific Center programmers and the Project MAC programmers would remain on friendly terms and would occasionally get together in the bar on the ground floor of that building after work.)

All of IBM's contractual relationships with MIT were turned over to the new Scientific Center to administer. The Scientific Center was also expected to take the lead in making IBM "respectable" to the academics. So, only weeks after he had arrived in Cambridge, Rasmussen had to deal with MIT's very negative reaction to System/360. Within days of the System/360 announcement, the chief S/360 architect, Gene Amdahl, came to Cambridge to meet with Professor Corbató and his colleagues, but that meeting seems only to have made matters worse.

As a loyal IBMer, Rasmussen was deeply embarrassed by IBM's failure to heed the advice of such an important customer, and he became determined to make things right, to do whatever was necessary to make System/360 right for MIT and other customers. To do that, he knew that he would need very talented people, so he set about attracting the best people he could find. He was fortunate to be able to start by taking over the staff of IBM's MIT Liaison Office. From the Liaison Office came two very skilled Systems Engineers, Les Comeau and John Harmon, as well

---

[25] Now with Teleprocessing, Inc.

[26] The building was a hotbed of time-sharing activity. "At one time in the mid-60's, I counted ten or fifteen time-sharing systems being coded or tested or accessed in Tech Square." (J.B. Harmon, private communication, 1989.)

as a quiet, unassuming Customer Engineer named Fritz Giesin, who would come to be treasured by generations of programmers at the Center.[27] Next came another excellent IBM programmer, Ron Brennan, from the Federal Systems Division. Shortly after that, one of the seven CTSS authors, Lyndalee Korn, left MIT to join the Center.

One of the first jobs for the staff of the new Center was to put together IBM's proposal to Project MAC. In the process, they brought in many of IBM's finest engineers to work with them to specify a machine that would meet Project MAC's requirements, including address translation. They were delighted to discover that one of the lead S/360 designers, Gerry Blaauw, had already done a preliminary design for address translation on System/360.[28] Unfortunately, he had been unable to convince his superiors to incorporate that function into the basic S/360 design.

Gerrit Blaauw

The machine that IBM proposed to Project MAC was a S/360 that had been modified to include the "Blaauw Box". This machine was also bid to Bell Labs at about the same time. It was never built, however, because both MIT and Bell Labs chose another vendor. MIT's stated reason for rejecting IBM's bid was that it wanted a processor that was a main-line product, so that others could readily acquire a machine on which to run Multics. It was generally believed, however, that displeasure with IBM's attitude toward time-sharing was a factor in Project MAC's decision.

---

[27] For the next twenty-five years, Giesin would perform daily hardware miracles in support of the Cambridge programmers. "Fritz was a stalwart of the Center, very dependable, Santa Claus in his goodie lab!" (J.B. Harmon, private communication, 1990.)

[28] G.A. Blaauw, *Relocation Feature Functional Specification*, June 12, 1964. "Nat Rochester (one of the designers of the 701) told us, 'Only one person in the company understands how to do address translation, and that's Gerry Blaauw. He has the design on a sheet of paper in his desk drawer.'" (R.J. Brennan, private communication, 1989.)

Losing Project MAC and Bell Labs had important consequences for IBM. Seldom after that would IBM processors be the machines of choice for leading-edge academic computer science research. Project MAC would go on to implement Multics on a GE 645 and would have it in general use at MIT by October, 1969. Also in 1969, the system that was to become UNIX would be begun at Bell Labs as an offshoot and elegant simplification of both CTSS and Multics, and that project, too, would not make use of IBM processors.

But getting back to the Summer of 1964: Norm Rasmussen had just begun his fight to make System/360 acceptable to the academics and was not having an easy time of it. During that summer, Professor Corbató (a man widely known for his gentlemanliness) published a Project MAC Report containing a devastating analysis of the weaknesses of the S/360 as a machine on which to implement a time-sharing system.[29] Other customers were also expressing concern about the lack of time-sharing capability in System/360. In August, SHARE's Advanced Planning Division presented a survey of the currently operating on-line programming systems. One of the speakers was Fernando Corbató, who emphasized the potential for growth of the computing industry due to time-sharing.[30]

Rasmussen's response to all this was to decide that the Cambridge Scientific Center would write a time-sharing system for the System/360.

Meanwhile, inside Project MAC, Bob Creasy was upset by the inability of his colleagues to come to terms with IBM. He was impressed by the promise of machine upward compatibility offered by S/360,[31] and he wanted Multics to be a mainstream system. When he heard that Rasmussen intended to build a time-sharing system based on S/360 and needed someone to lead the project, Creasy also left MIT to move to the Scientific Center.

---

[29] F.J. Corbató, *System Requirements for Multiple Access, Time-Shared Computers*, Project MAC Report MAC-TR-3, undated (probably August, 1964).

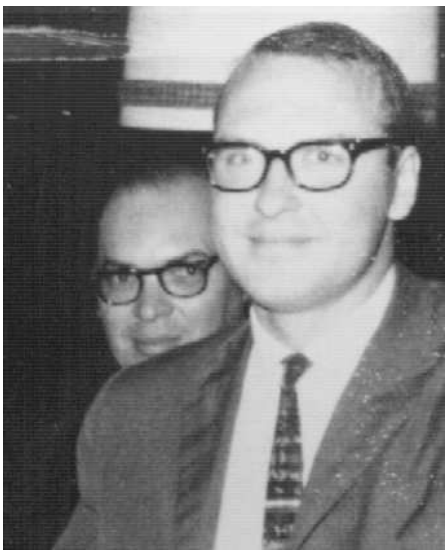
[30] "Corbató noted that FORTRAN probably added a factor of 10 to the number of computer users and that time sharing will at least equal this factor of 10 and probably surpass it." (*Proceedings of SHARE XXIII*, August, 1964, p. 3-18.) The systems described in addition to CTSS were JOSS, Quicktran, and SDC's system for the Q32.

[31] Creasy had, of course, spotted the most important aspect of the System/360 announcement, that programs written for one model of S/360 would run on any other model as long as they contained no timing-dependent code. From the System/360 "blue letter" (April 7, 1964):

> Whatever your customer's data handling requirements are now, whatever they will be in the foreseeable future, the System/360 can be custom-fitted to perform his job. In fact, this amazing new system makes possible, for the first time in the industry, a truly long-range growth plan for all customers. For it can accommodate virtually any combination of processing and computing functions. And can expand in easy, economical steps as the customer's needs change—with little or no reprogramming, no retraining of personnel, no disruption of service.

The decision to make System/360 a family of processors with the same instruction set throughout can safely be said to have made IBM's fortune.

Dick Bayles and Les Comeau



John Harmon and Bob Creasy

Inside IBM, losing the Project MAC bid was immediately recognized as a serious problem. A corporate task force was formed to get the company into a position to be able to win bids for time-sharing systems.[32] The task force was composed of the most knowledgeable time-sharing people from around the company. CSC was represented by Rasmussen, Harmon, Creasy, and Comeau. Other task force members included Andy Kinslow, who had written an experimental time-sharing system called BOSS ("Big Old Supervisory System") for the 7090, and John Moressey, the author of Quicktran. The products of this task force were rough specifications for a new processor, which would incorporate the Blaauw Box and be called the S/360 Model 67, and for a new operating system, which would be called TSS (the Time-Sharing System). IBM's management accepted the recommendations of the task force and put Andy Kinslow in charge of TSS Development. Rasmussen and his staff soon prepared a successful proposal for a 360/67 for MIT's Lincoln Laboratory.[33]

---

[32] The task force was known as the "Flewellin House Task Force", for the building on the grounds of IBM Research at Yorktown Heights in which it met.

[33] "Lincoln had a role in the design of the time-sharing machine. I have a copy of IBM's response to Lincoln's Request for Quotation, which specified a Model 66. This machine was later to become the 360/67, but I don't know why the model number changed. A group of six sites (Lincoln Lab, University of Michigan, Carnegie University, Bell Labs, General Motors, and Union Carbide, I believe) had a non-disclosure agreement for the development of the 360/66. This group was called the 'Inner Six'. At one meeting in Yorktown Heights, we met with IBM people to discuss relocation hardware. We discussed whether an address should be 31 or 32 bits. We eventually voted and recommended 31 bits. We also discussed the design of the relocation register and had some heated discussions with the IBM team. The Inner Six met with IBM representatives behind closed doors at a SHARE meeting. We six sites discussed various features of TSS and made recommendations to IBM. This was the beginning of the SHARE TSS Project." (J.M. Winett, private communication, 1990.)

C. CP-40 and CMS

In the Fall of 1964, the folks in Cambridge suddenly found themselves in the position of having to cast about for something to do next. A few months earlier, before Project MAC was lost to GE, they had been expecting to be in the center of IBM's time-sharing activities. Now, inside IBM, "time-sharing" meant TSS, and that was being developed in New York State. However, Rasmussen was very dubious about the prospects for TSS and knew that IBM must have a credible time-sharing system for the S/360. He decided to go ahead with his plan to build a time-sharing system, with Bob Creasy leading what became known as the CP-40 Project.

The official objectives of the CP-40 Project were the following:

- The development of means for obtaining data on the operational characteristics of both systems and application programs;
- The analysis of this data with a view toward more efficient machine structures and programming techniques, particularly for use in interactive systems;
- The provision of a multiple-console computer system for the Center's computing requirements; and
- The investigation of the use of associative memories in the control of multi-user systems.[34]

The project's real purpose was to build a time-sharing system, but the other objectives were genuine, too, and they were always emphasized in order to disguise the project's "counter-strategic" aspects.

Rasmussen consistently portrayed CP-40 as a research project to "help the troops in Poughkeepsie" by studying the behavior of programs and systems in a virtual memory environment. In fact, for some members of the CP-40 team, this was the most interesting part of the project, because they were concerned about the unknowns in the path IBM was taking. TSS was to be a virtual memory system, but not much was really known about virtual memory systems. Les Comeau has written:

> Since the early time-sharing experiments used base and limit registers for relocation, they had to roll in and roll out entire programs when switching users....Virtual memory, with its paging technique, was expected to reduce significantly the time spent waiting for an exchange of user programs.
>
> What was most significant was that the commitment to virtual memory was backed with no successful experience. A system of that period that had implemented virtual memory was the Ferranti Atlas computer, and that was known not to be working well. What was frightening is that nobody who was setting this virtual memory direction at IBM knew why Atlas didn't work.[35]

---

[34] R.J. Adair, R.U. Bayles, L.W. Comeau, and R.J. Creasy, *A Virtual Machine System for the 360/40*, IBM Cambridge Scientific Center Report 320-2007, Cambridge, Mass., May, 1966.

[35] L.W. Comeau, "CP-40, the Origin of VM/370", *Proceedings of SEAS AM82*, September, 1982, p. 40.

Creasy and Comeau spent the last week of 1964[36] joyfully brainstorming the design of CP-40, a new kind of operating system, a system that would provide not only virtual memory, but also virtual *machines*.[37] They had seen that the cleanest way to protect users from one another (and to preserve compatibility as the new System/360 design evolved) was to use the System/360 *Principles of Operations* manual to describe the user's interface to the Control Program. Each user would have a complete System/360 virtual machine (at first called a "pseudo-machine").[38]

The idea of a virtual machine system had been bruited about a bit before then, but it had never really been implemented. The idea of a virtual S/360 was new, but what was really important about their concept was that nobody until then had seen how elegantly a virtual machine system could be built, with really very minor hardware changes and not much software.

Creasy and Comeau were soon joined on the CP-40 Project by Dick Bayles,[39] from the MIT Computation Center, and Bob Adair, from MITRE. Together, they began implementing the CP-40 Control Program, which sounds familiar to anyone familiar with today's CP. Although there were a fixed number (14) of virtual machines with a fixed virtual memory size (256K), the Control Program managed and isolated those virtual machines in much the way it does today.[40] The Control Program partitioned the real disks into minidisks and controlled virtual machine access to the disks by doing CCW translation. Unit record I/O was handled in a spool-like fashion. Familiar CP console functions were also provided.

---

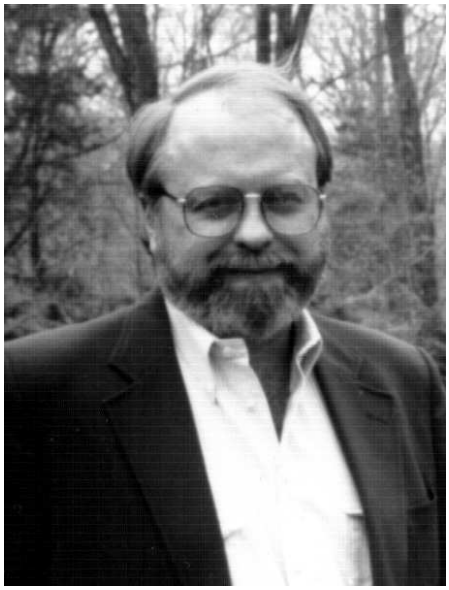
[36] Creasy had decided to build CP-40 while riding on the MTA. "I launched the effort between Xmas 1964 and year's end, after making the decision while on an MTA bus from Arlington to Cambridge. It was a Tuesday, I believe." (R.J. Creasy, private communication, 1989.)

[37] R.J. Creasy, *General Description of the Research Time-Sharing System with Special Emphasis on the Control Program*, IBM Cambridge SR&D Center Research Time-Sharing Computer Memorandum 1, Cambridge, Mass., January 29, 1965. L.W. Comeau, *The Philosophy and Logical Structure of the Control Program*, IBM Cambridge SR&D Center Research Time-Sharing Computer Memorandum 2, Cambridge, Mass., April 15, 1965.

[38] For the first few weeks, the CSC people referred to their concept as a "pseudo-machine", but soon adopted the term "virtual machine" after hearing Dave Sayre at IBM Research use it to describe a system he had built for a modified 7044. Sayre's M44 system was similar to CP-40, except for the crucial difference of not providing a control program interface that exactly duplicated a real machine. The CP-40 team credited Sayre with having "implanted the idea that the virtual machine concept is not necessarily less efficient than more conventional approaches." (L. Talkington, "A Good Idea and Still Growing", *White Plains Development Center Newsletter*, vol. 2, no. 3, March, 1969.) "The system built by Dave Sayre and Bob Nelson was about as much of a virtual machine system as CTSS—which is to say that it was close enough to a virtual machine system to show that 'close enough' did not count. I never heard a more eloquent argument for virtual machines than from Dave Sayre." (R.J. Creasy, private communication, 1990.)

[39] "Dick Bayles was not only a great programmer, he was also the fastest typist I have ever seen." (W.J. Doherty, private communication, 1990.) "When Dick Bayles sat down [at a keypunch], he wrote code as fast as it could punch cards. Yes, the machine was slower than Bayles composing code on the fly." (R.J. Creasy, private communication, 1989.)

[40] R.A. Meyer and L.H. Seawright, "A Virtual Machine Time-Sharing System", *IBM Systems Journal*, vol. 9, no. 3, 1970, pp. 199-218.



Dick Bayles

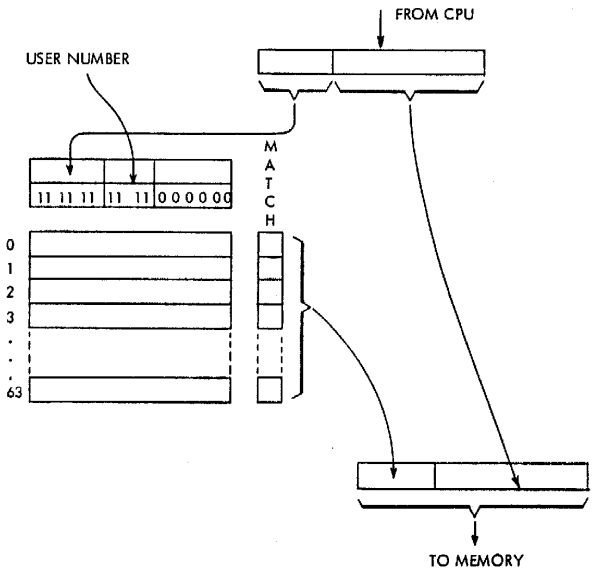
Bob Adair

This system could have been implemented on a 360/67, had there been one available, but the Blaauw Box wasn't really a measurement tool. Even before the design for CP-40 was hit upon, Les Comeau had been thinking about a design for an address translator that would give them the information they needed for the sort of research they were planning. He was intrigued by what he had read about the associative memories that had been built by Rex Seeber and Bruce Lindquist in Poughkeepsie, so he went to see Seeber with his design for the "Cambridge Address Translator",[41] which was based on the use of associative memory and had "lots of bits" for recording various states of the paging system.

Rex Seeber and Bruce Lindquist

The Cambridge 360/40

[41] "The CAT was a jewel." (R.J. Creasy, private communication, 1989.)



360/40 Address Translation [42]

Seeber liked the idea, so Rasmussen found the money to pay for the transistors and engineers and microcoders that were needed, and Seeber and Lindquist implemented Comeau's translator on a S/360 Model 40.[43] Comeau has written:

> Virtual memory on the 360/40 was achieved by placing a 64-word associative array between the CPU address generation circuits and the memory addressing logic. The array was activated *via* mode-switch logic in the PSW and was turned off whenever a hardware interrupt occurred.
>
> The 64 words were designed to give us a relocate mechanism for each 4K bytes of our 256K-byte memory. Relocation was achieved by loading a user number into the search argument register of the associative array, turning on relocate mode, and presenting a CPU address. The match with user number and address would result in a word selected in the associative array. The position of the word (0-63) would yield the high-order 6 bits of a memory address. Because of a rather loose cycle time, this was accomplished on the 360/40 with no degradation of the overall memory cycle.[44]

---

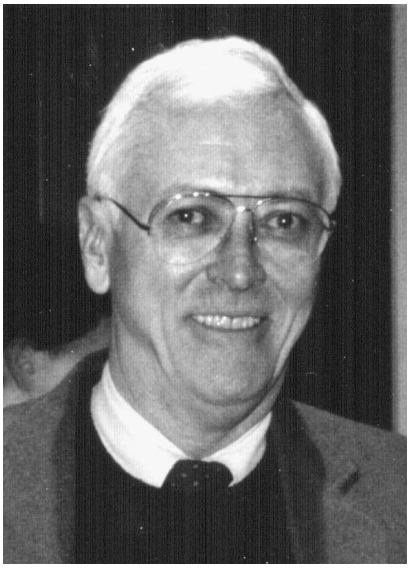
[42] Adair *et al*, *op. cit.*, p. 13.

[43] A.V. Lindquist, R.R. Seeber, and L.W. Comeau, "A Time-Sharing System Using an Associative Memory", *Proceedings of the IEEE*, vol. 54, no. 12, December, 1966, pp. 1774-9. The Center actually wanted a 360/50, but all the Model 50s that IBM was producing were needed for the Federal Aviation Administration's new air traffic control system.

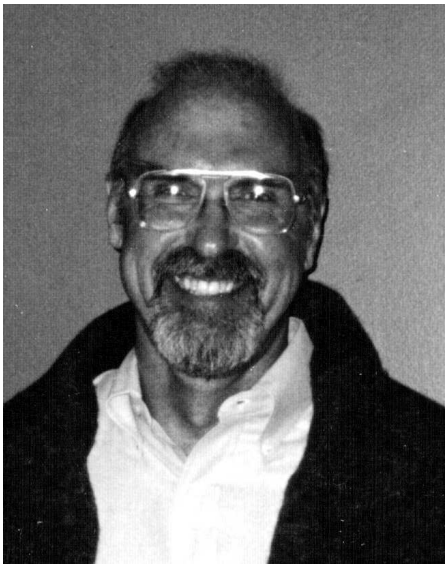
[44] Comeau, *op. cit.*, p. 41.

The modifications to the 360/40 would prove to be quite successful, but it would be more than a year before they were complete. Dick Bayles has described the process that he and Comeau and Giesin went through in debugging the modifications:

> One of the fun memories of the CP-40 Project was getting involved in debugging the 360/40 microcode, which had been modified not only to add special codes to handle the associative memory, but also had additional microcode steps added in each instruction decoding to ensure that the page(s) required for the operation's successful completion were in memory (otherwise generating a page fault).
>
> The microcode of the 360/40 comprised stacks of IBM punch card-sized Mylar sheets with embedded wiring. Selected wires were "punched" to indicate 1's or 0's. Midnight corrections were made by removing the appropriate stack, finding the sheet corresponding to the word that needed modification, and "patching" it by punching a new hole or by "duping" it on a modified keypunch with the corrections.[45]



Les Comeau



Bob Creasy

Back during that last week of 1964, when they were happily working out the design for the Control Program, Creasy and Comeau immediately recognized that they would need a second system, a console monitor system, to run in some of their virtual machines. Although they knew that with a bit of work they would be able to run any of IBM's S/360 operating systems in a virtual machine, as contented users of CTSS, they also knew that they wouldn't be satisfied using

---

[45] R.U. Bayles, private communication, 1989. "The Model 40 was a Hursley (UK) product, announced in 1964. It used the first programmable ROS (invented by Tony Proudman, I believe) called Transformer Read-Only Storage (developed in 1961/2). In the Model 40 the circuit on the Mylar sheets wound around 60 cores, hence allowing the storage of 60-bit words; the Model 40 had 4096 60-bit words. It was this re-programmable storage that made the Model 40 modifiable, as you describe." (M.F. Cowlishaw, private communication, 1990.)

any of the available systems for their own development work or for the Center's other time-sharing requirements. Rasmussen, therefore, set up another small group under Creasy to build CMS (which was then called the "*Cambridge* Monitor System"). The leader of the CMS team was John Harmon.[46] Working with Harmon were Lyndalee Korn and Ron Brennan.

Like Multics, CMS would draw heavily on the lessons taught by CTSS. Indeed, the CMS user interface would be *very* much like that of CTSS.

Since each CMS user would have his own virtual machine, CMS would be a single-user system, unlike CTSS. This was an important factor in the overall simplicity and elegance of the new system.[47] Creasy has written that one of the most important lessons they had learned from their CTSS experience was "the necessity of modular design for system evolution. Although [CTSS was] successful as a production system, the interconnections and dependencies of its supervisor design made extension and change difficult."[48]

CP-40 would be far more modular than CTSS, in that it would be divided into two independent components. In the words of Bob Creasy:

> A key concept of the CP/CMS design was the bifurcation of computer resource management and user support. In effect, the integrated design was split into CP and CMS. CP solved the problem of multiple use by providing separate computing environments at the machine instruction level for each user. CMS then provided single user service unencumbered by the problems of sharing, allocation, and protection.[49]

As the weeks went by and the real power of the virtual machine concept unfolded before them, their excitement grew. In discussing the decision to create exact replicas of real machines, Les Comeau has written, "It seems now that the decision to provide a Control Program interface that duplicated the System/360 architecture interface was an obvious choice. Although it was, given our measurement objective, it wasn't, given our in-house interactive system objective."[50] He credits "the strong wills and opinions of the group" for providing further motivation for selecting such a well-defined interface[51] between the CP and CMS components:

---

[46] J.B. Harmon, *General Description of the Cambridge Monitor System*, IBM Cambridge SR&D Center Research Time-Sharing Computer Memorandum 3, Cambridge, Mass., May 12, 1965.

[47] Bob Creasy has commented, "Simplicity was important because of our limited resource. I didn't expect the design [of CMS] to hold for more than a couple of years. We recognized the importance of multiple processes in a single-user environment, but we could not afford the complexity. To put it another way, we weren't smart enough to make it simple enough." (R.J. Creasy, private communication, 1990.)

[48] R.J. Creasy, "The Origin of the VM/370 Time-Sharing System", *IBM Journal of Research and Development*, vol. 25, no. 5, September, 1981, p. 485.

[49] Creasy, *op. cit.*, p. 485.

[50] Comeau, *op. cit.*, p. 43.

[51] "The CP group would not discuss what they were doing with us. They just said, 'Read the *Principles of Ops*.'" (J.B. Harmon, private communication, 1989.) "I credit Bob Adair as the non-compromising standard bearer of [this aspect of] the CP design. Later, Ed Hendricks

> I think that most designers recognize the need for good separation of function in programming system design, but compromise becomes the rule very early in the effort. With the particular group assembled to build CP/CMS, the personalities reinforced that design principle, rather than compromising it.
>
> The choice of an architected interface, the System/360, ... turned out to have been most fortunate. It permitted simultaneous development of CP and CMS; it allowed us to measure non-virtual systems, OS and DOS, in a virtual memory environment, and it also provided a high level of integrity and security.[52]

John Harmon

Ron Brennan

CMS at this stage was rudimentary, consisting of structural elements on a blackboard (which Harmon updated daily).[53] "At that time the file system was viewed as supporting the needs of VM as well as the rest of CMS. Performance and simplicity were seen as major goals, because there was much concern about the overhead that would be introduced by the time-sharing function. Performance was achieved by keeping the function minimal and the path lengths short."[54]

---

would join that very small group." (R.J. Creasy, private communication, 1990.)

[52] Comeau, *op. cit.*, pp. 42-43.

[53] "John, understanding operating systems and programmers, constantly applied the KISS principle." (R.J. Creasy, private communication, 1990.)

[54] R.J. Brennan, private communication, 1989.

The file system design was clearly crucial. If CMS were to be attractive to use, it had to have "a convenient and simple method" for reading and writing disks. Ron Brennan left to work on TSS in April, 1966, but before he left Cambridge he had completed specifications for the CMS file system[55] (which was then called the "Disk Service Program") and had begun the implementation. In producing his design, Brennan drew upon his own knowledge of CTSS and upon Adair's knowledge of the file system for STRETCH.[56] By September of 1965, file system commands and macros already looked much like those we are familiar with today: "RDBUF", "WRBUF", "FINIS", "STATE", etc. Many of the decisions that were to be key to the elegance of the CMS file system had already been made:

- CMS would use the simple filename-filetype-filemode naming convention, rather than using OS-like file names;
- Records would be mapped to fixed-size blocks;
- Records could be read or written by relative record number;[57]
- From the user's view, a file would be created simply by writing to it; and
- In many commands, the filemode could be defaulted, in which case the disks would be searched in a fixed order.

Another major focus of the CMS team was to determine the nature of the command language:

> It was clear, based upon the experience gained with CTSS, that a user-friendly command language was key. Another thing we had learned was that the system had to be very forgiving, and although options were desirable, default-mode, non-required parameters were to be a paramount design consideration in CMS.[58]

While his staff worked at designing the new system, Rasmussen sought ways to pay for it, taking advantage of every bit of luck that came his way. When IBM gave the 7094 to the MIT Computation Center, it retained the night shift on that machine for its own use. So, because the Scientific Center had inherited IBM's contracts with MIT, Rasmussen "owned" eight hours of 7094 time per day. He traded part of that time to the Computation Center for CTSS time for his programmers to use in doing their development work. He "sold" the remainder to IBM hardware developers in Poughkeepsie, who badly needed 7094 time to run a design automation program that was critical for S/360 hardware development. With the internal funds he acquired this way, he paid for the modifications to the Model 40. Although he could not use these funds to pay for regular "head-count" employees, he could use them to pay for part-time employees, mainly MIT students, and to pay the salaries of IBMers who came to Cambridge to work on the system while remaining in the "head count" of some other part of the company. This method of funding the project with unbudgeted revenues had the advantage of allowing it to keep a very low profile.

---

[55] R.J. Brennan, *Disk Service Program User Specifications*, IBM Cambridge SR&D Center Research Time-Sharing Computer Memorandum 4, Cambridge, Mass., September 16, 1965.

[56] "I gave Ron full responsibility for the file system and he did all the work himself—an excellent job." (R.J. Creasy, private communication, 1990.)

[57] "Interestingly, some of the byproducts of this simple design, such as mapping the file records to fixed-size blocks on disk storage and retrieving the imbedded records by a relative block number, were precursors of the 'fixed block architecture' and the 'direct file access method' used by so many of IBM's later systems." (R.J. Brennan, private communication, 1989.)

[58] Comeau, *op. cit.*, p. 43.

Norm Rasmussen

Another bit of luck during the first year was a surprise visit to the Scientific Center by IBM's much-feared President, T. Vincent Learson (known informally as "The Hatchet Man"). Rasmussen was at lunch when Learson arrived, so Lyndalee Korn was given the task of entertaining him.[59] Lyndalee, unaware of Learson's reputation, charmed him thoroughly as she

---

[59] "I can remember Rasmussen saying, 'Oh, my God, I'm wearing a blue shirt!'" (L.K. Korn, private communication, 1990.) Bob Creasy has described his own encounter with Learson:

> Wham!! The door to my oversized office burst open! Recognizing the large, rude man, I vaulted my fancy desk and immediately introduced him to my life insurance, an M.I.T. student on a work-study program with whom I was conducting an evaluation.
>
> "Mr. Learson, welcome to Cambridge. What can I do for you?", I asked.
>
> T. Vincent Learson, who as a leader in IBM struck fear into many an impure heart, was angry but now contained as he strolled across the carpet and looked out one of the windows at Technology Square. "Nice place you have here", he said as he fixed his eyes on the deep hole being dug for a new building. "Where's Rasmussen?", he asked.
>
> "At lunch.", I replied, knowing where he wanted Norm to be at that moment. Learson then queried me as to why Norm was at lunch. Not knowing what was going on, I simply stated that in Cambridge we ate lunch around noon.
>
> "I'll be back.", he said as he turned abruptly and left the office.
>
> The M.I.T. student and I got back to business. Only a few minutes later I learned that earlier our receptionist had thrown gasoline on the smoldering Learson and pointed him toward my closed office door.

demonstrated CTSS and explained their plans for implementing a similar system on a S/360.[60] Following his visit, Learson arranged for a Model 40 to be diverted to CSC for the programmers to use while awaiting the arrival of their modified Model 40. Rasmussen sold the spare time on that machine, too.

Implementation of CP and CMS was begun in mid-1965, and the design continued to evolve rapidly during the implementation. Much of the early programming was done under CTSS using a S/360 assembler for code generation and a S/360 emulator for testing. Before Learson's Model 40 arrived, the programmers scavenged what time they could on various S/360 machines in Cambridge and Boston and on their modified Model 40 while it was sitting on the factory floor in Poughkeepsie. The CMS developers worked under the very early System/360 operating system, BPS, until they got enough of CMS together so that they could IPL it standalone from a real card reader (using a deck of the BPS loader followed by a deck of the CMS nucleus).[61] Once CMS was reasonably stable running standalone on a real machine, the developers worked under it as much as possible. Later, when CP became runnable, they moved CMS into a virtual machine and continued their development there.[62]

The programming was done by the designers and a few other people. Claude Hans, who came to Cambridge on loan from a customer installation in France, did some of the early work on CMS, including getting enough of the OS simulation together to allow a FORTRAN compiler to run under CMS. His wife Danielle also worked on the system and on the documentation.

Another key participant was a 21-year-old MIT student named Stu Madnick, who began working on CMS in June of 1966. His first project was to continue where Brennan had left off with the file system. Drawing upon his own knowledge of the CTSS and Multics file systems, Stu extended the design of the file system[63] and got it up and running. He continued working part-time during the following school year and added several other important functions to CMS, including the first EXEC processor, which was originally called the COMMAND command. He had written a SNOBOL compiler for S/360, so he got that working under CMS, too. He needed a word processor to use to prepare papers for his courses, so he wrote Script, which was inspired by the CTSS Runoff program.[64] Stu had been told that Dick Bayles (whom everybody

---

[60] CTSS was the only thing they had to demonstrate at that point. Lyndalee remembers that she showed Learson the interactive debugging function of CTSS (which she had written), explaining to him how much more quickly programs could be debugged interactively.

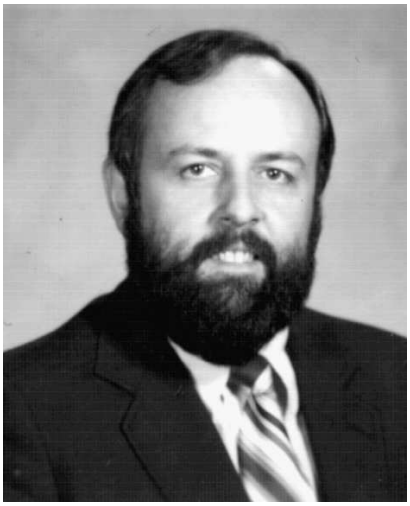
[61] "We went to loading from tape early. The first time I felt CMS was viable was when we had a card reader not working and the developers could keep working with CMS, but OS could not run without the reader." (J.B. Harmon, private communication, 1989.)

[62] "Bob Adair said to me, 'If we can run two virtual machines, we can run *n*.', but we actually had problems running the third one. After that, though, it worked." (J.B. Harmon, private communication, 1990.)

[63] S.E. Madnick, *A Guided Tour Through the CMS File System, Part I—Data Bases*, June 1, 1967.

[64] Madnick started with a program written by Dick Bayles that was very closely modelled on CTSS's Runoff. The early Script system printed a startup message "SEM—Version x". After Script had migrated to several sites, people started asking what "SEM" meant. Although they were told that it meant "Script Environment Module", it was no coincidence that Madnick's

acknowledged to be a brilliant programmer) had written the CMS editor in a week, so he wrote Script in a week. In 1968, he designed a new file system for CMS that anticipated important features of the UNIX file system, but that was never implemented. Stu was to continue working on CMS until 1972, when he finished school and had to get a real job. He is now a professor at MIT.



Claude Hans

Stu Madnick

Many other memorable student employees passed through the Scientific Center in the early years and made valuable contributions to the system. Another of the stars among the students was Nick Negroponte, "a charming and intelligent young man, clearly outstanding," who went on to become co-founder and director of MIT's Media Lab.

CP-40 and CMS were put into production with a regular user schedule in January, 1967. OS/360 had been made to run in a virtual machine by then,[65] and the CSC staff was beginning to recognize and study some of the unpleasant truths about the behavior of virtual memory systems, such as page thrashing.[66]

---

initials are "SEM". Even today, Madnick's name appears in the source for Waterloo Script. The comment "MADNICK IS NARCISSITIC" *[sic]* appears on a PRINT NOGEN statement added by somebody who was tired of seeing Madnick's name displayed in every macro expansion.

[65] "One of the milestones in the CP-40 Project was getting pre-release P (or was it Q?) of OS/360 to boot under CP for the first time—on the shop floor at the Boardman Road laboratories in Poughkeepsie. We discovered after a lot of trial and error that OS/360 violated one of the design principles of System/360—that no operating system software was to be timing dependent. It turned out that a portion of the boot code for OS/360 was timing dependent, and we had to 'kludge' up a two-stage simulation of the S/360 IPL process for it to boot." (R.U. Bayles, private communication, 1989.)

[66] "One afternoon, we weren't getting any response from the 40. Three or four of us went to the

D. 360/67 and TSS

In August, 1965, IBM announced the System/360 Model 67 and TSS, the Time Sharing System.[67] TSS was an elegant and very ambitious system, but the customers who bought the early 67s soon found that TSS had serious stability and performance problems,[68] for it had been snatched from its nest too young. Then, unfortunately, IBM attempted to address the problems in TSS by "throwing bodies" at them, an approach that had already been found to be highly counter-productive in the case of IBM's primary System/360 operating system, OS/MVT.[69]

Machine Room and saw that the disks were going like mad. At first, we didn't understand it at all." (J.B. Harmon, private communication, 1990.) Les Comeau later spent some time studying thrashing (L.W. Comeau, "Operating System/360 Paging Studies", *IBM Storage Hierarchy System Symposium*, December, 1966) and concluded that thrashing had been the basic problem with the Ferranti Atlas machine.

[67] From the August 16, 1965, "blue letters":

> We are pleased to announce that the special bid restrictions have been removed from the System/360 Model 67—a system designed to let many remote users share time on the same high-performance computing facility. With its own powerful operating system, Model 67 provides the user with virtually instantaneous access to and response from the computer and, through multiprocessor configurations, with a high degree of system availability. Time-sharing provides a closer working relationship between the man with the problem and the computing power he needs to solve it.
>
> IBM System/360 Model 67 is supported by a Time Sharing System monitor (TSS) that will take advantage of the unique capabilities of a multiprocessor system. The monitor performs dynamic relocation of problem programs using the dynamic address translation facilities of the 2067 Processing Unit, permitting response, within seconds, to many simultaneous users.

[68] "As a member of Lincoln Lab, I had the assignment of going to Kingston to run TSS before it was delivered to Lincoln. I worked the second and third shift using a 360/67 and ran TSS. It took a long time to IPL and did not stay up very long. In the Fall of 1967, a team of ten IBMers came to Lincoln to help us get TSS running in a production environment. They stayed for about three months before we converted to CP-67/CMS." (J.M. Winett, private communication, 1990.)

[69] F.P. Brooks, *The Mythical Man-Month: Essays on Software Engineering*, Addison-Wesley Publishing Co., Inc., Reading, MA, 1975.

E. CP-67 and CMS

In September, 1966, without having access to a Model 67, the folks in Cambridge began converting CP and CMS to run on the 67. CMS was relatively straightforward to move to the 67, but it was also being enhanced rapidly. The CMS work was done by several people. Claude Hans worked on it for a while but left to return to France in 1967,[70] at which time he was replaced by Tom Rosato. Tom was to be the lead CMS developer for the next several years; it is to him that we owe much of what CMS is today. Another of the CMS developers of that era was a brilliant programmer named John Seymour, whose initials, "JAS", are still to be found here and there throughout CMS today.[71]

CP was more work to move to the 67 than CMS was, because the address translation hardware on the 67 was rather different from that on the 360/40. The CP design was also generalized substantially, to allow a variable number of virtual machines, with larger virtual memories. The first four programmers on CP-67 were Dick Bayles, Dick Meyer, Harit Nanavati, and John Seymour (who worked on CP for a few months before going over to CMS). Bayles managed the CP group and was the primary architect of CP-67.

In CP-40, the control blocks describing the virtual machines had been a hard-coded part of the nucleus. For CP-67, Bayles designed a new control block structure and added the concept of free storage, so that control blocks could be allocated dynamically. The inter-module linkage was also reworked, and the code was made re-entrant.

---

[70] Hans would continue to influence VM, however. He became the leader of a very strong group of VM people at the Grenoble Scientific Center. Rich Kogut, an American who worked for IBM France for ten years and who became a much-esteemed IBM representative to the SEAS VM Project, has described IBM France's contributions to VM:

> The University of Grenoble ran CP-67, which explains the interest of the Grenoble Scientific Center. In addition to Claude's early work, another keystone was the work Alain Auroux did while on assignment to Cambridge. In the mid-70's many people from Claude's group influenced VM. Claude went to Endicott on assignment as a manager and was responsible for convincing IBM to work on and release Xedit and EXEC2. He arranged for assignments for people from his former group: Xavier de Lamberterie, of course; Maurice Bellot, who did a lot of work on VM/BSE R1 CMS; Amine Zhiri who worked on an ill-fated project to combine VM and VS1; Jean-Pierre LeHeiget, who codeveloped the MSS prototype during a short assignment at Yorktown; Pierre Sauvage, who did a lot of the original 370/138 and 370/148 ECPS implementation. I participated in small projects; I remember helping out the Change Team for two weeks when the number of outstanding APARs skyrocketed, in 1976 I believe, and I somehow got hornswoggled into writing the 3340 alternate track support, which is still in the product. All of the early IBM reps to the SEAS VM Project were from IBM France, first Claude Hans, then Jean-Pierre LeHeiget, then myself. (R.M. Kogut, private communication, 1989.)

[71] For example, in DMSSVT:

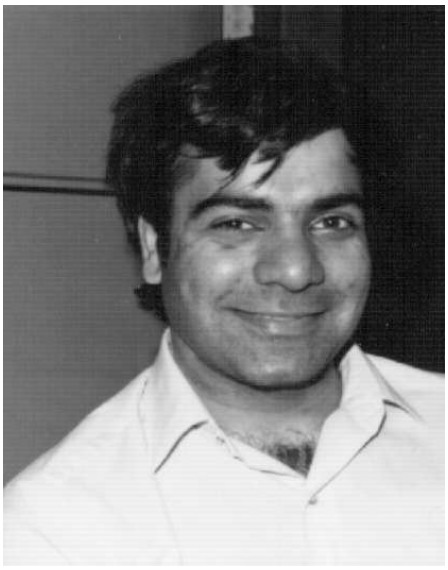
```
SVC     X'CA'              (JAS -- 23 AUGUST 1967)
```

(Note that 202 is said to have been chosen for use as the CMS SVC because in hexadecimal that number forms the first two letters of the word "Cambridge".)

Tom Rosato

John Seymour

Dick Meyer



Harit Nanavati

Because Cambridge didn't yet have a Model 67, the developers had to modify CP-40 to simulate a Model 67, including the address translation hardware and the unique instructions in the Model 67's instruction set. One of these unique instructions was Search List (SLT). Bayles had designed the CP-67 control block structure to take advantage of SLT, so SLT was one of the instructions that CP-40 was modified to simulate. Early in 1967, having gotten a "CP-67" system

together on the Model 40, the developers dumped the system to tape and took it to Yorktown, where they'd been allocated some Saturday test time on a real Model 67. They IPLed the system and watched it immediately flame out with an opcode exception on an SLT instruction. When they told the CE who was standing by that SLT was broken, he replied, "What's an SLT?" It was then that they discovered that the SLT instruction was an RPQ.[72] Soon after that, they began testing CP-67 on the Model 67 at MIT's Lincoln Laboratory, which did have SLT.[73]

Lincoln's was one of the earliest 360/67s, and Lincoln was having severe problems with TSS. It was said to take ten minutes after an IPL to get the first user logged on, but the system's mean time to failure was less than that. So, when Lincoln's computer center manager, Jack Arnow, saw Dick Bayles IPL CP on the Lincoln machine and have all the consoles up in less than a minute, he told IBM that he wanted that system. This demand rocked the whole company,[74] but IBM was so desperate to keep a system at MIT that it would deny Lincoln nothing,[75] so Lincoln was given CP and CMS, which they had in daily operation by April, 1967. There was speculation that the whole affair had been engineered by Norm Rasmussen,[76] who was known to have used various

---

[72] "The 360/67 SLT instruction RPQ was designed at Lincoln by Jack Nolan. He was interested in using it for database list processing. Once it was implemented, IBM found use for it to process lists in the CP nucleus. I don't know if it was ever used by TSS or for any applications program." (J.M. Winett, private communication, 1990.)

[73] The SLT simulation code was moved into CP-67, so that CP-67 would run on machines that didn't have the SLT instruction, but CP-67 customers were advised to order the instruction. Many didn't, however, because a processor could not have both 7090 emulation and the SLT instruction, and many customers needed 7090 emulation. SLT was gradually phased out of the system over the next few years.

[74] "Throughout 1967 and very early 1968, IBM's Systems Development Division, the guys who brought you TSS/360 and OS/360, continued its effort to have CP-67 killed, sometimes with the help of some IBM Research staff. Substantial amounts of Norm Rasmussen's, John Harmon's and my time was spent participating in technical audits which attempted to prove we were leading IBM's customers down the wrong path and that for their (the customers'!) good, all work on CP-67 should be stopped and IBM's support of existing installations withdrawn. Luckily, SDD's own development efforts (on both the TSS and OS fronts) were so behind on function, performance, and schedule that we got away with it. But only because the customers—Lincoln, WSU, University of Alberta, Lockheed, and others—were so vocal and ultimately influential." (R.U. Bayles, private communication, 1989.)

[75] "Lincoln Lab was untouchable." (N.L. Rasmussen, private communication, 1989.)

[76] The early CP and CMS developers were to acknowledge Rasmussen's role repeatedly over the years: "The Center Director, N.L. Rasmussen, is to be congratulated for creating and maintaining an atmosphere conducive to advanced systems research." (J.B. Harmon, *CMS User's Guide*, IBM Cambridge Scientific Center, Cambridge, Mass., March 16, 1967.) "The existence, and success so far, of the Cambridge Scientific Center VMCP, the CP/67 system, is due to the foresight of the Center's manager, Mr. Norman Rasmussen,...". (M.S. Field, *Multi-Access Systems: The Virtual Machine Approach*, IBM Cambridge Scientific Center Report 320-2033, Cambridge, Mass., September, 1968.) "The complete support of N.L. Rasmussen, manager of the Cambridge center, was crucial to the success of the entire project." (Creasy, *op. cit.*, p. 490.) "Norm provided a great environment for us." (R.J. Creasy, private communication, 1990.)

subterfuges to protect his "counter-strategic" CP/CMS project;[77] nevertheless, the Data Processing Division found the money to fund further development of CP-67 to provide temporary relief to Lincoln until TSS could be stabilized.

Ed Hendricks

With the additional funding, the CSC staff grew. Among the new staff members acquired in 1967 were Mike Field, an SE from England who was to do good things to CP, such as adding support for "remote" consoles, so that people didn't have to share the single system console, and Ed Hendricks, who had gained fame at MIT as the author of a really good "SpaceWar" game. Hendricks was to go on to do good work in graphics and in support of guest systems under CP, before becoming co-author of RSCS. (He also became reknowned for developing slick little multi-tasking monitors, each of which was said to be a descendant of his "SpaceWar" game.) In 1968, an almost twenty-year-old MIT student named Dave Tuttle began working at CSC, at first working with Ed Hendricks on various communications projects and on running guest systems, and later becoming a CP developer.

Lincoln had highly skilled system programmers of its own, who began enhancing CP and CMS as soon as they were delivered. The Lincoln and Cambridge people worked together closely and exchanged code on a regular basis. Lincoln programmers, including Frank Belvin, Hal Feinleib, and Bob Jay,[78] made a number of fundamental contributions to CP and CMS, contributions as basic as the GET and PUT commands in the editor.[79]

---

[77] "Norm was told several times to kill CP-67, but he kept it alive under one guise or another. He really wanted to do the right thing for the company from a time-sharing viewpoint." (J.B. Harmon, private communication, 1990.) "You have to have leadership. You have to be able to take risks. Being aware of quality is something they don't teach you in business school." (N.L. Rasmussen, private communication, 1989.)

[78] An IBM Service Bureau employee on assignment to Lincoln to work on TSS. (His surname has since been restored to Jesurum.)

Dave Tuttle

Jim March

One of the programmers at Lincoln during that period was Jim March.[80] (Before I go any further with this story, I must assure you that Jim is a very good programmer, whose contributions include introducing the concept of stacked console input and designing the dual directory scheme of the CMS file system jointly with John Harmon.) One evening in 1968, Jim was working late and found he needed to sort a list of a few hundred items, so he threw together a "quick and dirty" sort/merge program and sorted his list and forgot all about it. Shortly after that, he left Lincoln to go to IDC and didn't have the opportunity to use a "vanilla" CMS system again until 1977, when he moved to Bank of America. The Bank sent him to a GUIDE meeting at which there was much complaining about CMS SORT, so when he got home he printed off a listing and sat down to take a look. To his horror, he immediately recognized that he was the author of the reviled CMS SORT command. He was so embarrassed that he wrote a good CMS SORT and distributed it to all the members of the GUIDE VM Project.

It is important to keep in mind that the CP-40 and early CP-67 work was experimental. The people at Cambridge and Lincoln were simply creating a system for their own use (albeit with the hope that it might later become useful to others). Because of this, the environment in which the

---

[79] Joel Winett's description of one of Lincoln's many CP enhancements, a waiting list for logging on to the system, gives one an idea of the scarcity of computing resources in those days: "CP could support only a fixed number of users, but we had more terminals and users than could be serviced. So we allowed users to be logged on but not given running status. Each user was given a number in a queue (like a bakery number), and when someone logged off the next person was allowed to run. You could query the system to find out how far down in the queue you were. Sometimes you were as high as twentieth in the queue and had to wait hours to be enabled to run. (This was implemented by Rose O'Donnell.)" (J.M. Winett, private communication, 1990.)

[80] Now with V/March.

developers worked was conducive to experimentation, learning, and creativity. Creasy has described it as follows:

> The design of CP/CMS by a small and varied software research and development group for its own use and support was, in retrospect, a very important consideration. It was to provide a system for the new IBM System/360 hardware. It was for experimenting with time-sharing system design. It was not part of a formal product development. Schedules and budgets, plans and performance goals did not have to be met. It drew heavily on past experience. New features were not suggested before old ones were completed or understood. It was not supposed to be all things to all people. We did what we thought was best within reasonable bounds. We also expected to redo the system at least once after we got it going. For most of the group, it was meant to be a learning experience. Efficiency was specifically excluded as a software design goal, although it was always considered. We did not know if the system would be of practical use to us, let alone anyone else. In January, 1965, after starting work on the system, it became apparent from presentations to outside groups that the system would be controversial. This is still true today.[81]

Love Seawright

Bob Seawright

At about the same time that Lincoln decided to run CP-67, another influential customer, Union Carbide, made the same decision.[82] In February, 1967, Union Carbide sent two of its system programmers, Bob Seawright and Bill Newell, to Cambridge to assist in the development of the

---

[81] Creasy, *op. cit.*, p. 487.

[82] This decision is supposed to have been the result of a Union Carbide Vice President living next door to an IBM Vice President, who had told him about CP-40.

system.[83] They both subsequently made important contributions to CP. Union Carbide's IBM SE, Love Seawright, was sent to Cambridge at the same time to learn to support the system. Love tackled the job of documenting the system, figuring out how it worked by using it and reading the listings. As her temporary assignment kept being extended, she worked at documenting, testing, debugging, and giving demonstrations. Later, she would package Version 1 of CP-67 and then help to support it by teaching courses, answering the hotline, and editing the *CP-67 Newsletter*.

Walt Doherty

Stu Toledano

By September, 1967, CP-67 was also running at IBM Research in Yorktown. Stu Toledano tells us that he used to drive to Cambridge from Yorktown every week or two to pick up the latest version of CP-67, which consisted of about 6,000 cards. He would talk with the people to make sure that what he was being given would really assemble and would then drive back to Yorktown. On one rainy trip back, his station wagon developed a leak and the 6,000 cards got warped, so he spent an evening using all the keypunches at Yorktown to duplicate CP and CMS. That was to be the beginning of a very strong CP/CMS tradition at Yorktown, which subsequently produced much excellent function that made its way into the VM product. It also led to the unleashing upon the world of the great Walt Doherty,[84] who was to go forth from Yorktown over and over

[83] Union Carbide also paid for the Model 67 that was later installed at Cambridge.

[84] "[In mid-1965, I] was assigned to be T.J. Watson's man in TSSland at Mohansic. While there, I participated in a number of design meetings and met Lee [Varian], Ted Dolotta, Oliver Selfridge, Jack Arnow, Frank Belvin, and Joel Winett. The last four were at Lincoln Labs. Jack Arnow was Director of Computing there. Frank Belvin and Joel Winett worked for him. Oliver Selfridge was in the Psychology Department. Oliver suggested that I come work with them for a while on an editor project, called the Byte Stream Editor.... I went up to Lincoln for about a year. During that time I came to know Norm Rasmussen and the Cambridge Scientific Center. Sometime early in 1966, Pete Markstein and I visited the Cambridge

again to convince customers' managers and IBM's of the economic value of sub-second response time.

Lynn Wheeler

Dick Newson

In January, 1968, CP and CMS were installed on another Model 67, at Washington State University (WSU). At that time, WSU's primary system programmer was an undergraduate named Lynn Wheeler. Wheeler made numerous enhancements to CP and CMS while at WSU, particularly in the area of performance. In fact, he managed to reduce the CP overhead of running MFT under CP-67 by eighty percent, by such means as reducing path lengths, adding fast paths, replacing SVC linkages between CP modules with BALR linkages, improving the page replacement algorithm, and making CP modules pageable. After he graduated in February, 1970, Wheeler drove through snow and sleet straight to Cambridge, where he began a career that

---

Scientific Center to see the projects that were going on there at that time. CP-40 was one of those projects.... Peter suggested to Norm that he redo CP-40 for the Model 67 since it was becoming clear to us that an alternative to TSS might be required. But also, it was clear that the virtual machine concept was a very interesting concept for experimentation. Peter also offered my help to Norm to do that. Norm turned us down, saying that he wanted to build up the reputation of the Cambridge Scientific Center and wanted to do it himself.... I doubt if we were the first ones to make that suggestion to Norm, but it didn't hurt.... When my time at Lincoln was up, I returned to Mohansic and became more convinced than ever that TSS wouldn't work well, and certainly not on schedule.... Just before the August, 1967, SHARE it became obvious to me that TSS would never be viable in anything like its promised schedule, so I was asked by Bill Dorn (our Director of Computing at T.J.'s Place) and by Peter Markstein to bring CP/CMS from Cambridge down to Yorktown. Stu Toledano was assigned to this as well. We made many late night trips to Cambridge.... Mike Field and Harit Nanavati were very helpful to Stu and me.... So, in the Fall of 1967, CP/CMS was running on the Model 67 at Yorktown for about a hundred people. It was doing quite well." (W.J. Doherty, private communication, 1990.)

proved to be enormously beneficial to VM.[85]

By the time SHARE XXX convened in February, 1968, there were eighteen 360/67s installed.[86] Most of these machines were running TSS (or trying to). SHARE's TSS Project had great *esprit de corps*, much as the VM Project later would. It was working very closely with IBM to improve TSS and was devoting tremendous energy to supporting the system.[87] On Monday of that SHARE week, IBM released a "blue letter" announcing the decommittal of TSS.

On Wednesday of that week (February 28), Dick Bayles and Harit Nanavati made a presentation on CP-67 and CMS to the TSS Project. The CP/CMS session was well attended and generated some interest, but most of the Project's energy that week was devoted to protesting the decommittal, which had come along just when it appeared that TSS was finally beginning to achieve acceptable levels of stability and performance. One result of the CP/CMS presentation was that an IBM SE named Dick Newson, who had himself given a TSS presentation that week, stopped by Cambridge on his way home to Alberta, and became hooked on CP. Newson made his way back to Cambridge in February, 1969, where he began a very distinguished career as a CP developer.

Version 1 of CP-67 was released to eight installations in May, 1968, and became available as a TYPE III Program[88] in June. Almost immediately after that, two "spinoff" companies were

---

[85] Wheeler moved to AIX Development in August, 1987.

[86] *Proceedings of SHARE XXX*, 1968, p. 6-292.

[87] The TSS users may have been the most thoroughly organized group of customers that IBM has ever had to deal with. The following is from a November, 1966, letter from Oliver Selfridge, of Lincoln Lab, to other TSS users proposing a formal collaboration between TSS sites (which was subsequently implemented, except for the networking):

> This memo is concerned with the development of a collaborating community of TSS/67 systems. We propose that each organization possessing such a machine resolve to follow these guidelines in building and using software systems. We propose that the 67s form a *de facto* net for testing and exchanging software systems on-line. We are sure that a little planning and a fair amount of cooperation can lead to a vast increase in the power of our software and hardware, and thus in our own powers and capabilities.
>
> Each organization should coordinate its plans for software systems so as to avoid duplication. Some clearinghouse should be set up to communicate such plans and to aid coordination. It is strongly urged that coordination be started too early rather than too late. It will be found that doing so will not hinder experimentation but will in fact tend to make it more useful in the long run.

[88] "CP-67/CMS is a Type III (IBM employee contributed) Program, and has not been submitted to any formal test. Type III Programs are provided by the IBM Corporation as part of its service to customers, but recipients are expected to make the final evaluation as to the usefulness of the programs in their own environment. There is no committed maintenance for Type III Programs, nor does IBM make any warranty, expressed or implied, as to the documentation, function or performance of such programs." (*An Introduction to CP-67/CMS*, IBM Cambridge Scientific Center Report 320-2032, May, 1969.)

formed by former employees of Lincoln Lab, Union Carbide, and the Cambridge Scientific Center, to provide commercial services based on CP/CMS. Dick Bayles, Mike Field, Hal Feinleib, and Bob Jay went to the company that became National CSS.[89] Harit Nanavati, Bob Seawright, Jack Arnow, Frank Belvin, and Jim March went to IDC (Interactive Data Corporation). Although the loss of so many talented people was a blow, the CSC people felt that the success of the two new companies greatly increased the credibility of CP-67.[90] Rip Parmelee became the manager of CP development when Bayles left.

Rip Parmelee

Joel Winett

The TSS decommittal was rescinded in April, 1969, as the result of vigorous protests by customers, but by then some TSS sites had switched to CP/CMS. SHARE had formed a CP/CMS Committee under the leadership of Joel Winett, of Lincoln Lab,[91] and CP-67 was running at fifteen accounts. VM had begun what later came to be known as its "Doubtful Decade".

---

[89] One of the founders of National CSS was Dick Orenstein, one of the authors of CTSS.

[90] "While it may sound self-serving, I believe IBM's belated decision to announce relocation hardware on the S/370 series was influenced in some part by the commercial success of NCSS and IDC. By the time the first relocating S/370 (the 370/148) was shipped, these two companies had about 50% of IBM's installed base of 360/67 systems—11 systems at NCSS in three data centers on two coasts and several at IDC." (R.U. Bayles, private communication, 1989.)

[91] The SHARE and GUIDE CP/CMS Committees were formed at the same time, at the joint SHARE/GUIDE meeting in Atlantic City in the Fall of 1968. Love Seawright helped establish these committees and became the chief IBM liaison to them.

Bruce Marshall

Liz Levey

Dick Meyer assumed responsibility for both CP and CMS Development in 1969; he was to remain in that position until he, too, left to go to IDC in 1974. Version 2 of CP-67 was released in June of 1969. The authors of Version 2 were Diane Boyd, Clark Frazier, Liz Levey, Stu Madnick, Dick Meyer, Dick Newson, Tom Rosato, Love Seawright, and John Seymour.[92] Version 2 included PL/I support and a new scheduler developed by Bob Jay at Lincoln Lab, which was a real improvement over the Version 1 scheduler, which had no means of controlling page thrashing.[93]

CP at this stage was still quite primitive. For example, it didn't have dynamic page slot allocation. When a user logged on, space for his entire virtual machine (typically 256K) was allocated on the paging drum, if there was room, or on disk if the drum was full. Those slots were then his until he logged off. This had the effect of causing users to get to work earlier and

---

[92] The *Program Contribution Form* that came with Version 2 listed the authors' names as: D.R. Boyd, H.C. Frazier, Jr., E.H. Levey, S.E. Madnick, R.A. Meyer, S.R. Newson, T.D. Rosato, L.H. Seawright, and J.A. Seymour. And it contained the following acknowledgment:

> The CP-67 authors wish to acknowledge the contributions made to the system by R.J. Adair, R.U. Bayles, L.W. Comeau, R.J. Creasy, and J.B. Harmon, who developed the basic virtual machine concept; R.P. Parmelee, who formerly managed the CP-67 effort; M.S. Field, J.A. Kelch, and H.M. Nanavati, formerly with the CP-67/CMS project; E.C. Hendricks, C.I. Johnson, and D. Tuttle for their work in the operation of OS/360 in a virtual machine; as well as several users of the system, in particular the staffs of the MIT Lincoln Laboratory and Washington State University.

[93] L.H. Wheeler, "VM Performance History", *Proceedings of SEAS AM86*, October, 1986, pp. 215-224.

earlier, in order to acquire drum page slots. Bruce Marshall[94] was one of the earliest members of the SHARE CP-67 Project and remembers vividly how good it felt when he succeeded in modifying CP to do dynamic page slot allocation.

Version 3 of CP-67, the first version with Class A support, was released in November, 1970, to be followed by the last two releases, 3.1 and 3.2, in 1971 and 1972.[95] Version 3 had several important performance enhancements, including free storage subpool support. Under Version 2, systems that didn't have the SLT instruction spent as much as 20 percent of their CPU on free storage management; Version 3 reduced that by a factor of 10.

Release 3.1 contained the first "Wheeler Scheduler" to be available to customers. It had feedback controls, "fair-share" type priority calculations, and dynamically adjusted working set size predictions.

One important change the CMS developers made in Release 3.1 was to replace the SIO instructions in CMS with high-performance DIAGNOSE instructions, so that CP could provide CMS with fast-path I/O. This improved performance dramatically, but it meant that CMS would no longer run on the bare hardware.[96] For better or for worse, the interface between CP and CMS was no longer defined strictly by the *Principles of Operations*.

CP-67 ultimately ran on 44 processors, about one-fourth of which were internal.[97] By the end of the CP-67 era, much had been learned about making a virtual machine operating system perform well; Version 3.1 supported sixty CMS users on a Model 67.

---

[94] Then at Perkin-Elmer, now with ReadiWare Systems.

[95] Release 3.2 contained no new function; it was simply a maintenance release.

[96] Another reason for removing CMS's standalone capability was to avoid the requirement for adding hardware diagnostic support to CMS for Field Engineering.

[97] The Cambridge Scientific Center would continue to innovate in CP, CMS, and other components of VM until IBM closed it on July 31, 1992. That date would also mark the retirement from IBM of Bob Creasy, Noah Mendelsohn, Dick Newson, Love Seawright, Tom Rosato, and Lynn Wheeler.

F. VM/370

On June 30, 1970, IBM announced System/370, *again* without address translation hardware, which was very discouraging to both TSS and CP/CMS customers.[98]

In May, 1971, IBM held a meeting of its TSS customers at the Westchester Country Club. At that meeting, IBM permanently decommitted TSS (although a S/370 version was subsequently made available to customers who already had the S/360 version). IBM also revealed its future hardware and software plans, including relocate on the entire S/370 line; two new systems that would use relocate (these became OS/VS1 and OS/VS2); and, in the longer term, an architectural extension to provide 31-bit addressing.[99]

At the Westchester meeting, IBM also said that it hoped that customers would *not* need virtual machines. Despite that hope, development of a S/370 version of CP and CMS had begun in the Summer of 1970, in Cambridge. Since nobody in IBM wanted to fund the development of a S/370 version of CP/CMS, an unorthodox approach to getting funding was required. As it happened, Field Engineering had just decided that it had too many Software Field Engineers,[100] and it was trying to find new IBM positions for one thousand of them. It was willing to pay the first two years of salary for each one who was given a new position, but it didn't want to have to pay for their relocation. The result of this was that CP/CMS Development got thirteen of the sharpest software FEs in the Boston area and didn't have to pay their salaries for two years.[101] This doubled the size of the development group.[102] It also made some Boston customers very unhappy. The President of State Street Bank is said to have phoned the President of IBM to complain that they'd lost three of their four software FEs, all of whom they had trained themselves.

---

[98] Rumors abound that up until the last week before the original System/370 announcement, IBM had planned to announce relocation hardware. In fact, at least some S/370 Model 135s were shipped with "DAT" marked under the appropriate lamp in the PSW display.

"With respect to the availability of relocation on the System/370, my personal view is that the TSS debacle coupled with some OS/360 development team management who believed TSO could provide OS-compatible interactive data processing without virtual memory led to a long internal debate. In the end, I think it was CP's performance advantages over TSO coupled with successful non-IBM enhancements to CP that made the (former) DP Division actively press the development people to accede to relocation and VM. It was never an issue of the hardware not having the capability—there were hints of relocation in the 145 as well—but only the internal 'we want one operating system' debate that delayed it." (R.U. Bayles, private communication, 1989.)

[99] TSS/360 had 32-bit addressing, although only 31 bits of address could be used, because of the signed arithmetic used by the BXH and BXLE instructions.

[100] Later known as PSRs.

[101] "The new people all arrived for training one day in April or May of 1971. Many of them were really thrilled at getting in on the ground floor of a new operating system." (R.A. Meyer, private communication, 1989.)

[102] "Without that funding, VM/370 wouldn't have happened." (R.A. Meyer, private communication, 1989.)

CP/CMS Development had been split out of the Scientific Center for legal reasons following the "unbundling" announcement on June 23, 1969. However, the developers were still in the Tech Square building and were still working closely with people in the Scientific Center, including Bob Adair, Rip Parmelee,[103] Charlie Salisbury, and Alain Auroux, who was at Cambridge on loan from the Grenoble Scientific Center.

By the time VM/370 Release 1 was finished, the CP/CMS Development group, including the documenters, had grown to 110 people. These were the primary developers for VM/370 Release 1:[104]

CP Developers

Dick Newson, *manager*
Charlie Brackett
Larry Estelle
Ray Grein
Tom Heald
Ed Murray
John Seymour
Dave Thibodeau
Dave Tuttle
Charlie Weagle
Clyde Wildes
Carl Young

CMS Developers

Tom Rosato, *manager*
Fernando Arce
Bob Collins
Bob Downs
Paul Fay
Sharon Koblinsky
Alan Middendorf
Dick Milley
Jim Sullivan
Jim Walsh
Bobbie Wilson
Walt Wisnowski
John Xenakis

Bobbie Wilson John Xenakis
Alan Middendorf

[103] In speaking of his days in Cambridge as a VM developer, Paul Tardif has said, "Rip was one of these magnificent people. He and Bob Adair were upstairs. Whenever we had a difficult problem, we would go upstairs and talk to Adair and Parmelee and get our thinking straightened out." (P. Tardif, private communication, 1989.)

[104] From an unpublished address by S.R. Newson at the "VM Fifteenth Birthday Party", SHARE 69, August, 1987.

Dave Thibodeau

Tom Heald

Ray Grein

Paul Fay

(Xenakis was the author of the COPYFILE "compiler" among other things and was known as "Captain Midnight". Tom Rosato began calling him that because Xenakis was in the habit of working all night and then leaving lists of the new features in CMS taped to his colleagues' doors where they'd find them when they got to work in the morning.)

While the CP developers concentrated on re-structuring CP, the Scientific Center worked on bootstrapping CP-67 onto a System/370. One of the products of this work was a version of CP-67 that would create System/370 virtual machines on a Model 67. This system became critical to the people who were writing MVS, who were way behind schedule, in part because they had so few prototype 370s on which to test.[105] It may be fair to say that by saving the MVS developers, VM saved itself. Once MVS Development became dependent upon VM to virtualize each new level of the architecture, VM became much harder to kill.[106]

---

[105] An IBM newsletter announced the awards given for the virtualization of System/370 on the 360/67 ("Cambridge Men Modified CP-67, Providing Tool for Developers", *IBM News*, vol. 9, no. 15, August, 1972, p. 1.):

> CAMBRIDGE, MASS.: The work of four men at the Scientific Center here begun almost two years ago had an important role in the development of the operating systems announced this month. The four have received Outstanding Contribution Awards for their work. Two of the men, Dr. Richard Parmelee and Alain Auroux, are now with IBM France in Paris and Grenoble. Charles Salisbury and Robert Adair remain with the Scientific Center staff.
>
> The four extended and enhanced CP-67, the control program that provided virtual machine capability on a Model 67. Their enhancement of the program gave IBM developers access to virtual 370s on a Model 67 as well as a version of CP-67 which would run on System/370.
>
> The central modifications to the program made by the IBMers were: support of the new dynamic address translation facility, additional control registers, and some of the new 370 instructions and features. It became a tool for many IBMers writing the new virtual machine and virtual memory operating systems. Their extensions provided a means of testing 370 programs on Model 67 hardware even before 370 hardware was available internally.
>
> "Moving CP-67 from a 360 base to a 370 base meant that the CP/CMS Development Group working on VM/370 had something to start running on their 145", explains Dr. William Timlake, Scientific Center manager. "The people in SDD writing VS code also had something to use in developing virtual memory software." In mid-1970, an SDD advanced system programming team headed by Russ Hamrick asked the four to assist in developing a virtual machine system for the 145. "We obtained the architectural specifications of the 370 advanced functions and together developed the initial code", notes Adair.
>
> Two prototype systems resulted: one to provide virtual 370s on the 67 (required for testing, since the advanced hardware was not available), the other to provide virtual machines on a System/370. A critical test of the modified program came when Auroux travelled to Endicott to try the program on a prototype Model 145. "That was one of high points", recalls Salisbury. "Auroux loaded the disk pack onto the engineering model, a computer that had never run an advanced function operating system, and the system ran successfully. It demonstrated that software could be developed on a virtual machine for hardware not yet produced." After that, use of the modified CP-67 programs spread to several locations where development activity was underway.

Alain Auroux

Charlie Salisbury

Alain Auroux did most of the actual coding and testing for the bootstrapping, but Rip Parmelee, Bob Adair, and Charlie Salisbury were also heavily involved in working out the design. When Auroux started, Cambridge was running a 360/67, not a S/370, and that 67 was a production system, so he had to avoid destabilizing it. "Vanilla" CP-67 systems created System/360 virtual machines, but they did not virtualize the 360/67; that is, they did not allow a guest to create its own virtual storage. Auroux's first step was to modify CP-67 to create virtual 360/67s, which used 4K pages and 1M segments. Once he had convinced the Cambridge Operations Manager to run that as the production system, he could then proceed to develop a CP-67 that virtualized the System/370 architecture.

The System/370 relocation architecture was different from the 360/67 architecture; it allowed both 2K and 4K pages and both 64K and 1M segments. So, Auroux modified his modified CP-67 to support 64K segments and the new System/370 instructions. He ran that system second-level, so he could run a virtual S/370 third-level. He developed a prototype "CP-370" in that third-level virtual machine. Then, to test this CP-370's virtualization of System/370 virtual memory, he had to run it both third- and fourth-level, with a couple of CMS machines running fifth-level. He remembers doing much of his work from home at night using an "old, slow, noisy teletype".[107] His prototype CP-370 had been debugged in simulation by the end of 1970. Late in January,

---

[106] There is a widely believed (but possibly apocryphal) story that anti-VM, pro-MVS forces at one point nearly succeeded in convincing the company to kill VM, but the President of IBM, upon learning how heavily the MVS developers depended upon VM, said simply, "If it's good enough for you, it's good enough for the customers."

[107] A. Auroux, private communication, 1989.

1971, just before Auroux was to return to France,[108] he and Bob Adair and Rip Parmelee took a copy of his system to Endicott so that they could test it on a prototype 370/145 with relocation hardware. It IPLed the first time.

A real S/370 (a 145 with relocate hardware) was finally delivered to Tech Square during the Fall of 1971 (amidst frantic security precautions for fear that nosy neighbors would figure out that a S/370 being delivered to CSC *must* have relocate hardware).[109] By then, the CP developers had incorporated Auroux's work into their own enhanced version of CP and were at last able to run their system first-level.[110] They first IPLed a full-function VM/370 CP one day in February, 1972. They packaged an alpha-test version for internal distribution on July 5, 1972. Thus, working at break-neck speed, the small group in Cambridge managed to get their system ready for IBM's spectacular System/370 Advanced Function announcements on August 2, 1972. On that day, IBM announced:

- Two new computers, the 370/158 and the 370/168;
- Address relocation hardware on all 370s; and
- Four new operating systems:[111]
  — VM/370,
  — DOS/VS, a virtual storage version of DOS,
  — OS/VS1, a virtual storage version of MFT, and
  — OS/VS2, a virtual storage version of MVT.

On announcement day, VM/370 was up and running for demonstration purposes at all of IBM's Field Support Centers, unlike VS2.

At an announcement session at SHARE XXXIX the following week, IBM listed the advantages of virtual storage as:

- Elimination of artificial memory constraints;
- Enhanced processor storage utilization; and
- Enhanced machine accessibility for application program development and maintenance.

---

[108] Auroux was on assignment at CSC from July, 1969, to February, 1971. He has described his first few minutes at CSC as follows: "I remember my first day of assignment: when I arrived at the Center, Norm Rasmussen, who was at the time the Center Manager, brought me in a dark meeting room, with a TV set in it, and told me: 'Sit down, we will discuss later'. I did so, and one minute later I watched on TV Apollo XI leaving for the first trip from Earth to the Moon!" (A. Auroux, private communication, 1989.)

[109] "Cambridge was not a good place for developing unannounced products. I can remember one evening at a bar in Cambridge having an MIT student tell me that one of his biggest objectives was to find out what IBM was doing with that 145 in 545." (R.A. Meyer, private communication, 1989.)

[110] At that point, the virtual memory architecture was still changing, and they had to track the changes in the instructions, which changed every couple of months when they got a new microcode load.

[111] OS/VS2 would ultimately be made available in two versions, which would be known as SVS and MVS. VM/370 would finally be withdrawn from marketing on April 24, 1989.

Particular emphasis was placed on the productivity gains IBM itself had achieved by doing OS maintenance and testing in virtual machines under CP-67 and VM/370.[112]

This is the text that went with the first few slides in the VM/370 announcement package:

> To help you take advantage of the real and virtual storage capabilities of System/370, we are going to give you a presentation today on a new IBM product, Virtual Machine Facility/370, or VM/370.
>
> Here is a prism. Consider for a moment what happens when a beam of light falls upon it...many colors evolve from one light source. Let me emphasize that point...many from one.
>
> By way of analogy, think of the beam of light as an IBM System/370; the prism, as Virtual Machine Facility/370. The many colors produced by the prism from the one light source are now many virtual 370s produced by VM/370 from one real 370. And each virtual 370 has the capability to run its own programming system, such as OS, DOS, or CMS. Many from one...many virtual 370s from one real 370. And VM/370 makes it happen![113]

Further on in this "pitch", IBM cited keeping your system programmers happy as being one of the big advantages of VM.

VM/370 was announced with two components, CP, the "Control Program", and CMS, which was now to be called the "*Conversational* Monitor System". VM/370 was shipped to the first customers at the end of November, 1972.

The design point for VM/370 Release 1 was a 512K 370/145. The largest machine that IBM had announced at that time was an 8M 168. The marketing forecasts for VM/370 predicted that no more than one 168 would ever run VM during the entire life of the product. In fact, the first 168 delivered to a customer ran only CP and CMS. Ten years later, ten percent of the large processors being shipped from Poughkeepsie would be destined to run VM, as would a very substantial portion of the mid-range machines that were built in Endicott. Before fifteen years had passed, there would be more VM licenses than MVS licenses.

Thirty-two CP-67 customers migrated to VM/370 Release 1. All of them blessed the CMS developers for having made the CMS/370 file system upward compatible. Old modules wouldn't work, however, because the user area had been moved from X'12000' to X'20000'.[114]

---

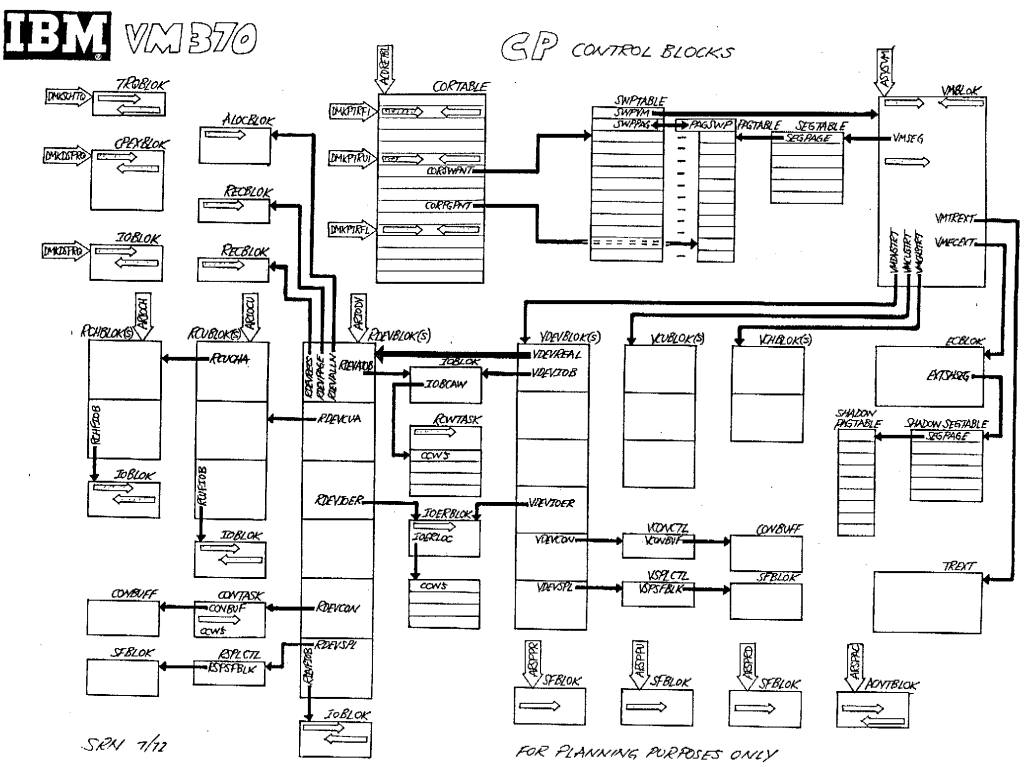
[112] J.P. Hogan, "Value of Virtual Storage", *Proceedings of SHARE XXXIX*, August, 1972, pp. 301-329.

[113] *IBM Virtual Machine Facility/370 (VM/370) Demonstration*, GV20-0388, IBM Corp., 1972.

[114] Another change to CMS in VM/370 Release 1 was the removal of the old "virtual RPQ device", the pseudo-chronolog at virtual address 0FF, which was replaced by a DIAGNOSE instruction. Under CP-67, I/O to device 0FF returned a buffer containing the date, time of day, and the CPU utilization for the virtual machine. CMS tested for the existence of 0FF to decide whether it was running standalone. (I am told that at one stage when one IPLed CMS standalone or after having detached 0FF, it put out a message that said, "Aren't you embarrassed to be running CMS on a bare machine?")

An important change to CMS in Release 1 was the introduction of the multi-level update capability in the UPDATE command. This made possible the VM source maintenance procedures we were to use and love for the next fifteen years. Multi-level UPDATE was the joint work of Dave Tuttle, from CP Development, and John Xenakis, from CMS Development.

Unlike CMS, CP was largely restructured for VM/370. Dick Newson designed the new control block structure. Just before Release 1 was announced, Dick drew this diagram of the CP control block logic:



CP Control Blocks, July, 1972 [115]

(which makes an interesting contrast to the MVS control block flow of about the same era). To go with the new control block structure, Newson and Carl Young[116] designed new register usage conventions, command scanning routines, and module linkage macros.

---

[115] "VM—10 Years—A Retrospective", *Proceedings of SHARE 59*, August, 1982, p. 1874.

[116] "Carl Young was responsible for writing vast portions of the code. He owned a third of the major components of the system." (P. Tardif, private communication, 1990.)

The most important new CP function in Release 1 of VM/370 was the ability to run VM under VM. Alain Auroux's virtualization of the 360/67 was never made available to customers, although customers, such as Bruce Marshall, had made the same modification even earlier than Auroux. In VM/370 Release 1, however, it was official. VM system programmers could test VM under VM. They could now go years at a time without having to take standalone test time early on weekend mornings. This became an important factor in instilling the passionate love that was to keep the VM community struggling to save VM through the coming dark years.[117]

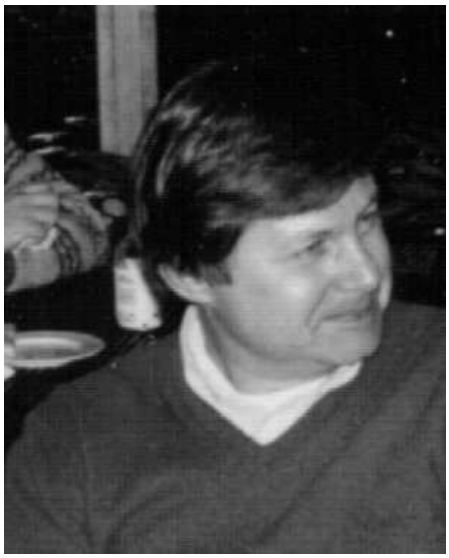
---

[117] In 1973, the ACM held a conference on virtual systems. Robert Goldberg (who had also virtualized the 360/67 and who later became one of the founders of BGS Systems) described the reasons for the increasing interest in virtual systems:

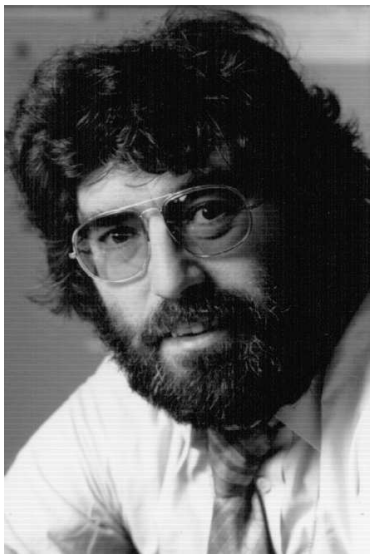
> The development of interest in virtual computer systems can be traced to a number of causes. First there has been a gradual understanding by the technical community of certain limitations inherent in conventional time-shared multi-programming operating systems. While these systems have proved valuable and quite flexible for most ordinary programming activities, they have been totally inadequate for system programming tasks. Virtual machine systems have been developed to extend the benefits of modern operating system environments to system programmers. This has greatly expedited operating system debugging and has also simplified the transporting of system software. Because of the complexity of evolving systems, this is destined to be an even more significant benefit in the future.
>
> As a second point, a number of independent researchers have begun to propose architectures that are designed to directly support virtual machines, *i.e.* virtualizable architectures. These architectures trace their origins to an accumulated body of experience with earlier virtual machines, plus a set of principles taken from other areas of operating system analysis. They also depend upon a number of technical developments, such as the availability of low-cost associative memories and very large control stores, which now make proposals of innovative architectures feasible.
>
> A third reason for the widespread current interest in virtual machines stems from its proposed use in attacking some important new problems and applications such as software reliability and system privacy/security. A final point is that IBM has recently announced the availability of VM/370 as a fully supported software product on System/370. With this action, IBM has officially endorsed the virtual machine concept and transformed what had been regarded as an academic curiosity into a major commercial product.

(R.P. Goldberg, *Proceedings of ACM SIGARCH-SIGOPS Workshop on Virtual Computer Systems*, March, 1973, pp. *ii-iii.)*

The ability to run a system under itself has remained relatively rare, despite the enormous benefits in increased system availability. IBM provided the "PolyASP" cabability for the MVT subsystem ASP, but did not carry that capability forward into ASP's successor, JES3. Late in the life of TSS, there was much discussion of adding virtual machines to TSS, but only for running OS, not for running TSS itself. From a system programmer's point of view (at least, from this system programmer's point of view), UNIX's most glaring weakness is in not being virtualized.



Carl Young



Dick Newson

The group developing VM/370 Release 1 produced at least a thousand lines of code per person per month, to the astonishment of IBM's auditors, and they managed to enjoy themselves while they were doing it. To get even with Dick Jensen, who was doing "systems assurance" and was hassling them about their flow charts, they added "junk code" to the system to see if he would find it. He did find and remove the code that wished the various developers a happy birthday whenever a system was IPLed on one of their birthdays. He did not delete the code that typed "BONG BONG BONG BONG" across all the terminals at midnight. Dick Newson claims that his group also considered making the DIAL command reply, "Aren't you glad you use DIAL?", but they were afraid that would result in legal hassles.[118] While having their fun, they developed a good, clean system, with a tight, simple structure.

In the CP-67 days, if a customer had a problem, he simply phoned Cambridge and talked to the person responsible for that part of the system. Customers could also dial into the Cambridge system to download the current fixes. But VM/370 was supported by Field Engineering, so things were now more formal. CP and CMS Level II for the U.S. for VM/370 Release 1 was Lyn Hadley.[119] Lyn was later to spend a good deal of time struggling to get debugging tools into the

---

[118] "DIAL" is the name of an American deodorant soap which has long been advertised with the slogan, "Aren't you glad you use DIAL? Don't you wish everyone did?".

[119] He and Terry Gibson, of IBM Canada, had been sent to Cambridge before the announcement to learn the system by reading dumps for the developers. Lyn tells us that one day he was given a really awful dump in which page 0 had been overlaid. He couldn't figure out how he was going to shoot this dump; he had found the trace table without benefit of the overlaid trace table pointers, but had no idea of how to determine the current trace table entry. So, he took the dump to Carl Young, who glanced through the trace table and then pointed out the current entry. When Lyn asked him how he had figured that out, Carl pointed to the next entry and said that that one couldn't logically follow the one before it; therefore, it must be

product; the release of the IPCS component was due largely to him. He is now the CP Change Team's ombudsman.

Lyn Hadley

Peter Callaway

Eight days after VM/370 was announced, the SHARE VM/370 Project had its first meeting, replacing the CP-67 Project. Six months later, the membership of the Project had tripled, and ten SHARE installations already had VM/370 in production.

In 1971, as the Technology Square building became more and more crowded, portions of VM Development began moving to the New England Programming Center in Burlington, just outside Boston. The move was completed by January of 1973. The developers thrived in Burlington and delivered new function at an astonishing rate, *via* the service tapes, which were known as "PLC tapes".[120] Some PLC tapes amounted to new releases. Release 1 PLC 09, for example, added CMS BATCH, console spooling, support for the 370/168 and the 2305 drum, and Carl Young's "biased scheduler", which replaced the primitive scheduler in the Release 1 base.

VM/370 Release 2 came out in April, 1974, and included support for several new devices, as well as the new Virtual Machine Assist (VMA) microcode. Release 2 also included support for 3270s—the now familiar "MORE" and "HOLDING" and "NOT ACCEPTED"—which was the work of Dick Newson, who had wandered around MIT watching how people worked with tubes and trying to figure out the best human factors. (He regrets not having thought of a RETRIEVE key.)

Release 2 PLC 05 added native support for the 3705, a project of Dave Tuttle's. Release 2 PLC 11, in January, 1975, added the new component RSCS. Two months later, PLC 13 added VS1 handshaking and the CP monitor, which was the work of Peter Callaway. In February,

---

the oldest entry. Lyn never did find out whether Carl was putting him on.

[120] "PLC" stood for "Program Level Change".

1976, PLC 23 added another new component, IPCS. That same month, Release 3 became available, including VMCF and support for 3350s and the new ECPS microcode.

Edgar (the "Display Editing System"), a program product full-screen editor written by Bob Carroll, also came out in 1976. Edgar was the first full-screen editor IBM made available to customers, although customers had previously written and distributed full-screen editors themselves, and Lynn Wheeler and Ed Hendricks had both written full-screen editors for 2250s under CMS-67.

G. Supporting the new VM users

By this time, VM was spreading rapidly, and education of new VM system programmers was becoming a problem. One important tool that helped the new VMers get "up to speed" rapidly was the excellent *Virtual Machine Facility/370 Features Supplement*, which was released in January of 1974, and was, in my view, the best manual IBM ever published. The author was Barbara McCullough.

Barbara McCullough

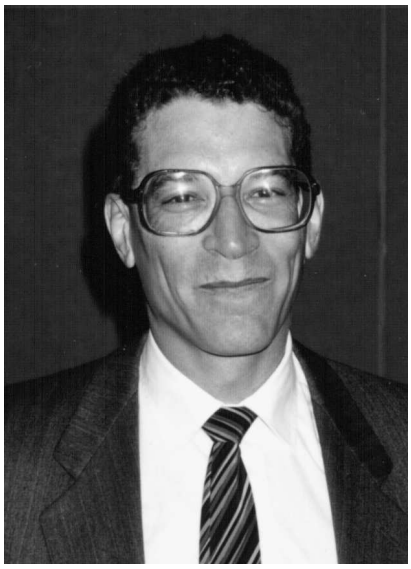
During this period, one noticed geographic pockets of VM activity that corresponded to the presence of a believer inside IBM who was doing his or her best to promote and support VM. Many of these heros and heroines should be cited here, but I can mention only a few of them. One especially influential person was Canada's "Mr. VM", Paul Tardif, an SE who had been a VM developer. Claude Hans and others who had been at the Grenoble Scientific Center were a strong pro-VM force in Europe. After IBM closed the New England Programming Center in Burlington, many of the VM developers became SEs, some remaining in the Boston area, others moving elsewhere, and all of them spreading the word about VM. Love Seawright continued to be the primary liaison to the user groups and to play an important role in conveying customer concerns back to the development group. GUIDE rewarded her efforts with a button in her honor.

"It is absolutely untrue that CMS only does it for Love"

Paul Tardif

Despite the efforts of the heroes and heroines inside IBM, much of the support for new installations had to come from the user community. Most IBM branches were openly hostile to VM, and many used extreme measures to discourage customer managements from installing VM. Few branches provided good VM support, so new users were often in trouble.

Sandra Ward



Romney White

Existing VM installations were conscious of TSS's fate and knew that VM would die unless it grew, so they built the necessary support structure:

- ***The VM Library:*** In 1973, the great guru Romney White[121] volunteered the University of Waterloo to become the home of the VM Library, more familiarly known as "the Waterloo Tape". Sandra Ward graciously took on the job of supporting this project and was able to report by the end of its first year that the Library had grown to four reels of tape. These tapes of tools, modifications, utilities, cookbooks, etc., were contributed by VM people around the world and became vital to installations getting started in VM.

- ***Regional VM Users Groups:*** In December, 1973, Simcha Druck founded the Metropolitan VM Users Association ("MVMUA") in New York. MVMUA is still going strong and recently celebrated its fifteenth anniversary. Other local users groups soon sprang up around the world, and many have been extremely active in supporting their members and new installations. The UK VM Users Group, whose slogan is *"VM System Programmers do it Virtually All the Time"*, was founded in February, 1974; celebrated its tenth birthday in 1984; and is still very active. In much of Europe, IBM marketing people were even more anti-VM than they were in North America, so the local user groups were even more essential there. *Le club Francais des utilisateurs VM* was founded by Frédéric Roux in 1977, with help and encouragement from Claude Hans.

Frédéric Roux


Simcha Druck

- ***VM Workshops:*** In 1977, Romney White came up with another great idea—that the University of Waterloo should hold the first of the annual VM Workshops. The Workshops allow VM people from all over to get together to spend a few days attending technical presentations and discussing their installations' problems. Since Waterloo started the tradition, many other universities have hosted the Workshop. Many people have devoted their time to making the Workshops work, notably John O'Laughlin, Simcha Druck, and Harry Williams. The next VM Workshop will be at the University of Kentucky, June 4-7, 1991.

---

[121] Now with Velocity Software.

Harry Williams

John O'Laughlin

- ***The SHARE VM New Users Project:*** In 1977, SHARE formed a project specifically to support new VM installations. In the years since then, many people have worked very hard to make that project a success, perhaps the most notable of them being "the fastest wit in the West", JoAnn Malina, and her successor as the New Users Project Manager, Roger Deschner.

JoAnn Malina

Roger Deschner

H. Making VM faster

Possibly even more important to new users was the work that the older users were doing to make the system run faster. Not many IBMers believed that VM's performance problems could be solved, so not much was being done about those problems inside IBM.

Most installations had started running VM so that they could run two flavors of OS on one processor while migrating from DOS or MFT or MVT to a VS system. Typically, then, the system programmer had fallen in love with VM and was doing everything he could to keep it around. He liked being able to test operating system changes during the middle of the day, and he really liked CMS. One of my earliest memories of a SHARE VM Group meeting is the CMS guru Dave Gomberg telling how he had made a mod that got back enough of his CPU that he'd be able to keep VM for four more months. Everyone cheered.

Dave Gomberg

The process of making guest systems perform better began as soon as the customers got their hands on CP. Lynn Wheeler had done a lot of work on this while he was a student at Washington State, but he was by no means the only one who had worked on it. The CP-67 Project had frequently scheduled sessions in which customers reported on modifications to CP and guest systems to make the guests run better under CP. These customers had measured and monitored their systems to find high overhead areas and had then experimented with ways of reducing the overhead.[122] Dozens of people contributed to this effort, but I have time to mention only a few.

Dewayne Hendricks[123] reported at SHARE XLII, in March, 1974, that he had successfully implemented MVT-CP handshaking for page faulting, so that when MVT running under VM took a page fault, CP would allow MVT to dispatch another task while CP brought in the page. At the following SHARE, Dewayne did a presentation on further modifications, including support for SIOF and a memory-mapped job queue. With these changes, his system would allow multitasking guests actually to multitask when running in a virtual machine. Significantly, his modifications were available on the Waterloo Tape.

---

[122] For example, see S. Wecker, "OS/360 as a Preferred Machine Under CP-67", *Proceedings of SHARE XXXVII*, August, 1971, pp. 48-57.

[123] Then at Southern Illinois University, now with Allegro Technology.

Dewayne Hendricks

Dewayne became the chairman of the Operating Systems Committee of the SHARE VM Project. Under his guidance, the Committee prepared several detailed requirements for improvements to allow guest systems to perform better. At SHARE XLV, in 1975, the Committee presented IBM with a White Paper entitled *Operating Systems Under VM/370*,[124] which discussed the performance problems of guests under VM and the solutions that customers had found for these problems. Many of the solutions that Dewayne and others had found, such as PAGEX, made their way into VM fairly quickly, apparently as the result of customers' persistence in documenting them. By SHARE 49, Dewayne was able to state that, "It is now generally understood that either MFT or MVT can run under VM/370 with relative batch throughput greater than 1."[125] That is to say, they had both been made to run significantly faster under VM than on the bare hardware. Dewayne and others did similar work to improve the performance of DOS under VM. Other customers, notably Woody Garnett[126] and John Alvord, soon achieved excellent results with VS1 under VM.

Having gotten MFT, MVT, DOS, and VS1 to perform well under VM, the user community still had before it the challenge of making SVS and MVS perform well, too. Many people attacked this problem, including Sean McGrath, who succeeded Dewayne as the chairman of the Operating Systems Committee, and who continues today to do wonderful things to make MVS run better under VM. The problems in running SVS and MVS under VM were mostly due to CP overhead, rather than to I/O constraints as in the other cases. One of the VM manuals said explicitly that MVS under VM was not a production configuration. And that was true. People were seeing relative batch throughputs as low as 2 percent.

---

[124] *Proceedings of SHARE XLV*, August, 1975, pp. 493-497.

[125] D. Hendricks, "Primer on Operating MFT/MVT Under VM/370", *Proceedings of SHARE 49*, August, 1977, p. 475.

[126] Now at IBM.

Sean McGrath

Woody Garnett

Robert Fisher was a system programmer at Texas Instruments who was faced with the problem of migrating from MVT to SVS to MVS with only one 168. His only choices were either to get MVT and SVS and MVS to perform well under VM or else to stay on MVT forever, so he made the guest systems perform. At the outset, he resolved to make no mods to the guests. However, he did ultimately ZAP one instruction in SVS:

```
LRA   R2,0   ===>   SR    R2,R2
```

because that Load Real Address was consuming about 15 percent of his processor, which seemed an awful lot just to zero a register.

After a great deal of measuring and experimenting, Fisher concluded that the reasons for poor SVS- and MVS-under-VM performance were shadow table maintenance, CPU timer maintenance, and VM page wait. He wrote some very sophisticated CP modifications to address the shadow table maintenance and timer maintenance problems, introducing multiple segment table origin stacks, “virtual=shadow” support, and fast paths through the program check first-level interrupt handler. Then he put together an impressive presentation in which he described his results and suggested changes to VMA that, together with his changes, could result in MVS under VM running with 90 percent relative batch throughput.

The first slide in Fisher’s presentation was:

VS2 will perform
well on VM
when.....

and the last slide was:

....when
IBM wants it to.

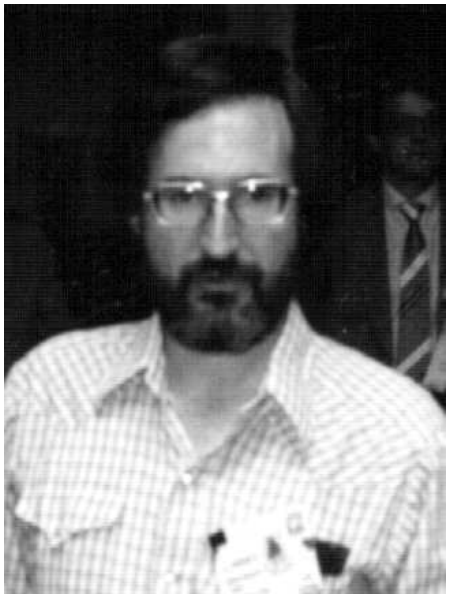
Fisher first gave the presentation at SHARE XLVI, in February, 1976. He continued to give it at SHARE and GUIDE over and over until IBM had to acknowledge his results. He then worked closely with IBM while they developed the Systems Extensions Program Product (SEPP), which implemented a great many of his suggestions.

During this same era, the VM community was working hard to improve CMS performance. IBM had taken the position that CMS was not a production system either, but during VM's first five years, several customers, such as Pat Ryall[127] and others at Bell Northern Research, successfully established CMS-only data processing centers. Doing this meant adding function that CMS was missing, improving CMS performance, and convincing IBM that VMA could be useful for CMS as well as for guests.

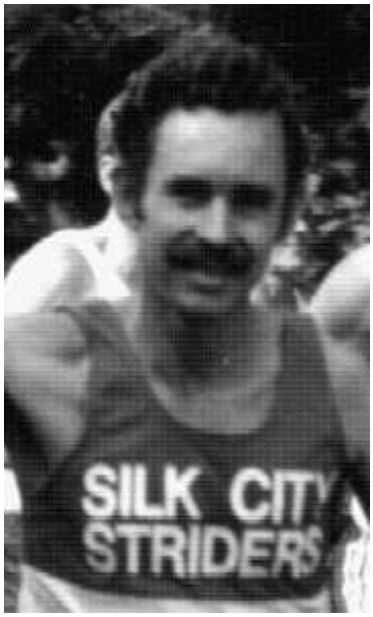
Pat Ryall

[127] Now with Apple Computer.

Considerable work was also being put into making CP itself run faster. Jim Best, of Pratt & Whitney, developed the drum multiple exposure modification, three lines of code that trebled the number of pages a system could turn in a given time. Two installations, Cornell and MITRE, wrote new CP schedulers that were a great improvement over the IBM scheduler, and they made their code available to other installations. Larry Brenner,[128] of Cornell, implemented drum-to-disk page migration, which greatly improved interactive response time.



Larry Brenner



Jim Best

The VM Project in SHARE was also working very hard to provide IBM with guidance. The Project produced White Papers on running guests under VM, on the CMS file system, on the CP scheduler, on RSCS, and on other topics. Furthermore, members of the community had already implemented most of the facilities discussed in their White Papers, so they were in a very good position to tell IBM exactly what was needed.

[128] Brenner is now in AIX Development at IBM. He made many other important contributions to VM over the years, including the CMS "IPLer" concept, which greatly facilitated supporting multiple versions of CMS.

I. The birth of VMSHARE

The most important step the community took to support the system and unite the users was announced at SHARE XLVII in Montreal in August, 1976. There the great David N. Smith announced the birth of the VMSHARE electronic conference with this slide:

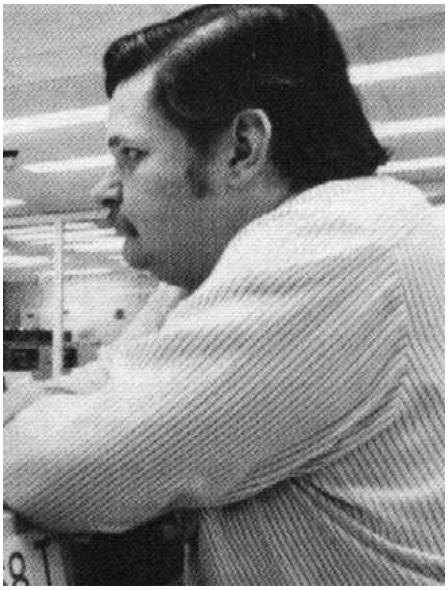
SHARE

VM Project

Community

Bath

(and Login)



Dave Smith

Dave worked at TYMSHARE and was the SHARE CMS Committee Chairman. He was greatly admired for the wonderful things he was doing with CMS, particularly in the area of performance. One of his most famous statements was, "CMS is like a sponge—touch it anywhere and performance squirts out."

Electronic conferences were new back then. A few people had begun playing around with the idea, but not much was happening yet, and Dave was unaware that anybody else was working on conferencing. He came up with the idea as the result of Ed Haskell's having asked him, "Why can't you do something with that fancy network of yours[129] so that we can communicate between SHAREs?"

[129] TYMNET.

Dave wrote all the software required for the conference[130] and also persuaded his employer, TYMSHARE, to provide the conference and the networking to the SHARE VM Project at no charge.

Then, Dave went to SHARE and told the VM Project to start using the conference. It can be very difficult to get a conference off the ground, for there is no incentive for anyone to log on until a goodly number of other people are already logging on regularly. But, Dave knew how to take care of that problem. Members of the VM Project soon learned that to go a week without logging on to VMSHARE was to invite a phone call from Dave. Dave's stature within the Project—physical, intellectual, and moral—was such that nobody said "no" to him. As a result, the conference soon blossomed into the heart and soul of the VM community and became essential to people supporting VM systems.

By the following SHARE, in March, 1977, more than half of the Project members had registered to use the conference, and there were some very vigorous discussions taking place. At that SHARE, Dave went around posting banners reading,

VMSHARE: At least once a week!

[130] In implementing VMSHARE (in EXEC Classic), Dave had to build one of the earliest CMS "padded cells", which resulted in one of my favorite of the early VMSHARE files:

<<< MEMO SECURE >>>

Secure memo you can never see!!

If you type the contents of this memo you are entitled to one free drink at SCIDS at Houston. This file is protected by an access list containing only $AD and thus invisible to you. (Of course!!)

Send mail if you crack my 'security'.

dave smith

*** CREATED 01/06/77 22:51:35 BY TST ***

gotcha!

*** APPENDED 10/10/78 16:35:58 BY AMD ***

John Mort, an IBM representative to the Project, wrote a letter[131] the following month commending the Project for setting up VMSHARE and noting that VMSHARE had already saved IBM trouble and expense:

> I want to commend the VM/370 Project for their creativity in establishing VMSHARE.
>
> Its benefit to users in avoiding redundant effort is obvious. I would like to comment upon VMSHARE's contribution toward improving the Project's ability to impact the product itself.
>
> Questions pertaining to resolutions are now asked more frequently than previously. Prior to Houston, VMSHARE allowed us to accept two resolutions and avoided our rejecting another through lack of additional information. In another instance, VMSHARE allowed us to provide missing microfiche on a PLC at least 2-3 months earlier than 'normal business'.
>
> The technical dissertation presently going on as a follow-on to a White Paper response at Houston is extremely valuable to us in understanding customer needs and requirements.
>
> To sum it all up, SHARE is now 365 days a year rather than 4 times a year. Please convey my admiration to the developers.

VMSHARE saved IBM from expense and embarrassment over and over in the years that followed, as well as providing customers with expert assistance and early warning of problems. It also allowed more dialogue between IBM and customer technical people than had ever been possible before.

Many of us are convinced that VM would not have survived if Dave hadn't given us VMSHARE.

Early in 1979, the next VMSHARE Administrator, Charles Daney, extended VMSHARE to Europe, when he arranged for members of the SHARE European Association (SEAS) to participate in VMSHARE, following an ardent campaign for this by Jeff Gribbin, of SEAS.

Charles Daney

[131] J.R. Mort to E.S. Haskell, April 11, 1977, *SHARE VM/370 Project Minutes*, 1977.

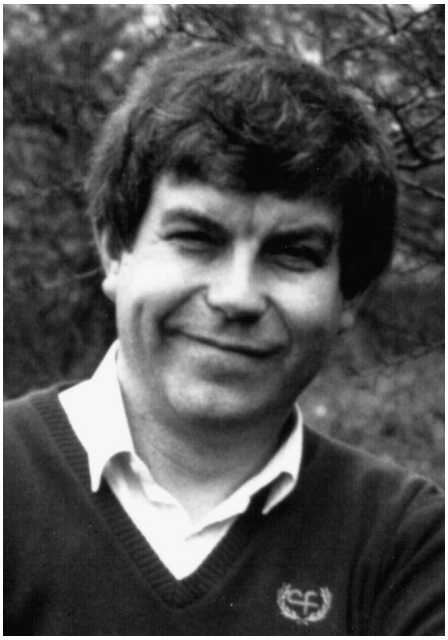
<<< MEMO VMSEAS >>>

AT LAST I'M LEGAL!!! EVERYBODY PLEASE NOTE MY NEW PROJECT CODE IS _CU. ALL VMSEAS USERS MAY BE IDENTIFIED BY THE '_' PREFIX ON THEIR USERID. I HOPE THAT WE ARE ALL GOING TO BE USEFUL CONTRIBUTORS; THERE'S ANOTHER 16 USERS COMING UP WITHIN THE NEXT WEEK OR TWO.

REGARDS, JEFF GRIBBIN (COMMERCIAL UNION ASSURANCE, U.K.)

*** APPENDED 02/20/79 02:48:41 BY _CU ***

Jeff, as SEAS VMSHARE Administrator, insisted that SEAS members start using VMSHARE, just as Dave Smith had done with SHARE members two years earlier. Their response was enthusiastic, and soon the VM community had truly become an electronic global village.



Jeff Gribbin

J. The birth of VNET

VNET, IBM's internal network, united and strengthened the VM community inside IBM in the same way that VMSHARE united and strengthened the VM community in SHARE and SEAS.

Tim Hartmann

"VNET—1000 Nodes—1983"

The VNET network, like many of the other good things we have today, was put together "without a lot of management approval", to quote Tim Hartmann,[132] one of the two authors of RSCS. VNET arose because people throughout IBM wanted to exchange files. It all started with Hartmann, a system programmer in Poughkeepsie, and Ed Hendricks, at the Cambridge Scientific Center. They worked together remotely for about ten years, during which they produced the SCP version of RSCS (which came out in 1975), and the VNET PRPQ (which came out in 1977). After that, RSCS was turned over to official developers.

The starting point for RSCS was a package called CPREMOTE, which allowed two CP-67 systems to communicate *via* a symmetrical protocol. Early in 1969, Norm Rasmussen asked Ed Hendricks to find a way for the CSC machine to communicate with machines at the other Scientific Centers. Ed's solution was CPREMOTE, which he had completed by mid-1969. CPREMOTE was one of the earliest examples of a "service virtual machine" and was motivated partly by the desire to prove the usefulness of that concept.

[132] Now with VNET Communications.

CPREMOTE was experimental and had limited function, but it spread rapidly within IBM with the spread of CP-67. As it spread, its "operational shortcomings were removed through independent development work by system programmers at the locations where [new] functions were needed."[133] Derivatives of CPREMOTE were created to perform other functions, such as driving bulk communications terminals. One derivative, CP2780, was released with VM/370 shortly after the original release of the system.

By 1971, CPREMOTE had taught Hendricks so much about how a communications facility would be used and what function was needed in such a facility, that he decided to discard it and begin again with a new design. After additional iterations, based on feedback from real users and contributions of suggestions and code from around the company, Hendricks and Hartmann produced the Remote Spooling Communications Subsystem (RSCS).

When the first version of RSCS went out the door in 1975, Hendricks and Hartmann were still writing code and, in fact, the original RSCS included uncalled subroutines for functions such as store-and-forward that weren't yet part of the system. The store-and-forward function was added in the VNET PRPQ, first for files, and then for messages and commands.

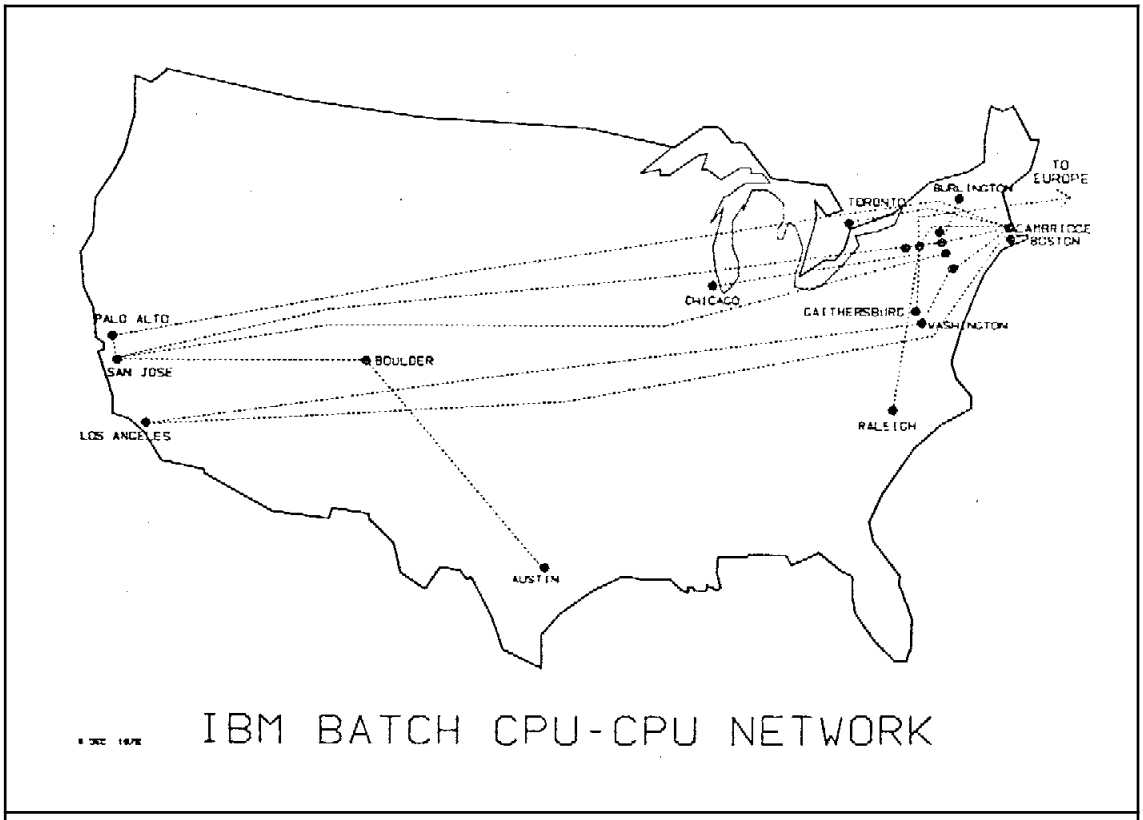
Once that capability existed, there was nothing to stop a network from forming. Although at first the IBM network depended on people going to their computer room and dialing a phone, it soon began to acquire leased lines. The parts of IBM that were paying for these lines were not always aware of what they were paying for. Since the network grew primarily because the system programmers wanted to talk to one another, a common way of acquiring leased lines for the network was to go to one's teleprocessing area and find a phone circuit with nothing plugged into it.

The network was originally called SUN, which stood for "Subsystem Unified Network", but at first it wasn't actually unified. It was two separate networks that needed only a wire across a parking lot in California and a piece of software (which became the RSCS NJI line driver) to make them one. Hartmann spent some time in California reverse-engineering the HASP NJI protocol, which hadn't really been written down yet, and finally got that last link up late one evening. Wishing to commemorate the occasion, he transferred some output from a banner printing program running on his system in Poughkeepsie through the network to a printer in San Jose. His co-worker in San Jose, Ken Field, the author of the original HASP NJI code, thought Tim's output was pretty nifty, so he asked for more copies and taped them up on the walls before finally going home to get some sleep. When Field got back to work late the next morning, he found the place in an uproar over the apparent unionization attempt. The banners had read:

Machines of the world unite! Rise to the SUN!

[133] E.C. Hendricks and T.C. Hartmann, "Evolution of a Virtual Machine Subsystem", *IBM Systems Journal.*, vol. 18, no. 1, 1979, p. 122.

After that got quieted down, the network began to grow like crazy. At SHARE XLVI, in February, 1976, Hendricks and Hartmann reported that the network, which was now beginning to be called VNET, spanned the continent and connected 50 systems. At SHARE 48, a year later, they showed this map of the network:



VNET Network, 1976 [134]

By SHARE 52, in March, 1979, they reported that VNET connected 239 systems, in 38 U.S. cities and 10 other countries. In August, 1982, VMers celebrating VM's tenth birthday imprudently attempted to hang the current VNET network map up at SCIDS. By that time, a circuit analysis program was being used to generate the network maps. In 1983, VNET passed 1000 nodes. It currently connects somewhat more than 3000 nodes, about two-thirds of which are VM systems. Nobody even attempts to print a map of the network any more.

VMers inside IBM became "networked" very early, and that fact was critical for the survival of VM. One of the most important uses of the network in the early days was for the distribution of the *VM Newsletter*, more commonly known as "Capek's newsletter", for its editor, Peter Capek. The *VM Newsletter* went through fifty editions in the late 1970's and early 1980's, ultimately attaining a circulation of 10,000. To submit an article to the newsletter was to assure that it would be brought to the attention of most VMers in the company.

---

[134] *SHARE VM/370 Project Minutes*, 1977.

One especially delightful item from the *VM Newsletter* was the following:

> I am happy to announce the recent marriage of Kittredge Cary and Michael Cowlishaw. Since Kittredge had been living in the U.S. and Mike in England, a major portion of the courtship occurred using the network, and has been recorded down to the tiniest coo.[135] In order that others may profit from the wooing technology that has been developed, it is the subject of two upcoming "how to" issues of the *VM Newsletter*, one each from the male and female perspective. Congratulations again to Kittredge and Mike!

Mike Cowlishaw

Peter Capek

Capek's newsletter was ultimately superseded in 1983 by a more interactive use of the network, the electronic conference, IBMVM. IBMVM is much like VMSHARE in that it contains many individual files (or "fora"), each pertaining to a particular topic. Participants may "append" comments to these files using Mike Cowlishaw's Tools system,[136] which posts their appends to the master copy of the conference disk and also keeps more than 250 "shadow" copies of the conference disk around the world synchronized with the master. (VMSHARE appends are posted to IBMVM on a daily basis, providing many IBMers with their only customer contact. A mechanism also exists for IBMers to post appends to VMSHARE.)

The existence of VNET has greatly facilitated the functioning of the company and the spread of information. By 1986, *Think* magazine estimated that VNET was saving the company $150,000,000 per year as the result of increased productivity. VNET has made it much more possible for IBMers in different parts of the world to work closely together, so much so that there

---

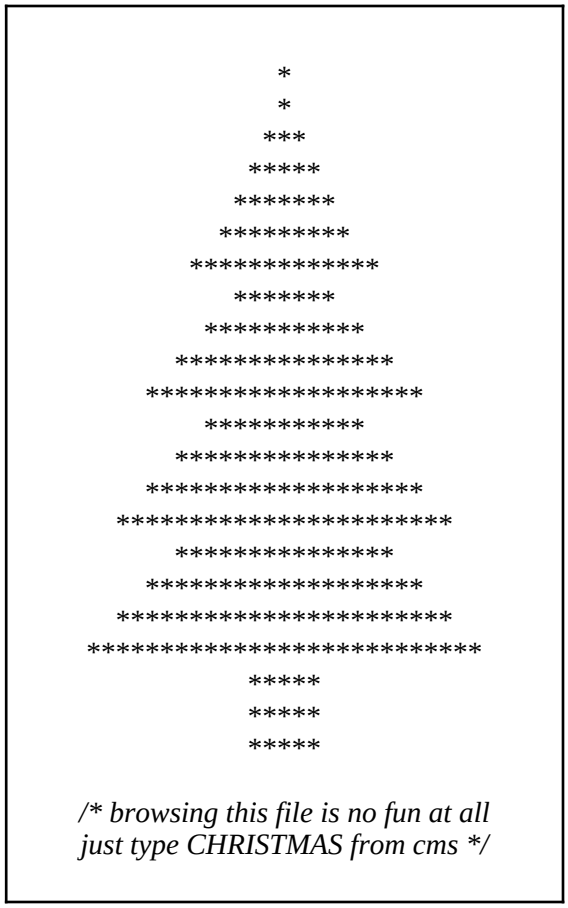
[135] It has been suggested that I should state explicitly that IBM does not actually record the contents of mail files exchanged between its employees.

[136] A subset of the Tools system was released a few years ago as the VM/DSNX product. DSNX synchronizes multiple copies of a disk around a network, but unfortunately does not contain the conferencing function from Tools.

is a word in IBM jargon to describe the condition one gets into when working with someone in a distant time zone:

> **Netlag** n. The result of one's internal (biological) clock being out of synchronisation with local time due to working on an electronic network. For example, there is a tendency among European IBMers to live on the USA clock in order to have a maximum working time overlap with their US colleagues. This means that they constantly look as though they just stepped off an overnight flight.[137]

IBM can really no longer function without VNET, as it learned in December, 1987, when a nasty little program called CHRISTMA EXEC snuck through one of the gateways into VNET from the U.S. university network BITNET after it had snuck into BITNET from the European university network EARN. Rumor has it that only two of the copies of CHRISTMA that went through the gateway actually got executed inside IBM. But, by the next morning, whole countries had pulled out of the network to protect themselves, and some systems had received as many as 20,000 copies of the file. That really brought home to IBM how dependent it is on the network, which is but one of several important gifts it has received from the VM community.



[137] M.F. Cowlishaw, *IBM Jargon and General Computing Dictionary*, Tenth Edition, May, 1990, p. 36.

K. The Roller-Coaster Ride

In the early 1970s, the VM/370 users badly needed to get IBM's attention. Then they were very fortunate in finding just the advocate they needed within IBM. Jerry DePass is a very gentle and gentlemanly man. He has hardly ever been heard to say an unkind word to anyone, except once, under extreme provocation, when he said to Fred Jenkins, "Your mother runs TSO!"

Jerry DePass

Jerry became the VM Product Administrator in 1974. He had been offered the choice of VSAM or VM and chose VM, not knowing that he was walking into a hornet's nest. Right after Jerry arrived in White Plains, he learned that he was expected to go to the November SHARE Interim in Montreal, but that he was also expected to be at a meeting with the developers in Burlington at the same time. He chose the latter.

Bruce Marshall was the SHARE VM Project Manager then, and Bruce was just about fed up. VMers had waited two years after System/370 was announced before IBM announced a S/370 that CP and CMS would run on. And even now, two years later, VM was still full of holes. Bruce's project was putting much effort into white papers and requirements, but nothing was coming of them. So, when *no* IBM rep showed up for the Montreal Interim, Bruce flipped out. After the Project Opening session, he walked down the hall, took a dime out of his pocket, and phoned *Computerworld.* That lost him his SHARE ribbon (although he later became a Director of SHARE) and almost lost him his job.

COMPUTERWORLD

THE NEWSWEEKLY FOR THE COMPUTER COMMUNITY

Weekly Newspaper Second-class postage paid at Boston, Mass., and additional mailing offices. © 1974 by Computerworld, Inc.

Price: $12/year December 4, 1974 Vol. VIII, N

Resolutions 'Ignored' for Two Years

Where Is IBM's VM/370 Help? Share Group Asks

By Don Leavitt

MONTREAL — Where is IBM's Virtual Machine facility (VM/370) heading? Why won't IBM respond to user's concerns about the facility?

These questions haunted Bruce Marshall, VM project manager of the IBM users group Share, as 30 to 35 members of the project got together here last week in a mini-Share meeting. "But IBM didn't even bother sending anyone to talk to us," he said.

The project membership had submitted some 95 resolutions to IBM over the past two years seeking clarification or additional support, he said, emphasizing that his frustration at the vendor's unresponsiveness is not a sudden thing.

The latest resolution charges simply that "IBM is not supporting its product in any manner." That summed up the situation so well, he said, it came to the top of a

3340 was first announced, even though the device would be a very nice one for VM.

Customer education facilities are poor, the resolution said. "IBM doesn't have very many classes. They don't have an operator training class. The don't have, in general, the comprehensive support for user education for VM," Marshall argued.

Field engineering support for maintenance and "fixes" was another area cited by the resolution as needing more attention. A branch office will not have the depth of personnel it should have. "They'll have one person trained in VM and if he goes away to school that's the end of it," the project leader charged.

Customer engineers (CEs) are "very wary" of VM. The facility is supposed to be very good for doing on-line hardware diagnostics, but "most CEs will not trust it," Marshall went on.

An outgrowth of the CP/CMS opera-

installed and being used in a production environment. The other shipments went to people who are "intrigued with the system and who get it in to try. The system programmer tries it and plays with it. It may catch on; it may not," he explained.

The machine base figures, accurate in Marshall's view although he wouldn't reveal his sources, are 10Fs, 370/115s, 50Fs, 145s, 30Fs, 155s and 158s, and another 10Fs, 165s and 168s. These obviously represent a "substantial amount of hard-

ware rental dollars," he noted.

'Poor Man's TSO'

IBM's original market probably wa DOS user on the 135 or 145 who wa time-sharing. Marshall thought. The subsystem would allow him to hav poor man's TSO."

The VM facility, which supports si taneous use of DOS and OS along CMS, might also be seen as a br product for some users moving from

(Continued on Page 5)

IBM Satellite Plans Hit As FCC Holds Hearings

By Ronald A. Frank

The session was held to discus

"Computerworld", December 4, 1974

The *Computerworld* article got Jerry a *lot* of attention inside IBM. Years later, Jerry could look back and say that it had been a good thing, because it got him audiences with IBM managers he'd never otherwise have been able to talk to about VM, but at the time it hadn't looked so good.

However, he never missed another SHARE between then and the time seven years later when he left IBM to become one of the founders of the Adesse Corporation.[138]

The VM people in SHARE were not the only ones who were unhappy. Shortly after this time, the GUIDE CMS Project Manager, Walt Hutchens[139] (the author of the MITRE Scheduler), wrote IBM an angry letter about the lack of new function in CMS.[140] The SEAS VM Project Manager, Francesco Carreras, sent an angry letter to IBM demanding that IBM send the highest person responsible for VM development to the next SEAS meeting.[141] Then he resigned as Project Manager (for unrelated reasons) and left Hans Deckers holding the bag. Hans led the SEAS VM Project for many years after that and was followed in that job by Iain Stinson, who later became President of SEAS. The VM community owes all of these gentlemen thanks for their tireless efforts on behalf of VM.

Iain Stinson

Hans Deckers

[138] One of his partners at Adesse was Bruce Marshall, who was still marvelling that Jerry held no grudge against him for the *Computerworld* escapade.

[139] Now with SysVM, Inc.

[140] Walter A. Hutchens to B. Suzanne Burr, April 19, 1977, *SHARE VM/370 Project Minutes*, 1977.

[141] Francesco Carreras to the Manager of VM Development, November 12, 1976, *SHARE VM/370 Project Minutes*, 1977.

After the *Computerworld* blast, Jerry started learning about VM and trying to figure out why the VM customers were so unhappy. He soon found that IBM had no plan for VM. IBM had no intention of committing more resources to VM. The current marketing forecast was that there would never be more than 500 VM customers. Having been convinced by the customers that VM was too good to let die, Jerry set out to convince the rest of IBM of the same thing and to get that forecast changed. He had no staff to help him, but he did have the customers on his side, and many of them could draw on the resources of their own big companies to help him in the fight. He kept struggling, and more than once was warned that he was being reckless. He later described his first few years as VM Product Administrator as "exhilarating—like riding a roller coaster".

SHARE XLVII, in August, 1976, is perhaps a perfect illustration of the roller-coaster that the VM community was on during this period. By that time, there were 300 VM accounts. One hundred fifty people attended the VM Project Opening session, at which IBM made a variety of very encouraging and very discouraging announcements:

- IBM reported that Development's move to Poughkeepsie[142] had been completed early and that Burlington had closed on August 22. Although an attempt was made to put a good face on this, everyone knew that it was a move to get VM under control.

- IBM reported that the "Wheeler Scheduler" (the Resource Manager PRPQ) had shipped earlier in the month. Unlike the rest of the operating system, the new scheduler was not free, but it was worth paying for. Not only did you get a visit from Lynn Wheeler, you also found that you could suddenly support ten percent more users.

- IBM announced that the VM Project's requirement VM-142-0276 was being closed as rejected. That was the first time that IBM rejected a VM source requirement.

- IBM accepted a requirement to provide handshaking for other SCPs besides VS1 and a requirement for improved computer networking facilities for VM.

- And, to the amusement of all, IBM announced the cancellation of a session that had been scheduled to discuss the notorious *IBM Systems Journal*[143] article on penetrating a VM system, giving the explanation that the people who had done the study felt that there was nothing more to say on the subject.

---

[142] Shortly after that, a VM Development effort started up in Endicott, as well, with the objective of supporting the mid-range processors built in Endicott, while the Poughkeepsie group became more and more tied to the objective of supporting the large processors built in Poughkeepsie.

[143] C.R. Attanasio, P.W. Markstein, and R.J. Phillips, "Penetrating an Operating System: A study of VM/370 Integrity", *IBM Systems Journal*, vol. 15, no. 1, 1976, pp. 102-116. The IBM order number for a reprint of this article is G321-5029.

Following the closing of Burlington in 1976, the roller-coaster ride got even rougher, as we began receiving new releases of VM that seemed to have more new bugs than new function. The problem was simply that there were very few people working on the product who actually knew anything about it. Only twenty-four percent of the Burlington personnel had moved to Poughkeepsie when VM was moved.[144] Most of the people we've been talking about up to now leave the VM story at this point, because Poughkeepsie was seen as a boring place to live and a bureaucratic place to work. Boston and its suburbs provided other opportunities that didn't require the VM developers to move away from the bright lights and disrupt the lives of their families. By the end of 1976, sixteen percent of the Burlington personnel were working for DEC. Forty-seven percent had found IBM positions that didn't require them to move to Poughkeepsie. To make matters worse, several of the most knowledgeable CP people who did go to Poughkeepsie, including Dick Newson and Per Jonas, were put into a separate group whose purpose was to build a tool that could be used for the development of MVS/XA. Starting with VM/370 Release 3, PLC 06, they went on to build a fast, stripped-down CP that could create XA virtual machines on a real 370.

Per Jonas

The roller coaster ride continued as 1977 brought us VM/370 Release 5, with the new program product VMAP, and the new systems extensions, BSEPP and SEPP, which incorporated the Wheeler Scheduler, as well as replacing some of the most common user modifications, such as accounting to disk (rather than to real punched cards), page migration, swap table migration, and shadow table maintenance enhancements.

[144] *Directory of the Virtual People*, December 17, 1976.

"The Class of '76"

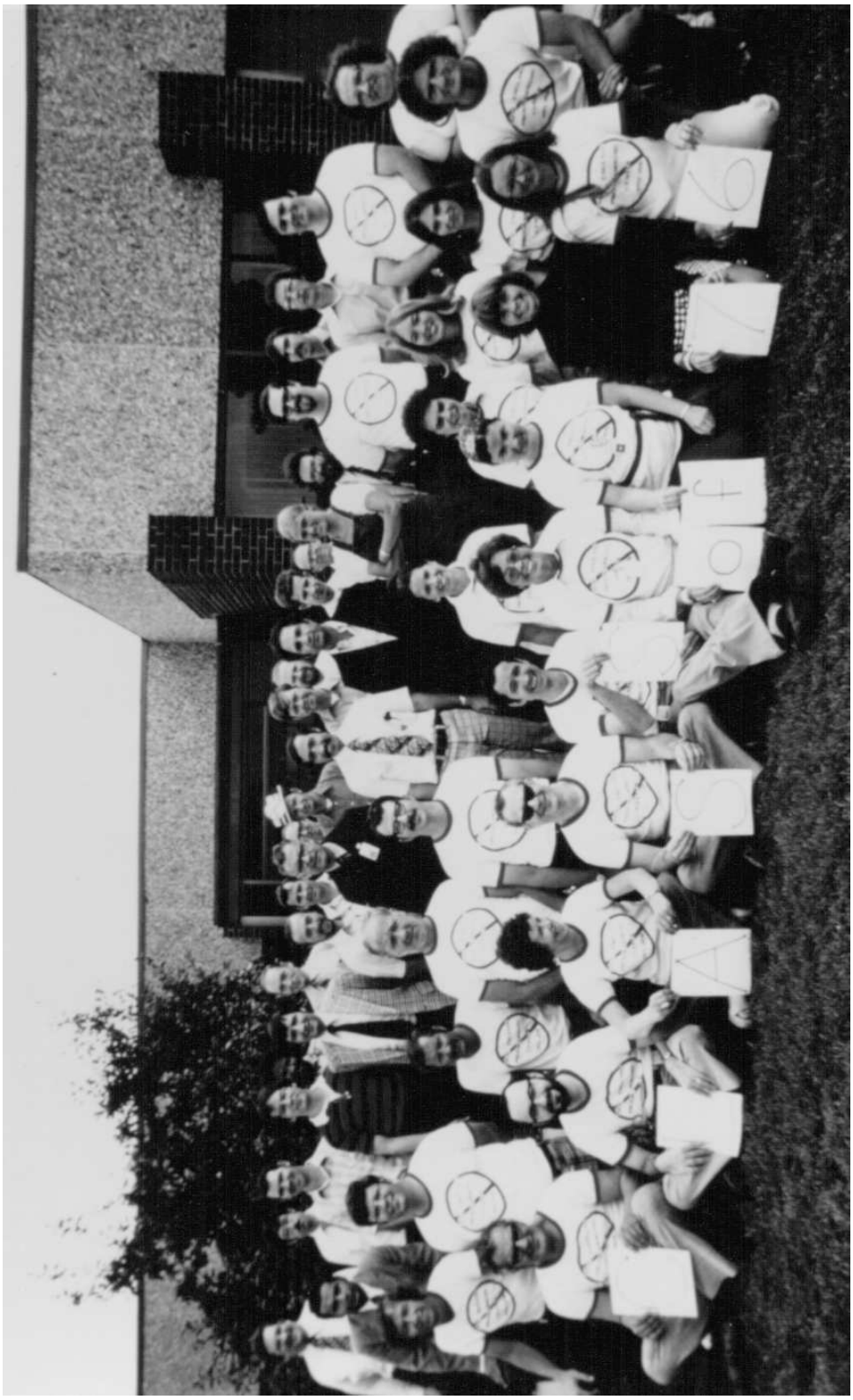
The photograph on the next page is of the Burlington personnel shortly before the closing of the New England Programming Center.

First row: Joe Steene, David Kelleher, Barbara Whitehall, Bob Harris, Hank Schmitz, Shirley Schick, Jane Devlin, Gail Kokko, and Patti Anklam.

Second row: Dick Milley, Bill Barrett, John Shaw, Frank Smith, Tom Rosato, Jean Chase, Judy Marcus, Sharon Milley, Pauline Balutis, and Paul Fay.

Third row: Eddie McNeil, Charlie Johnson, Richard Pedersen, Tom Cafarella, Don Wagler, John Seymour, Alec Vlahos, Felix Puopolo, Maureen McGuigan, Dave Tuttle, Morgan Stewart, Geoff Bartlett, and Ve Tavitian.

Fourth row: Curt Endee, Bob Downs, Art Reid, Bob Goodman, Walt Wisnowski, Roger Thompson, Fred Horton, Bob Doherty, (unidentified), Lyn Hadley, Dick Seymour, Dave Waite, and Joe Leary.



"The Class of '76"

L. Getting IBM's attention

At the suggestion of Jerry DePass, the SHARE VM Project had begun working in 1974 to put together a business case for VM that they hoped would be meaningful to IBM management. At about the same time, the GUIDE VM Project, with help and encouragement from Love Seawright, also began working on a business case for VM. SHARE's effort was organized and led by Ed Haskell, who later became President of SHARE. GUIDE's effort was under the leadership of the great innovator Walt Hutchens.

Walt Hutchens

Ed Haskell

The final form of GUIDE's business case was their presentation *VM/370 in 1978.*[145] The final form of SHARE's business case was a presentation entitled *Why VM?* that Ed made to the Director of IBM's Poughkeepsie Lab in November, 1977.

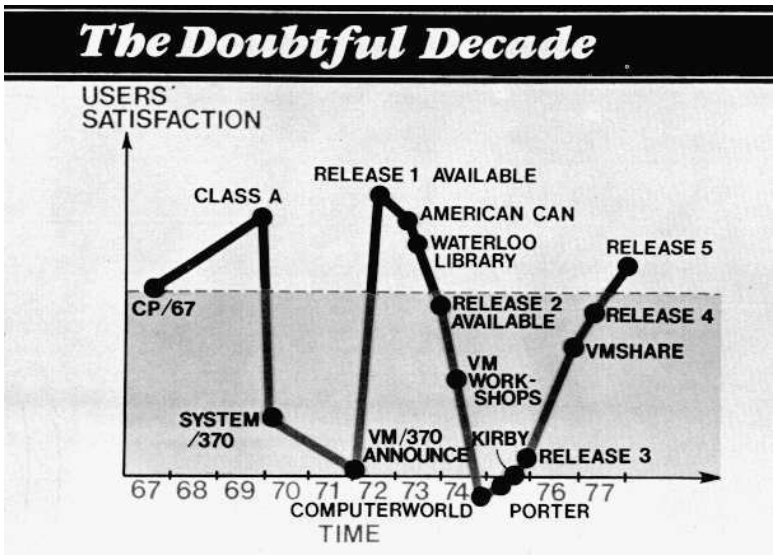
"VM/370 in 1978"

The background for all the title slides in the *Why VM?* presentation displayed a list of customers running VM.

"Why VM?"

[145] *Proceedings of SHARE 51*, August, 1978, pp. 2-26.

The presentation began with an overview of VM's history that portrayed the causes for customer dissatisfaction.



It then presented case studies of half a dozen VM customers with an emphasis on the growth rates of their VM systems and their plans for future expansion. That was followed by an explanation of why customers value VM:

VM/370 users are committed to interactive computing and simplicity.

and a quantification of the business opportunities that IBM was losing because of its inadequate support for VM. The presentation concluded:

VM/370 is valuable to IBM customers in ways that other SCPs are not.

**VM users grow rapidly
subject, in part, to
limitations on the availability
of enhancements to the SCP.**

**IBM can reap great benefits
by enhancing VM/370 in ways
that meet customer needs.**

M. After the Doubtful Decade

By the end of 1978, there were 1,000 VM installations. In 1979, an entire issue of the *IBM Systems Journal*[146] was devoted to VM. Clearly, the Doubtful Decade was over. Although it took years before we saw much new function, IBM did at last have a plan that included VM.

1979 brought us VM/370 Release 6 and Release 2 of BSEPP and SEPP, with logical device support, the EDF file system,[147] and the first of many very disappointing implementations of

---

[146] Volume 18, number 1. The IBM order number for this issue is G321-0056.

[147] On January 11, 1978, during the internal negotiations on the design of the new file system, Dave Smith, who had recently left TYMSHARE and joined IBM Research, produced the following classic letter:

> I have been studying the file system subject extensively for the last month and I am now prepared to hand down my blessing on the correct and proper path for CMS file systems development.
>
> I have studied a file system written by Chris Stephenson of Yorktown which is called the New File System or NFS for short. I have studied a file system designed by Dale Witt of Endicott which is called the New Control Block File System or NCB for short. I have also studied a third file system, the Antique File System or AFS for short.
>
> NFS is modern, elegant, sparse, tightly constructed, and is available for use. (One is tempted to say that NFS is tannic in flavor and needs several more years of bottle age.)
>
> NCB is traditional in design with modern overtones but with less bulk than NFS. It is still under construction and all I have seen is some initial blueprints.

HELP. The RSCS Networking Program Product was announced on the same day as Release 6.

Once it was confident that IBM was prepared to invest real money on VM, the SHARE VM Group withdrew all its outstanding requirements, which had been written when we knew we shouldn't ask for much, and set about developing new, longer-range requirements.

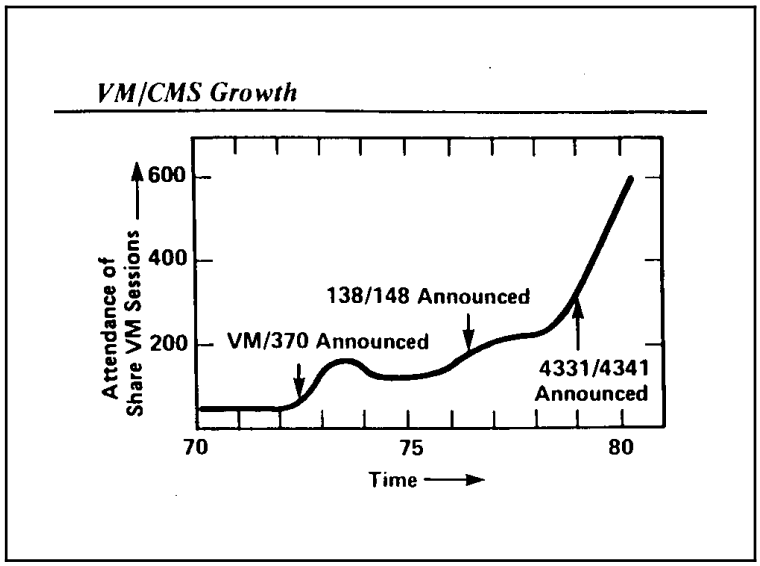
Fred Jenkins

In 1980, the VM Group's VM/CMS Task Force, under the leadership of Fred Jenkins (whose mother runs TSO), presented IBM with a White Paper entitled *VM/CMS for the Eighties.*[148] In the presentation accompanying their White Paper, the Task Force again used a list of VM installations for the background of the title slides, but the print had to be finer this time.

> AFS is a fine example of its period but showing the results of having had to adopt to the requirements of different ages. One can see the dust particles in the newer coats of paint as well as several chips and scratches. Additions to the design have not added to its basic beauty, but this is common in many older pieces.
>
> Despite the availability of newer pieces which might fit the usage pattern more exactly, I feel that it is important to preserve our heritage, and I therefore recommend that AFS be reconditioned. I feel that we cannot condemn the past since that would deny the future the propagation of the best of the present, but that we must deny the present so that we can propagate the past and preserve it for the condemnation of the future.

[148] *SHARE SSD*, no. 311, February/March, 1981.

The Task Force demonstrated VM's growth by a plot of the attendance at VM sessions at SHARE:



There followed a discussion of how fast VM would grow if the inhibitors to its growth were removed and a detailed discussion of those inhibitors. The presentation ended with a plea for improved dialogues between developers and customers to speed the process of removing the inhibitors:

Without dialogues,
if you tell them
you want something real bad,
you will get it real bad.

Beginning in the 1980's, as the number of VM installations grew dramatically, we began to see the birth of firms devoted to producing VM systems and applications software. The founders of this "cottage industry" were, for the most part, long-time VM gurus from customer shops and IBM, who knew from first-hand experience what function VM needed to make it a complete production system. They set about supplying that function commercially, thus enabling new VM installations to get started with substantially less expertise and initial investment than had been required earlier.

"VM/SP is waiting for you"

At the same time, we started seeing the results of IBM's new commitment to VM. VM System Product Release 1 came out late in 1980. SP1 combined all the [B]SEPP function into the new base and added an amazing collection of new function (amounting to more than 100,000 lines of new code): Xedit, EXEC 2, IUCV, MIH, SUBCOM, MP support, and more. SP1 was also amazingly buggy. The first year of SP1 was simply chaotic. The system had clearly been shipped before it was at all well tested, but the new function was so alluring that customers put it into production right away. So, CP was crashing all over the place; CMS was destroying minidisks right and left; the new PUT process was delaying the shipment of fixes; and tempers were flaring. When the great toolmaker Jim Bergsten[149] produced a T-shirt that warned "VM/SP is waiting for you", his supply sold out immediately.

[149] Then at Amdahl, now with Cthia Corporation.

Jim Bergsten
(courtesy of the artist, Wilt Byrum)

Again, as so many times in the past, IBM attempted to address the problems in SP1 by "throwing bodies" at them. In this case, many of the bodies were barely lukewarm, and they produced some stunningly awful "fixes", but eventually IBM and customers working together got the system stabilized.

SP1 included the first implementation of SNA for VM, VCNA. VCNA required a guest system, so VM system programmers unhappily set about learning SMP. The pain of that led one of them to coin the term "intruder virtual machine" to describe a guest whose only purpose is to implement function that should be native to VM.

Late in 1981, IBM released the Profs PRPQ, which was the "Electronic Office System" (EOS) that had been developed jointly by AMOCO and IBM for AMOCO's use. After the PRPQ proved to be a success, IBM released Profs as a full program product. By 1987, there were said to be a million Profs users outside IBM, and IBM itself had become heavily dependent on Profs.

About the time SP1 got settled down, in the Summer of 1981, the developers in Endicott switched from their heavily modified system to a "vanilla" system. Customers were very pleased to hear this, because they knew what the result would be. Sure enough, SP2 brought us the command retrieve function, and SP3 brought us PER, thus finally obsoleting two of the most common user modifications. SP2 also added the "productivity tools", which, combined with Xedit, greatly improved the end user interface. REXX finally came out in SP3, to the delight of all.

At the end of 1981, IBM announced two new flavors of VM, the High Performance Option (HPO) for high-end S/370 machines, and a rudimentary XA version of VM, the VM/XA Migration Aid, which was derived from "The Tool" that had been built for use during MVS/XA development.[150] Since then, all three flavors of VM have grown and sometimes prospered. The

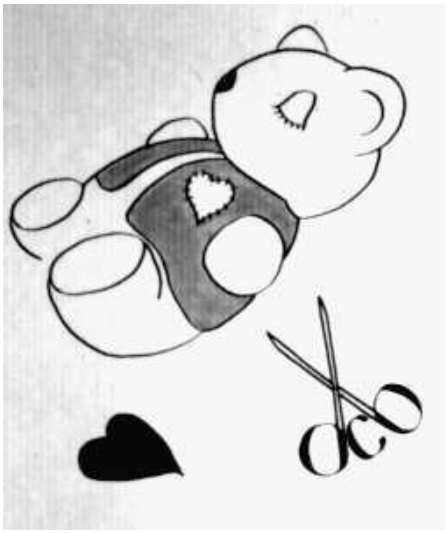
[150] "We started the design at the time of VM/370 Release 3 PLC 06 in 1976. We basically took the XA architecture and re-designed what had to be changed to make VM run on XA. We redesigned the RIO structure, the VIO structure, CCW translation, virtual storage

details are probably familiar to you all, so I will touch on only a few of the highlights (and lowlights) of the 1980's.

During most of that decade, IBM did so much for us and to us that the VM community operated mainly in reaction mode.

The most important thing that IBM did to us was the announcement on February 8, 1983, of the Object Code Only (OCO) policy. I fear that ten years from now another speaker will be standing here telling you that that was the day VM died, but I hope not.

Gabe Goldberg



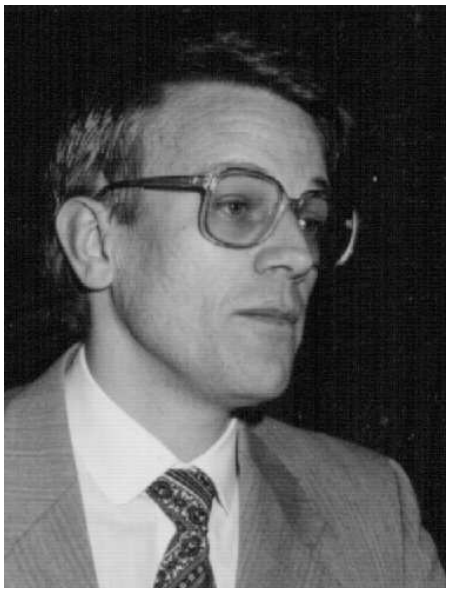
February 8, 1983

Since that day in 1983, the community has devoted enormous effort to attempting to convince IBM's management that the OCO decision was a mistake. Many, many people have contributed to this effort in SHARE and in the other user groups. The greatest of SHARE's source heroes is unquestionably Gabe Goldberg,[151] who has persevered and maintained hope and a sense of humor in the face of IBM's seemingly implacable position. In SEAS, Hans Deckers has been a particularly hard worker in the battle against OCO, and Sverre Jarp, the SEAS Past President, also deserves much praise for his role.

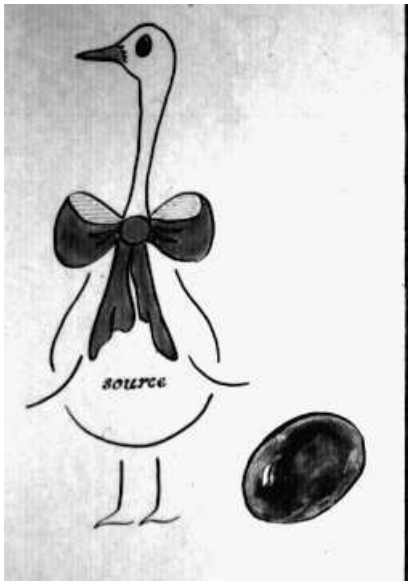
---

management, and a bunch of other stuff. We built a system that would run on a vanilla 370 and simulate an XA (except for 31-bit mode). We used this to develop the real XA operating system. First time we saw an engineering model (early 3081) with real XA on it, we IPLed and ran with no problem!! Virtual machines for you!!!" (S.R. Newson, private communication, 1989.)

[151] Of the VM Systems Group.



Sverre Jarp



"All Your Fears Are Sourceless"

In February, 1985, the SHARE VM Group presented IBM with a White Paper that concluded with the sentence, "We hope that IBM will decide not to kill the goose that lays the golden eggs." Though we had tried to make our White Paper reasonable and business-like, IBM chose not to reply to it.

A few months after the announcement of the OCO policy, IBM released the first OCO version of VM, VM/PC. VM/PC had a number of problems, including poor performance and incorrect or missing or incompatible function. Without source, the users were unable to correct or compensate for these problems, so nobody was surprised when VM/PC fell flat.

Woes such as this drew the community ever closer together in the 1980's. The tradition of supporting one another flourished. One of my favorite examples of this occurred right after we installed HPO 3.4 at Princeton. HPO 3.4 was a real boost for our system. The swapper reduced our trivial response time by a third. The reduction in cache interference between our 3081's two processors under HPO 3.4 got us back ten percent of the machine. The system was also extraordinarily stable. We had only one CP failure, caused by a bug in the paging subsystem that Bill Weeks, of SLAC, shot for me because I was so busy working on the "Source Force" White Paper.

However, HPO 3.4 introduced a major performance problem on our system. CICS response time was sixty percent higher than it had been under HPO 3.2. So, I was faced with a scheduler problem. I am not only not a scheduler guru, I have never even wanted to be a scheduler guru. However, the problem was quite obvious even to me: the V=V CICS guest's reserved pages were being trimmed and logically swapped, causing its paging rate to increase dramatically.

Having figured out that much, I tried a few simple-minded ideas for fixing the problem. One of my "fixes" made CICS response time better than it had ever been before, but completely destroyed the CMS users. Another of my fixes made both CMS and CICS response worse than they had ever been before.

Swapper Group and Executives

Standing: *Jim Cannavino, Betty Nichols, Bill Tetzlaff, Gerry Spivak, Gerry Greenberg.*
Seated: *Bill Buco, Tom Beretvas, Dave Patterson, Norb Tennessen.*

At that point, I sent a sorrowful note to Bob Cowles, at Cornell. Bob's first reaction was completely justified; he told me that I wouldn't have problems like this if I would buy his scheduler (the Adesse System Resource Manager). However, a few hours later, he gave in and sent me a three-line fix, with a note saying that if it assembled, it would solve my problem. It assembled and it solved my problem. Both CMS and CICS response time were improved in comparison with HPO 3.2.

Bill Weeks

Bob Cowles

There followed a few weeks during which Bob refined his fix to IBM's scheduler, at the end of which he sent me a note explaining what he'd done in terms of lollipops being stuck to a wall or stored away in boxes.[152] The "lollipop fix" became rather famous, as big VCNA systems began

[152] I think that the way this all works, it gives you some flexibility in terms of specifying how many "sticky" pages a virtual machine can have and you get to specify several levels of "stickyness". (I had a discussion with Stephanie [his 5-year-old daughter] this morning about sticky lollipops.) I think there might be an interesting analogy one could develop if one imagined that an area of wall space was set aside to contain a certain number of used lollipops (referenced and reserved). There is an overflow box (MINWS) for the ones that won't fit on the wall, or to hold used lollipops that are still good but have gotten a little grubby (unreferenced). Of course, there is always the garbage can for the real grubby ones that won't fit on the wall or in the box.

The exceptions: If PTSRS is called with the VMWSNONE flag set, as page migration does, all pages are put on the trim set (the cleaning lady threw out all the lollipops). If the virtual machine made it back into Q, then there is no logical swap set and PTSRS is called with VMWSINT+VMWSNINT=0—pages are left in memory until the larger of MINWS or the reserved count is satisfied (if the lollipop user is in the room, then the overflow box is empty and they can all be on the wall). The third exception occurs when the scheduler determines that a virtual machine did not use enough virtual time while in queue to be able to reference all the pages it probably needs (VMSWSMLV in VMSWSTAT) and so none of the resident pages are placed in the trim set—they are left in memory or are logically swapped ("Don't throw anything away, I'll be back in just a second").

Well, I think I've run out of the description (and the analogy) but I hope that helps explain the interaction between RESERVED and MINWS. (I didn't plan to write this much when I started—I planned to work on cleaning off my desk.)

running into the same problem as our CICS, and IBM ultimately brought out a similar fix. Princeton did the only thing possible to thank Bob—sent him two dozen long-stemmed lollipops. A couple of years later, Bob was at it again, solving another serious problem in the IBM scheduler for three other universities. While Bob displays uncommon skill, such willingness to help other VMers has remained a very common attribute of the members of the VM community to this day.

"VM/SP—over 10K customer installations"

By the time VM/SP Release 4 came out late in 1985, there were more than 10,000 VM installations, but as the decade rolled on and SP4 was followed by SP5 and SP6, a very disturbing trend developed. For the first time ever, we saw widespread dissatisfaction with the new function being added to VM:

- CMS HELP got worse rather than better, since with every little "improvement" IBM made to it, the source became less available for us to use in alleviating its real problems.

- Although the second implementation of SNA for VM was less foreign than the first, it was not well received. In particular, customers were disappointed that the new component GCS was such a closed system, because they could see that it had great potential. In the years since, the use of GCS has expanded much more slowly that it would have, had full source been available.

- CMS Session Services ("CMS Windows") was perceived as a bad implementation of a bad idea.

- Although everybody agreed that National Language Support was a good idea, the implementation was viewed as incomplete and awkward to use.

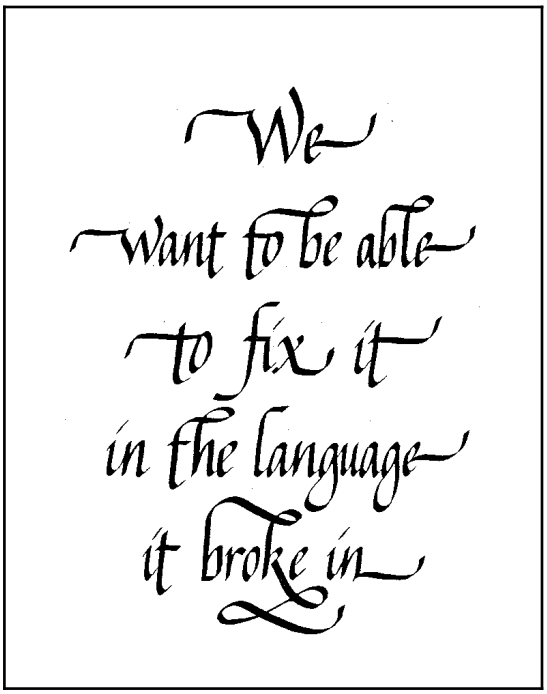
- Installations with large RSCS loads hurried to HPO Release 5 so that their systems could have more than 9900 spool files and then quickly discovered that spool access was so degraded

under HPO 5 that their RSCS virtual machines now needed to have more than 9900 spool files.

- Although the long-awaited Shared File System[153] appeared to be more promising than most of the other new function, there were concerns about its performance and the fact that it, like CMS Windows, needlessly polluted the CMS command name space, thus causing existing VM installations unnecessary migration problems.

This was all very troubling, and it resulted in normally leading-edge customers rapidly retreating to the trailing edge, many of them deciding to remain as long as possible on Release 4, which has come to be known as "the last real release of CMS", because it was the last release before the introduction of CMS Windows. I especially recommend to you the paper by Larry Chace on why Cornell University decided not to put SP5 CMS into production.[154]

Larry Chace



Equally troubling were the experiences of the installations attempting to move to XA. Throughout the decade, VM/XA steadily acquired function, although not rapidly enough to catch

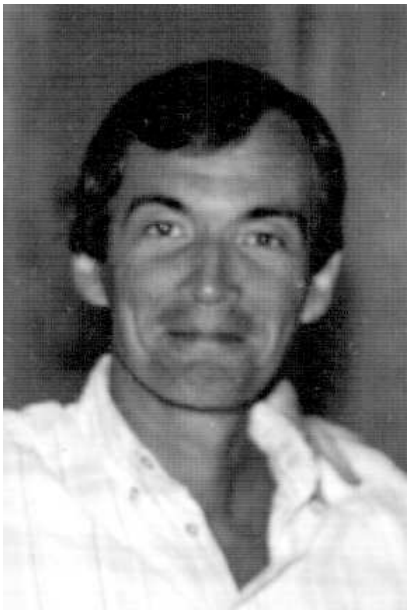
[153] "My principal 'if I could do it over again' item from the CP/67 and CMS days would have been to provide file-sharing extensions to the '370 virtual machine' early on. I don't believe that any main-line IBM software of the time had these hooks built in, so an implementation would have been one-of-a-kind. I vaguely remember in discussions toward the tail end of 1967 that to try to implement file-sharing would have increased our profile *vis-à-vis* the TSS group and made us appear to be competing with them (which we in fact were), so we didn't pursue it." (R.U. Bayles, private communication, 1989.)

[154] L. Chace, "CMS Release 5 Experiences at Cornell University", *Proceedings of SHARE 70*, March, 1988, pp. 544-552.

up with HPO. A bimodal (370 and XA) CMS was released in 1988, but has proved to be very difficult and expensive for users and software vendors to convert to.

The reluctance of software vendors to move products that require system extensions into VM/XA and bimodal CMS, which are both heavily infested with OCO modules, heightened the barriers to customer migration. The OCO policy and the new service tool in VM/XA/SP and VM/SP Release 6 dramatically increased the complexity of supporting VM systems. This exacerbated the long-standing problem of VM's growth being held down by the lack of knowledgeable VM system programmers.

As more and more of the system became object code maintained, VM installations began to encounter long, expensive delays in getting IBM to fix system bugs that the customers would have fixed themselves if they had had access to the source. Watching the XA accounts suffer through this convinced many of the rest of us not to migrate to XA before we absolutely have to.



Jerry Bozman

Ray Mansell

There have, of course, been some bright spots, too. We've all been encouraged by a number of small improvements, particularly in the areas of CMS programmability and performance. A very fruitful collaboration between Ray Mansell and Jerry Bozman gave us the minidisk directory hashing in SP4 and the faster ACCESS and FST sort in SP5, thus making the CMS file system even faster than before.

Another good innovation was the VM TCP/IP product, commonly known as "FAL" (after its product number, 5798-FAL).[155] For those of us who need to communicate with the rest of the world, FAL has been a real blessing. It is interesting to note that FAL was written by a small

[155] The MVS TCP/IP product, produced by the same group, is known as "HAL" (5735-HAL).

group of unusually talented programmers, who support their own code, and who engage in an on-going dialogue with their customers. Even better, FAL is distributed with complete source.

Early Members of the VM TCP/IP Development Group

Norm Laughlen *Jay Elinsky*
Matt Korn *Elbert Hu* *Barry Appelman* *Galina Kofman*

Late in 1989, *CMS Pipelines*, the most significant enhancement to CMS since REXX, was finally made available worldwide (after having been available in Europe since 1986).[156] Although we are still waiting for *"Pipes"* to be incorporated into the CMS base, there was general rejoicing[157] at its being released as a PRPQ.

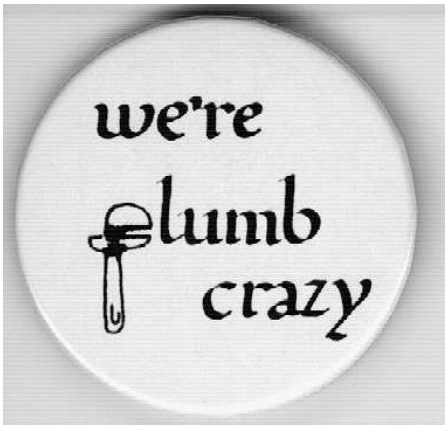
---

[156] In most of the world, the product is *CMS Pipelines* (5785-RAC). In the U.S., it is *CMS Pipelines* Programming RPQ P81059 (5799-DKF).

[157] John Lynn, of Mobil Research, a "master plumber" from his IBM days, sent the following tongue-in-cheek note to friends to express his delight at the announcement:

> Well, I'm still getting over the pleasurable surprise. Reaction to the announcement has had tumultuous effect here at Mobil. There are huge crowds of application programmers from many different Mobil divisions now pressed against the fence of the Technical Center, as they have been throughout most of the night, chanting over and over *"PIPELINE, PIPELINE"*. There have been several fires, some arrests for controlled substance abuse, and at least one birth since the crowd started to form after yesterday's announcement. The mood here has been a roller coaster of emotion. Many of the programmers storming the Center are carrying huge posters of John Hartmann, the cult's alleged leader and creator of the product. There are some camera crews moving (or trying to move) into place at the entrance to the Center, so once the massive crowd allows room, we'll get some up-close coverage and perhaps an interview with one of the crazed, blanket-clad rioters. I'll keep you posted...

John Hartmann



"We're plumb crazy"

N. But We Still Have VMSHARE

Alan Greenberg

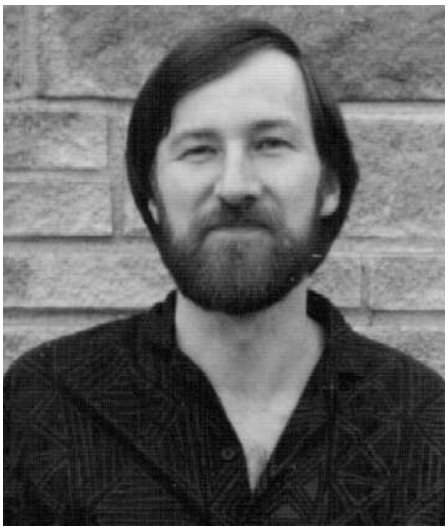
Anne-Marie Marcoux

Through the good times and the bad, VMSHARE has remained the center of the community and the place we all go to get help in solving our problems. By 1985, the time had come for us to stop imposing on TYMSHARE's generosity in supporting VMSHARE, so we reestablished VMSHARE on a self-supporting basis. This meant moving the conference from Cupertino,

California, to McGill University in Montreal. We took the conference down on the morning of August 29, and thanks to the skill of TYMSHARE's and McGill's staffs, it began running again late in the evening of the 30th, just in time to save us all from extreme withdrawal symptoms and feelings of anomie. The two people most responsible for this feat were Alan Greenberg, the Director of McGill's Computing Centre, and McGill's charming and very skillful system programmer, Anne-Marie Marcoux.

In 1986, Steve Howes, of Brigham Young University, made using VMSHARE easier for many of us when he built a server that made VMSHARE accessible *via* electronic mail from most of the world's academic and research networks, including BITNET, VNET, and the Internet.

Steve Howes



Tony Harminc

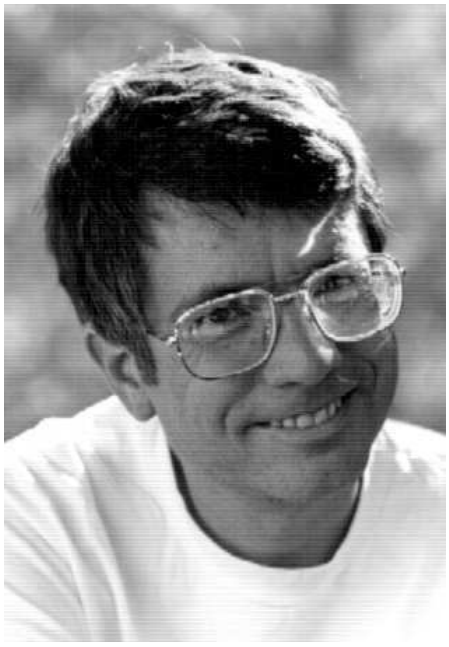
Thanks to the good folks at McGill and to our wonderful VMSHARE administrators, Tony Harminc and Dick Rawson, VMSHARE has managed to pay its way[158] and to grow rapidly since the move to McGill, and we are now able to offer membership in the conference to installations that belong to any of the eight major IBM user groups.[159]

---

[158] Dave Smith's original EXEC implementation of the conferencing software was replaced in 1982 by Charles Daney's PL/I implementation. The Adesse Corporation later acquired Charles' code and now markets it under the name CONTACT. Adesse has very generously provided VMSHARE with a free copy of CONTACT.

[159] SHARE, GUIDE International, COMMON, SHARE Europe (SEAS), G.U.I.D.E. (Guide Europe), ASG (Australasian SHARE/GUIDE), GLA (GUIDE Latin American), and JGS (Japan GUIDE/SHARE).

We have recently begun enjoying the participation of members of Australasian SHARE/GUIDE. ASG's VM Project Manager, Neale Ferguson, has been encouraging ASG participation in VMSHARE with the same enthusiasm that Dave Smith and Jeff Gribbin showed in getting SHARE and SEAS members started years ago.



Dick Rawson

Neale Ferguson

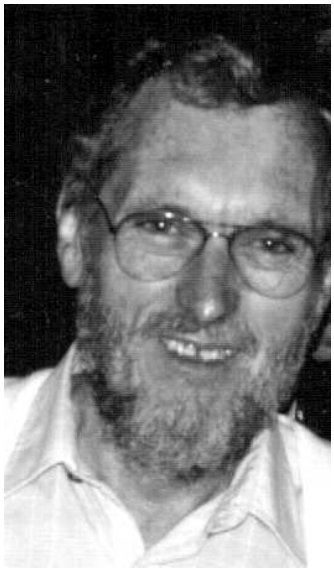
We have also been cheered by a recent resurgence in IBM participation in the conference, after a few years during which IBMers on the conference were seen but not heard.

I hope that those of you who don't yet use VMSHARE will soon join us on the conference. I have no doubt that you and your employers will find that your time there is well spent.

O. A few more pictures

I regret that I have had to leave so many worthy people out of this account of VM's history. To compensate a bit for that, I'd like to show you pictures of a few more members of the VM community, many of whom you'll encounter on VMSHARE:

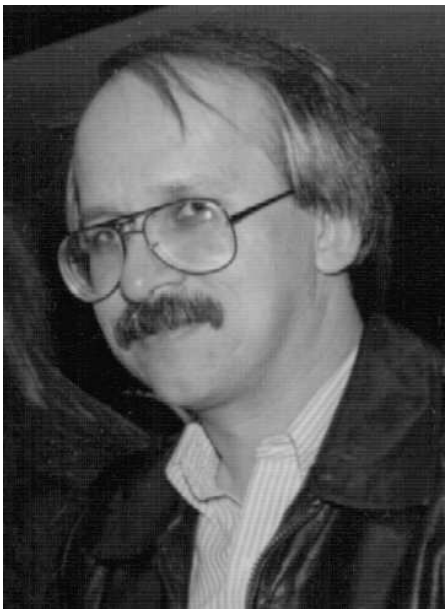
Mike Armstrong,
SHARE Past President



Ted Johnston,
SHARE VM Group Deputy Manager

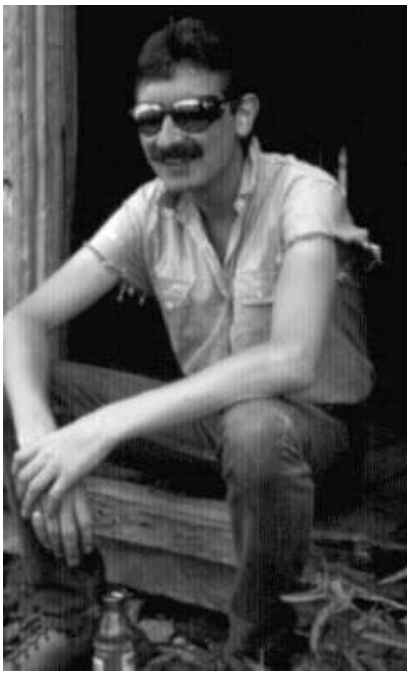
Aron Eisenpress, SHARE requirements coordinator,
and Jean Rall, SHARE VM Group Deputy Manager

Herb Weiner,
author of Cornell COMPARE

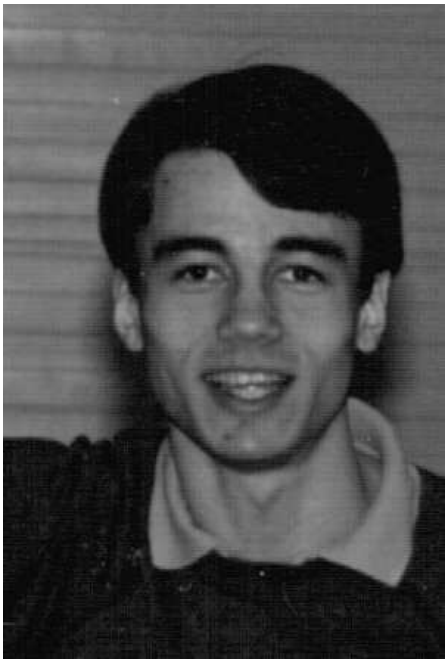


Tom Klensk,
long-time RSCS developer

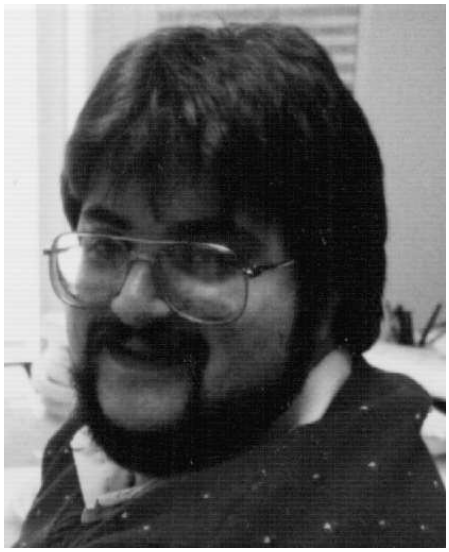
Ira Bland and Pam Swope Bland,
dynamic IBM representatives to SHARE and GUIDE



Arty Ecock,
author of SESSION



Eric Thomas,
our current enfant terrible



Phil Smith, III, aka *"phsiii",*
another of our technical heavies

Peter Koeppel, much-loved
performance measurement guru

Virginia Hetrick,
graphics guru
and SHARE Fellow

Bob Bolch,
CMS guru and defender of
the integrity of the VM spool

Charlie Whitman,
one of the greatest of the CMS gurus

TSO,
Charlie's late pig

Chris Stephenson, author of EXEC2 and the experimental system EM/YMS

Joyce Tomaselli, cherished IBM representative to SHARE

Jacques Myon, founder of the Canada VM Users Group

Adrian Walmsley, indispensable IBM representative to the SEAS VM Project

Sally Patterson, VM Developer since the Burlington days

Yon Bard, author of VM PPF

IV. WHAT WENT WRONG?

By the time we celebrated VM/370's fifth birthday in 1977, we were finally able to begin hoping that IBM would listen to us and not kill VM. A few weeks later, the director of IBM's Poughkeepsie lab came to SHARE to listen to our *Why VM?* presentation, and shortly after that IBM began to commit money to VM's future.

By the time we celebrated VM's tenth birthday at SHARE 59 in New Orleans in 1982, IBM had declared VM strategic, and the number of licenses was growing wildly. Our installations had already begun to enjoy some fruits of IBM's new commitment to VM, such as Xedit, the enhanced file system, and Pass-thru. Although IBM had been to SHARE to discuss the possibility of distributing only object code for its software, the general view was that they'd soon realize how unwise that was. In 1982, the VM community had a lot to celebrate. Most of us believed that CMS was about to take over the world, so we gave it a wonderful birthday party.

For VM's fifteenth birthday, in 1987, we had another grand birthday party, at SHARE 69 in Chicago. There were two huge birthday cakes, and several of the most honored old-time VMers gave delightful talks full of funny stories about our long-past struggles: Ed Haskell, Romney White, Pat Ryall, Bruce Marshall, Stu Toledano, Dick Newson, Lyn Hadley, and Jerry DePass.

But underneath these festivities, there was an air of unease. OCO had become real and loomed large. IBM had begun delivering truly unpalatable new function in VM, so that at that SHARE, for the first time ever, we were trying to find out whether we had a mechanism for asking IBM to delete new function. Five years after MVS customers got an Extended Architecture (XA) version of MVS, we still didn't have a full-function VM/XA. We were seeing some of the best people in VM, such as Dave Smith, Lynn Wheeler and Bob O'Hara, leave to work on other systems. At the end of the ceremony, one of the recent VM Group Managers, Sandra Hassenplug, turned to me and asked, "Will the twentieth birthday be a wake?"

"VM Soars with 20,000 Customers!"

Bob O'Hara

VM today is a huge success. There are now more than 20,000 VM licenses, and that number is still growing, but even those of us who love CMS unquestioningly no longer believe that it will take over the world. The reality today is that IBM has had to go hat-in-hand to AT&T to license UNIX, while nobody is coming to IBM to license CMS. All of us loyal VMers know that UNIX is ugly and cryptic and hard to use compared to CMS, and the lack of security in UNIX systems is appalling. So, why then has CMS not taken over the world? What mistakes did we make? What mistakes did IBM make?

There are many answers to these questions, but I would like to discuss briefly the ones that appear to me to be the most significant.

A. IBM made the PL/S language proprietary.

When IBM first developed its system language PL/S (the predecessor of today's PL/AS), it made an important mistake. It decided that the language would be proprietary and that no compiler would be released. It even threatened legal action against a customer who produced a PL/S compiler. SHARE members reacted so strongly to IBM's decision not to release the compiler that the period of the early 1970's is remembered as "The PL/S War", but IBM stuck with its decision.

By not making PL/S and PL/AS available, IBM guaranteed several unfortunate results:

1. **The evolution of PL/S (and later PL/AS) lagged so badly that they remained "trailing-edge" system languages.**

   IBM is a very product-oriented organization, which makes it hard to justify investing money in software that is not a product, even when it is an important software development tool, such as PL/[A]S. Because PL/[A]S was not a product, there were no user group requirements for improving it, and support was generally poor. Even today, for example, PL/AS doesn't have a decent supported subroutine library. If the compiler had been made a product, IBM would have ended up with a far more powerful system language for its own use.

2. **Developers resisted using PL/[A]S to avoid alienating their customers.**

   Many of the development groups, including VM's, didn't use PL/[A]S for a long time, not only because of the problems with the language and the compiler, but also because doing so would prevent customers from extending their products, and the developers understood how damaging that would be. In the case of VM, the developers also understood that their writing in PL/[A]S would mean that the users could no longer use VM's elegant source maintenance facilities.

   While the VM community is grateful to the developers for having held out against PL/AS as long as possible, VM would be a far more successful system today if the developers had been able to use a *good* higher-level system language that was also available to their customers. They would have been able to produce more function, faster, with fewer bugs.

3. **IBM systems remained non-portable.**

   One of Corbató's most startling innovations was to write Multics in PL/I. That was a controversial decision at the time, because the ink was barely dry on the specifications for the PL/I language and nobody had a PL/I compiler. Besides that, nobody really had a machine

large enough or fast enough to run a system written in a higher-level language. However, there were also many good arguments in favor of using a higher-level language, and the authors of UNIX understood those arguments. Although UNIX was originally written in assembler language, it was translated into the C language in 1973. At that moment, CMS, by default, lost the race to be the world's time-sharing system. UNIX was portable, while CMS was not. Even if CMS had had no other handicap in the race, that one would have been sufficient.

It may be that one of IBM's purposes in making PL/S proprietary was to make sure its systems would *not* be portable, or it may be that the advantages of portability were just not apparent in those days. At any rate, IBM now finds itself in possession of some highly non-portable systems.

Today, UNIX can be moved easily to any new processor that comes on the market. All that is needed is another C compiler; then the license fees start rolling in to AT&T. IBM's assembler language systems, on the other hand, are not portable even to new IBM architectures. Its PL/AS systems are also not easily portable to other IBM hardware because PL/AS is still too low-level a system language. Thus, while AT&T can license UNIX to the world, IBM must largely rewrite its systems even to move them to a slightly different architecture of its own.

It is worth pointing out that C is an SAA language, while PL/AS is not. IBM has been forced to learn AT&T's system language because it refused to allow the world to learn its own.

B. The users lost control of the product.

For much of VM's life, the ties between VM Development and the users, especially the internal users, were close enough that the users had a great deal of influence on the direction of the product. This seems not to have been so true in the past several years (although I believe we have seen a reversal of this trend recently). My impression is that the relationship between Development and the VM community broke down somewhere about the SP4 period, as the result of two events:

1. **The planners were seduced by the dream that the 9370s would bring in licenses numbered in the hundreds of thousands.**

   Believing in that dream greatly changed their priorities and strategies. It freed them from having to keep new releases compatible enough that their tens of thousands of existing customers could migrate without a major struggle. And it freed them from having to care what the old VM hands inside and outside the company thought about their new releases.

   Isolated from the users and relying primarily on their own ideas of what the system should be, they began producing incompatible new function that was often unattractive to existing customers. In several cases, particularly undesirable functions, such as CMS Windows, were released over the strong protests of knowledgeable VM people inside the company.

2. **Management decided to make VM Development "self-sufficient", *i.e.*, no longer to bring in new function from outside Development.**

   I believe that it's fair to say that up through VM/SP Release 4, most new function that really mattered to customers originated outside Development. On occasion, customer code was picked up in the product. That was always very rare, but even quite recently IBM has

acknowledged cases of adopting customer ideas to improve the system, such as the CMS file system read-ahead work that Dick Johnson[160] did while at SLAC. Much more commonly, new VM function began as local modifications to IBM's internal VM systems, written by local system programmers. In other cases, new function began as the work of internal end users or of IBM SEs in the field.

Dick Johnson

The decision for Development to become "self-sufficient" had two totally predictable results:

- *Loss of Continuity:* Continuity of VM tradition was lost when Development's "head count" grew enormously, more than quadrupling in a very short time. Quadrupling the number of VM developers resulted in an influx of many, many people with no VM background and no particular commitment to VM. They began developing code that felt "un-CMS-like" to the end users for the simple reason that the new developers didn't know CMS well enough.

- *Untried Function:* Development began delivering a great deal of very raw, untried function. Before "self-sufficiency", back when Development was picking up function that had been prototyped on internal systems, there was always a good chance of their getting what I described earlier as "really good software". The function they got this way had been tried out in at least one real system, where it had been successful enough to attract their attention. It had generally been created by one skilled person who wrote it because he really needed it for his own users. The author had been free to rework his design and code until they were right and had then generally shared his code with other sites and had incorporated their feedback to improve it.

  Absolutely new function added by Development is much less likely to be "really good software", because almost none of the conditions under which such software comes about exist in a large development lab. Because developers generally have little experience of

[160] Now with IBM.

the "real world", new Development function can hardly be expected to be anything more than an approximation to what users really need. It will certainly be raw and incomplete, because it won't have been honed by being subjected to use by thousands of real end users. The developers won't have been allowed to use an iterative approach in designing the function; they'll have had only one chance to guess how the system should work.

C. IBM wasn't trying to build a system that would take over the world.

While we were all dreaming that CMS would take over the world, IBM's plans were much less grand. Its goals were much too short-term to lead to such a successful outcome. IBM was simply trying to support Endicott and Poughkeepsie processors as inexpensively as possible. It was in the hardware business, after all. No one foresaw that IBM would ultimately lose prestige and business because its foremost time-sharing system was not truly world-class, so there was no commitment to making it so.

Even if such a far-sighted goal had been established, there are structural problems in IBM and shortcomings in its development process that would almost certainly have prevented the attainment of that goal:

1. IBM's development process discourages innovation (or even good housekeeping).

UNIX has always had an important advantage over CMS, which is that it was not a captive of IBM's development process, where it appears to cost far more to justify a line of code than it does to write one. For most of its life, UNIX has been developed by a small number of extremely good programmers who were free to add function just because it was a good idea, who were free to restructure the system when it had grown enough to need restructuring, who were free to innovate in any way they wished. The result of this is that although UNIX's user interface is something only a computer scientist could love, its underlying structure and function are elegant and very sound.

CMS, on the other hand, has a much more attractive user interface, but under the covers is a shambles. CMS has become such a pastiche structurally that it is inflexible and costly to extend. Furthermore, CMS still has gaping holes, where needed function is missing, and ugly blemishes of really inferior function. For example, the CMS SORT command is little improved since Jim March wrote it to use once that night in the 1960's.[161]

As one of the VM Group's wags has put it, "the livingroom is still full of lawn furniture".[162]

---

[161] IBM does sell separately a replacement for the CMS SORT command.

[162] The "lawn furniture" analogy for CMS's weak spots is derived from an append to the VMSHARE file MEMO VMSP5, by Val Breault, of General Motors Research, who was discussing CMS Windows:

> It strikes me as somewhat arrogant and thoughtless. Consider... How would your intentions be perceived if you were to surprise your wife with ghastly livingroom wallpaper and clashing drapes when for seven years she had been begging for real livingroom furniture to replace the lawn chairs you had been using? Would you *really* expect to get away with it by saying the furniture is a Future Objective, as

In UNIX, such horrors as CMS SORT, CMS DEBUG, and CMS COMPARE would long ago have been laughed out of existence, to be replaced by elegant, very high-tech commands contributed by skilled users. Because there are no explicit marketing requirements to remove such blemishes from CMS, they remain to embarrass us all.

2. IBM ignored the technical lessons it should have learned from TSS.

TSS, the Time-Sharing System, was the best operating system IBM ever wrote, but it was very definitely not a commercial success. It was simply too big for the machines of its day, which were not much different from an early IBM PC in memory size and I/O capacity. And, because TSS did not provide OS simulation, compilers and other applications had to be rewritten to run under it, thus greatly increasing the cost of TSS to both IBM and customers. However, TSS was superb technically and introduced many desirable innovations that VM still doesn't have after all this time. To mention a few of the features of TSS:

- An integrated shared file system,
- A very powerful symbolic debugger,
- A real session manager,
- Subtasking,
- 32-bit addressing,
- N-way multiprocessing,
- Device independence in the file system (which provided a page-oriented view of files),
- Ability to map files into memory,
- Virtualization of both program and supervisor state (which provided supervisor integrity in the virtual machine), and
- A dynamic linking loader that could load into shared memory.

TSS was also a good example of a two-level operating system, with a tight, well-defined kernel. Despite all these wonders, TSS's size and early instability were enough of a problem that the far less powerful CMS had a chance to come to the rescue and is the system that survived.

Over the years, it has become apparent to me that one of IBM's greatest weaknesses is its short "corporate memory", which is primarily the result of people changing jobs within IBM so often. This causes hard-won lessons to be forgotten very quickly. Thus, I suspect that today very few of the VM designers and developers have ever heard of TSS or understand that they could learn a great deal by studying it.

3. The VM community inside IBM was splintered by parochialism.

"Parochialism" is the nicest word I can think of to use. "Not Invented Here" may describe the problem more graphically. The prevalence of this attitude among IBM's VMers has time after time led the developers to re-invent the flat tire when there was a shiny magnesium wheel available for free. Throughout the company throughout the past two decades, VMers have split into factions that were unwilling to acknowledge the good work done by others. There have been splits between Research and the Scientific Centers, between different parts of Research, between Domestic and World Trade, between the guys at this end of the hall and those fools down at the other end. The VM developers, especially, have tended to become isolated from the rest of the VM community, in the company and in the world.

---

you had been saying since 1982?

Another terribly obvious aspect of this parochialism was the open warfare between the Endicott and Kingston[163] development labs, which did the system great damage over the years, as each site tended to allow the goal of supporting its own hardware to override concerns about the welfare of the system (or the company) as a whole. We can hope that the recent reorganization of VM Development will induce a spirit of cooperation, but when we see such silliness as SP6 CMS committing suicide when IPLed under an XA CP, we can hardly be surprised that CMS hasn't taken over the world.

D. IBM failed to support VM properly because it was always either hoping or fearing that VM would go away.

Because IBM took so long to make a commitment to VM, VM has been "playing catch-up" forever. Because IBM didn't really believe in VM, the resources required to produce a viable XA VM were withheld for far too long. For far too long, IBM's hardware developers were designing their high-end processors with only MVS in mind, so that VM performed less well than it should have on those processors.

Chris Thomas

IBM's policy for supporting new devices on VM is probably best described as "always eventually more-or-less supporting most new hardware". We are not surprised when we see VM "support" a device only for attachment to a guest. We are not surprised when we see support that should be virtualized not being virtualized. All in all, we've learned to be grateful if we get what Chris Thomas, of UCLA, calls "same century device support".

---

[163] The Poughkeepsie VM Development group was moved to Kingston, due to space considerations, in 1982.

Now, after all the years we spent convincing IBM to make VM strategic, after all VM's phenomenal growth, after all the revenue it has generated and all the accounts it has kept "blue", IBM appears once more to be questioning whether VM has a future. Though the rumor of VM's demise is greatly exaggerated, there is a danger that it could become a self-fulfilling prophecy.

E. OCO stifled innovation.

Looking back at VM's early years, Les Comeau evaluated the factors that led to VM's success as follows:

> The success of the CP/CMS system certainly is in no small way attributable to its friendly and forgiving user interface. A second contributor was the clean separation of function in the CP product, which made it easy for the sophisticated user population to remove, add, and substitute functions supplied by IBM, thereby greatly expanding the talent working on enriching the system.[164]

The Object Code Only policy has greatly *contracted* the talent working on enriching the system. Since that policy was promulgated in 1983, innovation in VM has been sharply curtailed. This must be acknowledged as a major factor in VM's current malaise.

At my own installation, I have had to learn to say "no" when management has asked for function that would require system modifications. It is better to refuse to give our users function they want than it would be to give it to them now, knowing that we will have to take it away from them two or three years from now. Fortunately for our users, we often have a third alternative: to implement the desired function on a non-IBM system.

---

[164] Comeau, *op. cit.*, pp. 45-46.

V. CAN VM BE SAVED?

I find it interesting to speculate on what might have happened if the VM community had done what the UNIX community did—established an alternate version of the system. What if there had been, say, a "Cornell CMS", just as there is a Berkeley UNIX?[165] I suppose the most obvious answer to this question is that IBM's lawyers would have gone into a feeding frenzy. But, what if we had been able to carry it off anyway, including getting suitable funding, as Berkeley did?

A large number of VM installations have always been far ahead of IBM in providing innovative function for VM. For example, by the mid-1970's, the TYMSHARE staff had added function to CP and CMS that we are still waiting for today from IBM. National CSS and Interactive Data Corporation added extremely useful and professional function, as did many universities, government agencies, and corporations. Although much of this function was shared *via* the Waterloo Tape, much of it was not, and much of it ultimately had to be abandoned because it was too expensive for one installation to carry from release to release. But a VM group comparable in size and skill to Berkeley's UNIX group, such as the system programming staff at Cornell, could, with appropriate funding and the support of the community, have built the best of the customer modifications into a very good and very powerful extension to IBM's system.

Killer Croquet at Cornell
Larry Chace, Cecilia Cowles, Larry Brenner, Ben Schwarz

We would not, of course, have been able to draw upon local modifications made by IBM's system programmers, but it seems likely that competition from us would have persuaded the IBM developers to draw upon them more. We might now be seeing the merger of the features of the two systems, in the same way that UNIX System V is picking up features of Berkeley UNIX.

---

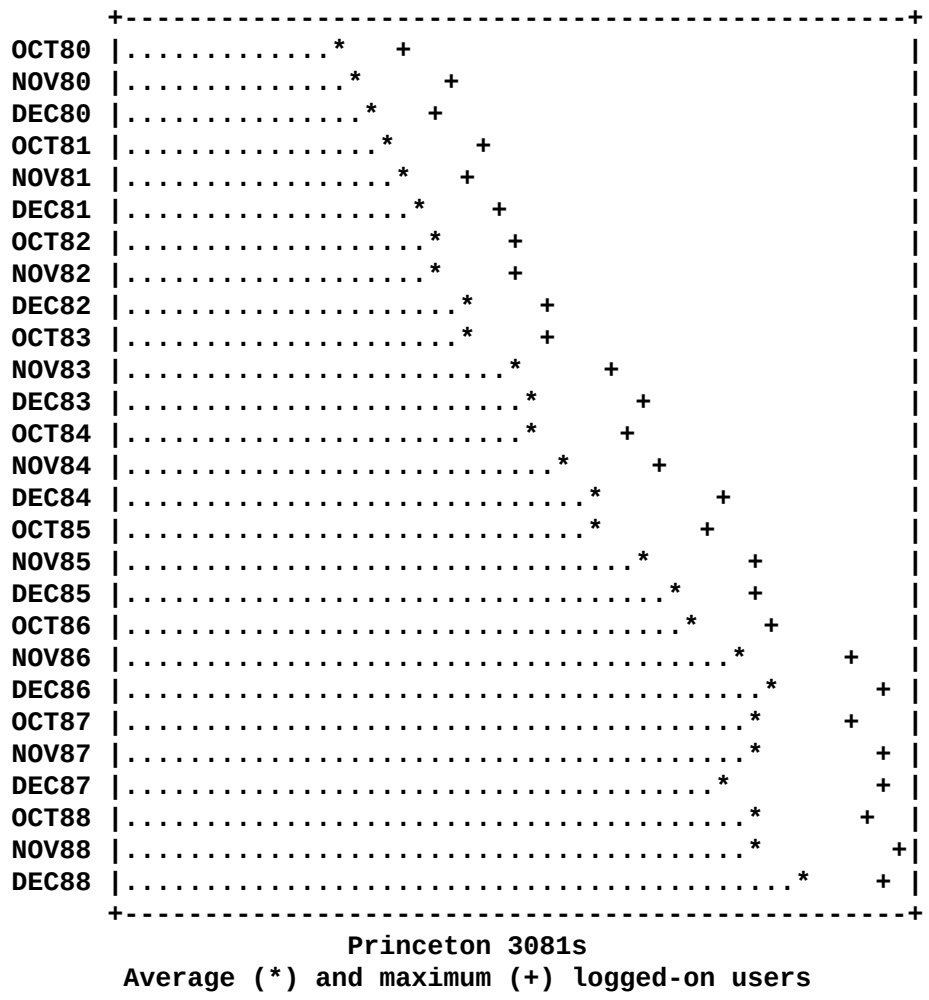
[165] I suggest Cornell here because it has long been a remarkable treasury of VM talent, although several other universities and research centers have also been outstanding, among them the University of Waterloo, Brown University, the University of Maine, the Stanford Linear Accelerator Center, and the University of Liverpool.

The result would surely have been a more robust CMS, with richer function and far fewer blemishes than today's CMS.

Why didn't we do it? Well, of course, there were all those lawyers. Then, too, we had a long tradition of working *with* IBM to build the system we needed. That had always worked very well, both for us and for IBM. Unfortunately, it took us too long to recognize that it had stopped working.

What can we do about VM's problems now?

I think it's too late to set up an alternate system. It appears to me that the only chance VM has now is for IBM to recognize that it has a serious software development problem. One can readily see the effects of this problem at Princeton; one need only look at the trend in CMS usage on our 3081s:



Princeton's CMS user load essentially stopped growing a few years ago,[166] yet, during those same years our total expenditures for computing grew at an unprecedented rate. If one recognizes that installations such as Princeton tend to be the bellwethers, it becomes clear that IBM has a problem. I contend that this problem is not merely a VM problem, but exists across IBM's mainstream software in general. It is more obvious in VM than elsewhere, because VM has more competition from other forms of computing than much of IBM's other software has.

So, we must persuade IBM to look at where its most successful software comes from. I am confident that it will find that its systems and components that have experienced explosive growth were the work of small numbers of unusually talented programmers working with little management interference and free to "hack at" their software, iterating the design based upon user feedback. If it then compares those conditions with the conditions prevailing in its large development labs, I think it will understand why so many people feel that IBM's current approach to software development is guaranteed *not* to produce real innovation.

What can IBM do to save VM?

I hesitate to answer this question, because IBM has heard my answers, from people much more prestigious than I am, many times before to little avail, but let me try:

1. *IBM should maximize on a longer time frame and across a broader scope.*

   The object of operating system development must be to build and sell a really good (and, thus, really competitive) operating system, rather than merely to produce the minimum amount of software necessary to "support" specific hardware. Technical excellence does pay off in the long-run. Too many makeshift solutions ultimately will cause any system to collapse under its own weight.

2. *IBM should make it easier for innovation to happen and easier for innovative software to reach its customers.*

   Currently, the development labs devote most of their attention to supporting new hardware and answering customer requirements. Neither of these is likely to lead to the sort of innovation that will greatly expand the uses of IBM's computers. Although the user groups try to make their requirements long-range, in fact, they almost always address improvements to existing function, rather than asking for entirely new function. Until a new function exists, few people will realize how badly they need it—how many of us knew we needed a spreadsheet program before spreadsheet programs existed?

   IBMers inside and outside the development labs continue to create wonderfully useful and innovative software, but the barriers to getting their software out into the real world are so formidable that few of them manage to do it, and even fewer attempt it again after their first try.

   I find myself hoping that somewhere inside IBM right now there is a hacker building a wonderful interface between CMS and workstations that will save CMS. I can't describe that interface to you, so I haven't tried to write a requirement for it, but I'll know it when I see it, if

---

[166] The slight increases in the logged-on user count late in 1988 can be accounted for by the large number of service virtual machines required by TCP/IP for VM.

that poor hacker has the stamina to get it out the door.

3. *IBM should transfuse some excitement into CMS as soon as possible.*

   If I were a powerful IBMer who wished to prove to the world that CMS has a future, I would take immediate steps to make CMS more competitive. First, I would survey the CMS "power users" within IBM to find out what they consider to be the twenty most useful internal-use-only CMS tools or applications. I would then make those twenty a part of CMS as soon as possible, without regard for whether they are "strategic" and without regard for whether they meet coding or documentation standards. At the same time, I would incorporate the *CMS Pipelines* and LEXX products into CMS. I suggest releasing all these functions as an integral part of CMS, because it is only when a function is available to all CMS users that its use can grow explosively. Until toolmakers and software vendors know that they can count on the availability of a function at most VM installations, they won't exploit it.

4. *IBM should recognize that it cannot afford the Object Code Only policy.*

   Looking back over the past decade, it is now clear that the winning strategy for the Eighties was open systems. Unfortunately for us all, IBM spent that decade moving as quickly as it could in the opposite direction. The time has come for IBM to re-examine this policy.

5. *IBM should manage its programmers more wisely.*

   It often appears that much of IBM's management believes that programmers are interchangeable. It often appears that they have no concept of how long it takes for even a talented programmer to "get up to speed" in a new system. It often appears that they are unconcerned about the loss of their best technical people, that they don't understand that their losing even one highly skilled programmer can impact their customers.

   It often appears that IBM's management is unaware of the diseconomies of scale in programming. IBM has been told so many times that the "Mongolian Hordes" approach to software development never, ever works well that I hardly dare say it again. But, I firmly believe that there are far too many people in VM Development today and that they work in much too structured an environment. Perhaps, as a first step, it would be best simply to suggest that every IBM manager responsible in any way for software development should be required to read *The Mythical Man-Month* once a year.

   I would like to believe that every VM development manager is familiar with a comment Les Comeau made about the success of VM:

   > It would be extremely gratifying to attribute that success to brilliant design decisions early on in the program, but, upon reflection, the real element of success of this product was that it was not hampered by an abundance of resources, either manpower or computer power.[167]

[167] Comeau, *op. cit.*, p. 38.

Should VM be saved?

It appears that no matter what the problem, this year's solution inside IBM is "AIX". But AIX also appears to be far from practical for supporting large and complex installations. More importantly, many of us who have watched VM being smothered would hesitate to commit our shops to another IBM system, because we know that the same thing will happen to it, should it also become strategic. It appears to me that unless IBM management makes some major changes in its approach to software development, all IBM software, no matter how brilliant and innovative, will eventually get pulled down into mediocrity.

Should VM be saved? Yes!

CP remains the most flexible way to support testing and migration of IBM systems and the only way to partition the resources allocated to different uses dynamically over the day, month, and year. CMS still has an unusually friendly user interface, among mainframe systems, and remains a very powerful tool for automating computer services and operation.[168]

Can VM be saved?

Let me echo Robert Fisher's blunt words of more than a decade ago:

Can VM be saved?

Yes, VM can be saved, if IBM wants it to.

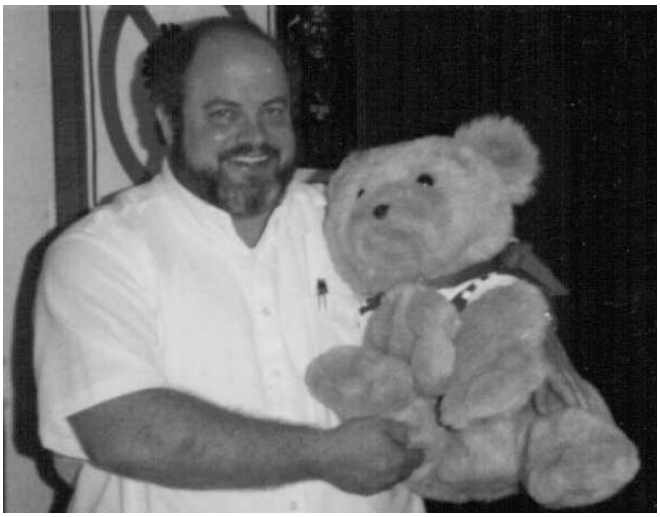
---

[168] For further discussion of CMS's strengths, I refer you to excellent papers by two of VM's brightest stars, Pat Ryall and Stuart McRae: P.R. Ryall, *Why CMS?*, Report RC 13932, IBM Research Division, Yorktown Heights, NY, and S.J. McRae, *CMS and UNIX: What They Can Learn from Each Other*, Document SJM-84.1, Systems & Telecoms Limited, Phoenix House, 1 Station Hill, Reading, Berkshire, RG1 1NB, United Kingdom.

VI. CODA

To end on a positive note, I'll go back to "The light at the end of the tunnel".

First, though, you must meet Kent Taylor, the VM Group Poet. Kent is one of the most widely loved people in the VM community and also one of the most creative. And, Kent doesn't mind being called a "VM Bigot". One of his most famous statements is, "TSO may be slow, but it sure is hard to use."



Kent Taylor

One Sunday evening at the beginning of a SHARE many years ago, we were all surprised to see an IBMer from Yorktown selling these buttons:

"The Light at the End of the Tunnel"

One can understand how frustrated he must have been, being a fan of TSO at Yorktown, and he expressed that frustration very artistically. However, he hadn't counted on Kent, so he didn't understand why VM people were standing in line to buy his button and then coming back for more. (He finally started refusing to sell his button to anyone he suspected of being a VM bigot.)

What Kent had done, you see, was to figure out the true significance of the button:

- Although CMS is, admittedly, in trouble here, anyone can see that it is out in front of TSO.
- More importantly, with CMS this typical dumb user is able to support himself, using very few resources, while with TSO he would need that mammoth expensive machine and a large staff.
- You will note, too, that TSO is shedding absolutely no light on the user.

VM is, admittedly, in trouble today, but it is still the best thing that IBM has going for it, so I hope that IBM and its customers **working together** will be able to set things aright.

Thank you for your attention.[169]

[169] The preparation of the printed text of this paper has been an exercise to demonstrate that CMS lives (although it needed a bit of help here and there from PC/DOS and UNIX). The text was prepared using Xedit and Waterloo Script. The photographs were scanned using "SCAN-IT" (from Howtek, Inc.) and the Howtek Scanmaster (manufactured by the Sharp Co., as the Sharp JX-450 Color Scanner). SCAN-IT was running on an AT&T 6310, using the AT&T Targa 16 system and a Sony Trinitron monitor. The image files were then ftp'ed through the University's networks to VM running on an IBM 3081. (This process made use of a Western Digital "EtherCard PLUS" and "PC/TCP" from FTP Software, Inc., as well as IBM's VM TCP/IP product.) On the 3081, the image files were converted from Howtek's "SIM" ("ScanIMage") format to "Encapsulated PostScript" using the "SIM2PS" program written by my colleague David Laur. (PostScript is a trademark of Adobe Corporation.) After formatting by Script, the file was printed on an Apple LaserWriter using a CMS LPR command written by my colleague Robert Knight.

I wish particularly to thank my colleagues Emily Heine and Toby Paff for their assistance in conquering Script and the staff of the Princeton Interactive Computer Graphics Laboratory (Kirk Alexander, David Laur, Brad Gianulis, and Alexandra Shulzycki) for many hours of assistance in capturing and improving the images. I am also grateful to Mike Cowlishaw for his advice on image enhancement and to John Hartmann for dramatically speeding up my image enhancement program.

Appendix A

CP/40—THE ORIGIN OF VM/370

L.W. Comeau[170]

Introduction

Perhaps what is significant is not the ability to sit down and plan invention, but the ability to recognize innovation when it occurs spontaneously. Such was the case with a software system called CP/40. Originally planned as a measurement tool, it has grown to become an equal partner with IBM's two other operating systems, DOS/VSE and MVS.

It is further interesting to note that this stepchild of the time-sharing community now enjoys more popularity than its well-funded contemporaries, IBM's TSS and GE's (Honeywell) Multics. It would be extremely gratifying to attribute that success to brilliant design decisions early on in the program, but, upon reflection, the real element of success of this product was that it was not hampered by an abundance of resources, either manpower or computer power.

An early consultant's report rated the modest goals of the CP project and claimed it was not price competitive with the larger systems, the IBM 360/67 and the GE 645. In a follow-up look, the same consultant said that the obvious error in his previous work was to accept the claims of the larger systems versus what was achievable. The CP/40 system met its goal of 15 users; the larger systems were incapable of supporting hundreds of terminals, which was their design point.

CP/CMS Team

To understand the rationale for CP/40, it is necessary to appreciate the backgrounds of its designers. There were five major contributors to the design of this system—Bob Adair, Dick Bayles, Bob Creasy, John Harmon, and myself. Of this group, three were programmers and the remaining, Harmon and Comeau, were systems engineers. None were neophytes in the time-sharing world. Bob Adair was involved in command and control systems being built by the MITRE Corporation, and the remaining four had been associated with the MIT Computation Center in the years preceding the CP design.

The MIT association, in particular the goals of and experience with Professor Corbató's CTSS system, strongly influenced the design of the terminal user interface for CMS. In fact, the emphasis on "user friendly" interfaces brought on by the success of Wang in the office systems area is hardly "new news" to this group. Those at MIT in the early 1960's put a strong emphasis on the needs of the non-computer-oriented professional.

Indeed, the artificial separation of word processing from data processing that occurred in the office system market comes as a complete surprise to the people who have been VM users over the years. Word processing has always been available to the VM group, first in the form of

---

[170] This paper is reproduced from the *Proceedings of SEAS AM82*, September, 1982, with the permission of SHARE Europe and L.W. Comeau.

editors and print formatters and, later, incorporating the advantages of electronic mail. This, too, was part of the MIT philosophy; that is, to be of any use a system should provide the basis for its own evolvement, including the requirement for documentation.

The CP/CMS team was formed as part of the Cambridge Scientific Center, which was established by IBM management in 1964 to provide a center for competency in time-sharing. Although IBM had worked with terminal systems, such as the Sabre, the American Airlines on-line reservations system, it was felt that the general purpose time-sharing environment was significantly unique so as to require a separate research.

Baseline

In the six months immediately preceding the design of CP-40, the Cambridge Scientific Center had been involved in preparing two proposals for time-sharing systems, one to Project MAC at MIT, which IBM lost to the GE 645 Multics system, and one to MIT's Lincoln Lab, which was a winning bid and was the genesis of the System/360 Model 67 and the TSS operating system.

The major technological change proposed for these systems was virtual memory. It was felt that this offered a solution to both the programmer productivity constraint and to the performance problems faced by earlier time-sharing efforts. It was widely accepted in the early 1960's that the cost of producing programs rose exponentially as one approached the memory limit of a particular machine. Virtual memory, by releasing the programmer from this constraint, would obviously lower the cost and time to produce a large application.

Since the early time-sharing experiments used base and limit registers for relocation, they had to roll in and roll out entire programs when switching users. There was some talk of the "onion-skin" technique, where small programs would displace only parts of large programs when users were swapped, but to my knowledge it was never implemented. Virtual memory, with its paging technique, was expected to reduce significantly the time spent waiting for an exchange of user programs.

What was most significant to the CP-40 team was that the commitment to virtual memory was backed with no successful experience. A system of that period that had implemented virtual memory was the Ferranti Atlas computer, and that was known not to be working well. What was frightening is that nobody who was setting this virtual memory direction at IBM knew why Atlas didn't work.

Similarly, the functions to be provided in the end-user terminal were the subject of debate in the mid-1960's. There was a requirement to provide application programs with each single character as it was typed. It was thought that with that capability errors could be turned around instantaneously, and the system could thereby save the user retyping time and effort. This requirement exists today in the "raw mode" available to programs written for the UNIX system.*

The requirement for upper and lower case received strong emphasis then. Today, of course, one finds few terminals without that capability, but there was little appreciation for that function amongst terminal builders of the early 1960's. Terminals were viewed as strictly data-entry devices.

---

* UNIX is a trademark of AT&T Bell Laboratories.

A third terminal requirement generated in this area was the ability to turn off printing in order to suppress printing when the user entered authorization codes. We take it for granted now, but most terminal manufacturers did not include this function in their early models.

It was against this background that the Cambridge Scientific Center undertook the CP/40 project. It was our intent to study programs and programmers in a time-sliced virtual memory environment.

Vehicle

The 360/40 was chosen as the vehicle on which to implement the Scientific Center's time-sharing experiment. This was not the result of extensive load analysis but because of a lack of availability of a 360/50, which was thought to have the CPU power required to support our interactive population. It turned out to be fortuitous, because the modifications required to segment the memory for virtual addressing were easily accomplished on that hardware (System/360 Model 40).

Virtual memory on the 360/40 was achieved by placing a 64-word associative array between the CPU address generation circuits and the memory addressing logic. The array was activated *via* mode-switch logic in the PSW and was turned off whenever a hardware interrupt occurred.

The 64 words were designed to give us a relocate mechanism for each 4K bytes of our 256K-byte memory. Relocation was achieved by loading a user number into the search argument register of the associative array, turning on relocate mode, and presenting a CPU address. The match with user number and address would result in a word selected in the associative array. The position of the word (0-63) would yield the high-order 6 bits of a memory address. Because of a rather loose cycle time, this was accomplished on the 360/40 with no degradation of the overall memory cycle. In addition to the translate function, the associative array was used to record the hardware use and change status and the software-noted transient and locked conditions relative to a particular block of 4K bytes in the memory.

Since the array functioned as a content-addressable store when in supervisor state, searches to satisfy the LRU algorithm were quite fast. There is considerably more information on the associative memory in the referenced IEEE article by Lindquist, Seeber, and Comeau <ref. 1>.

The major difference between the CAT (Cambridge Address Translator) associative memory and our current line of relocate mechanisms was that the CAT was a memory-mapping device, whereas today's hardware employs a program-mapping scheme.

In a memory-mapping translation mechanism, there is one relocating entry for each page of real memory. In a program-mapping scheme, the hardware contains relocation information relative to the particular program that is executing, and information relative to real memory blocks is maintained elsewhere. The System/370, for instance, maintains use and change data in its key storage, which is a totally independent mechanism from the relocation mechanism.

Since it appears logically that memory mapping, *à la* S/360 Model 40, is superior to program mapping, then why isn't it prevalent today? The answer is cost; the original array cost thirty-five times what a conventional memory cell did at that time, and since then it seems that associative logic is still roughly eight to ten times what conventional logic costs. There has been little work done within IBM on associative technology, and, therefore, there is little likelihood that it will ever become price competitive in our hardware. What we should now look at is absolute cost and what associative logic can give us in additional function.

There is an additional problem in a memory-mapping scheme which doesn't exist with program-mapping techniques; that is page sharing. Memory-mapping schemes have only one entry per page of the real storage. To allow access among a group of users requires either changing that entry or having a second, or all-user, userid. The latter, of course, doesn't accommodate a selective sharing of memory.

The value of sharing programs in memory is suspect, based on the measurements taken during the life of the CP/40 project. There are some applications where it is desirable to share data in memory, and here the memory-mapping scheme is deficient. Interestingly enough, although sharing data has always been unpleasant in VM, yet no one seems to have as yet put forth an elegant solution.

CP/CMS Design

The two different goals of our project, one to measure S/360 software in a virtual memory, time-shared environment, and, second, to provide the Scientific Center staff members with an interactive facility, led us to a design which cleanly separated those two requirements.

The measurement requirement dictated that the functions to be measured and the algorithms to be modified and tested be very localized. A second motivation for maintaining a distinct separation became apparent when the strong wills and opinions of the group became apparent. I think that most designers recognize the need for good separation of function in programming system design, but compromise becomes the rule very early in the effort. With the particular group assembled to build CP/CMS, the personalities reinforced that design principle, rather than compromising it.

It seems now that the decision to provide a Control Program interface that duplicated the S/360 architecture interface <ref. 2> was an obvious choice. Although it was, given our measurement objective, it wasn't, given our in-house interactive system objective.

We were more secure with the decisions for the CMS external interface. It was clear, based upon the experience gained with CTSS, that a user-friendly command language was key. Another thing we had learned was that the system had to be very forgiving, and although options were desirable, default-mode, non-required parameters were to be a paramount design consideration in CMS.

The choice of an architected interface, the S/360, between CP/40 and its operating system turned out to have been most fortunate. It permitted simultaneous development of CP and CMS; it allowed us to measure non-virtual systems, OS and DOS, in a virtual memory environment, and it also provided a high level of integrity and security.

Conversely, this same design made sharing programs and data somewhat difficult. Even today's VM system seems lacking in this regard.

Program Model

Even though we all realized that there were loops in programs, essentially our model was that a program progressed linearly in its execution and data references. There certainly was no notion of "working set" prior to our original experiments.

We hoped to determine the proper page size, the rate at which page turning would occur in a multiprogrammed environment, and the value of shared program pages. Program and user characteristics included the number of instructions executed between I/O requests, which is still key for system designers. What was the time slice required to deliver acceptable performance? How much time did the user think between commands?

Experiments

The experiments run on the CP/40 system yielded significant results in the area of virtual memory. First, we discovered the phenomenon currently known as "thrashing". I first reported it to an internal IBM conference on storage hierarchy in December, 1966 <ref. 3>. In a follow-up paper on virtual memory <ref. 4>, we showed the dramatic reduction in the required number of page swaps that could be achieved through some very simple user optimization procedures.

A third report finally published in 1971 <ref. 5 and 6> contains the largest amount of data ever compiled on a controlled virtual memory experiment. It originally took 63 hours to run, so it is doubtful anyone would find it worthwhile to repeat. The experimental factors chosen for that experiment were:

- replacement algorithm
- subroutine ordering
- problem programs
- memory size

For each factor, three variables were chosen, such that the total number of experimental values was 81. Although some of the factors could be argued as to their validity, the experiment which measured page swaps and active and inactive counts gave researchers a better feel for the interactions caused by these factors.

The most significant result of the CP/40 work was the recognition that a multiprogramming system naturally divides its function into three levels of privilege and protection:

- *level 0*—the control program level—assumes allocation responsibility for the serially reusable resources of the system. It is at this level that multiprogramming or inter-job management takes place.

- *level 1*—the job management level—contains the functions normally required by a single job to start, stop, do command interpretation, maintain data files, etc.

- *level 2*—the application level—contains the function a particular user wishes to accomplish.

In CP/40 and subsequent VM offerings, we implemented this three-level structure in what is essentially a two-level architecture (S/370, S/370). By defining a true three-level multiprogramming architecture, the overall job of providing those functions necessary to accomplish the level 0 and level 1 tasks would be greatly simplified, and the resultant pathlengths would be considerably reduced.

Although most system programmers I've discussed this structure with agree to its merit, I'm not familiar with any hardware implemented this way.

A Look Back

The virtual machine design, as represented by the CP/CMS system, certainly has proven its viability, lasting now over fifteen years. During this period, we have added many new devices and software functions, so the system also meets the test of extendibility.

The success of the CP/CMS system certainly is in no small way attributable to its friendly and forgiving user interface.

A second contributor was the clean separation of function in the CP product, which made it easy for the sophisticated user population to remove, add, and substitute functions supplied by IBM, thereby greatly expanding the talent working on enriching the system.

Perhaps the most significant factor which contributed to CP's success in the middle to late 1960's was the failure of its competitors, Multics and TSS, to meet the commitments made to the marketplace. The market had been "hyped" to expect major function and performance (in the hundreds of terminals), and when they failed to deliver, expectations were lowered, but the need for time-sharing still existed among the customers. The CP/CMS system was operational and, therefore, represented an acceptable alternative.

The lesson to be learned when comparing CP/CMS with its contemporaries, TSS and Multics, is that it is easier to provide functions to an extendible high-performance system than it is to improve the performance of an integrated rich function system.

The problem for the future seems to be maintaining an architected three-level system structure within VM. There are major differences in the way an architect approaches the definition of an interface versus the way a programmer approaches it. An architect tries to insure the durability and completeness of what he specifies. He recognizes the scarcity of resources, be they parameters, bit interrupts, etc., and tests his specifications to make sure there is no needless resource consumption. The programmer views his job as to satisfy the requirements for a particular function. There is little concern for durability and conservation in this community.

As an example, the virtual machine interconnection facilities today are: Channel-to-Channel Adapter, VMCF, and IUCV. Essentially, they are all trying to pass data between virtual machines, but instead of extending and modifying the architecture of the original technique, the programming community invented net new things.

DIAGNOSE is another example of this phenomenon. Functions are provided in CP, invoked through the DIAGNOSE interface, without an appreciation for the logical (three-level) structure behind the VM design. Because of this, movement of CMS without CP level 1 and level 2 code to a standalone environment has proved extremely difficult. Remember that, in the original implementation, CMS ran on a S/360 Model 40 without the requirement for the CP multiprogramming software.

In closing, I believe there is much to be gained if we build hardware which supports a three-level software structure, and I hope to stimulate interest in this by this presentation.

References

1. "A Time-Sharing System Using an Associative Memory", A.V. Lindquist, R.R. Seeber, and L.W. Comeau, *Proceedings of the IEEE*, vol. 54, no. 12, December, 1966.

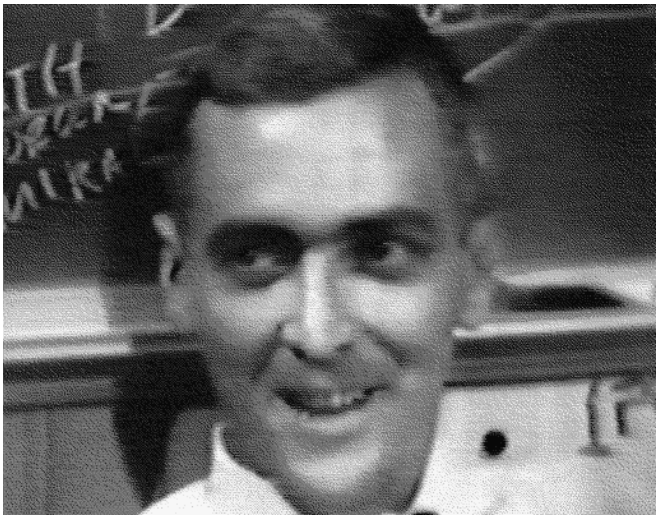
2. "A Virtual Machine System for the 360/40", R.J. Adair, R.U. Bayles, L.W. Comeau, and R.J. Creasy, *IBM CSC Report*, May, 1966.

3. "Operating System/360 Paging Studies", L.W. Comeau, *IBM Storage Hierarchy System Symposium*, December, 1966.

4. "A Study of the Effect of User Program Optimization in a Paging System", L.W. Comeau, *ACM Symposium on Operating Systems*, October, 1967.

5. "A Multifactor Paging Experiment: Part I, the experiment and conclusions", R.T. Tsao, L.W. Comeau, and B.M. Margolin, *Statistical Computer Performance Evaluation*, Academic Press, Freiburger *et al.*, ed., 1972.

6. Same as above, *IBM Research Report RC3443*, July 9, 1971.



Lesin Comeau, CSC, 1965

Appendix B

DAVE TUTTLE'S MEMOIRS

Melinda's Preface

Several years ago, when I was doing the research for an earlier paper, Romney White advised me to talk to Dave Tuttle, who had been one of the developers for VM/370 Release 1 and who had, like so many others, left VM (and IBM) when VM Development was moved from Burlington to Poughkeepsie. Like all the other advice Romney has ever given me, that was very sound. I quickly came to value Dave as an historian's dream source, for he combines an astonishing memory with warm and witty perceptions.

Over the past few months, as I've bombarded Dave with drafts of this paper, I've had the good fortune to be the recipient of several long notes containing reminiscences of his days working on VM. Having enjoyed his letters so much, I asked Dave to allow me to include them in this paper for others to enjoy.

—MWV

Dave's Preface

The time I was at IBM and involved with VM/370 was an important, and fairly intense, portion of my life and career. You may have, however, turned loose more than you counted on—by expressing an interest and being a good listener. Here are some of my recollections of "the early days", from the point of view of a young participant.[171]

(In writing up these events, I am not trying to emphasize my own part in things; **everybody** was a hero according to normal IBM standards. The things that I remember most vividly are the ones that I was involved in directly. I have found some curious discrepancies in what I remember. I know quite well the general sequence of events in several different "threads" of activity, but I'm not at all sure that I can relate them to actual dates or, in some cases, to each other.)

—Dave Tuttle

CP-67/CMS and CSC—1968 to 1971

Ed Hendricks and I met during 1967 while he was working in the M.I.T. Computation Center in a post-grad program and I was there as a part-time user consultant and system programmer. Through that contact, I managed to get a part-time position at the Cambridge Scientific Center in the fall of 1968. I started on October 18, 1968, one week before my 20th birthday. Not only was it an exciting technical environment, it was also the middle of the rising tide of social and political awareness in Cambridge and Boston.

---

[171]

My experience at the Cambridge Scientific Center started just after the first successes of CP-67/CMS. Bob Creasy had recently transferred to the Palo Alto Scientific Center, Les Comeau was off somewhere else within IBM, Dick Bayles had left IBM to start up National CSS in Connecticut—a time-sharing service bureau based partially on a modified CP-67/CMS—and a lot of the CSC people were working on using CP-67/CMS, rather than merely trying to make it work. The modified S/360-40 was no longer around; its place on the 4th floor had been taken by an IBM 1130/2250 Mod 4 system, and a 512K S/360-67, in one corner of the third floor, served as the main system for the Center.

Organization Notes

Although my memory is a little uncertain, I think there were three main groups at CSC. Craig Johnson ran the Graphics group, comprised of Ed Hendricks, myself, and several others whose names have disappeared into the mists. Dr. Tim Johnson of M.I.T. was working with us, as was Bob Seawright on a part-time basis. The CP-67/CMS system work was under Rip Parmelee, I think, and included Bob Adair, Dick Meyer, Harit Nanavati (one of the later founders [?] of Interactive Data Corp.), and the others which you have listed. The third group was known as Operations Research, run by Marty Schatzoff and including Don Hatfield, Stu Greenberg, John Ravin, and several more.

In addition to the CSC people, 545 Technology Square also housed the IBM Boston Programming Center, on the third floor, where the CPS/360 system was developed and maintained. Among their top people were Nat Rochester and Jean Sammett. The BPC reported through a slightly different chain of command. The five U.S. Scientific Centers (Palo Alto, Houston, Washington D.C., Philadelphia, and Cambridge) all reported into IBM Data Processing Division Headquarters (DPD-HQ) in White Plains. The BPC group, the Time-Life Development Center (New York), and a few others scattered around the country reported into DPD Industry Marketing and Development (DPD-IM&D). IBM Yorktown Research was an entirely separate organization, essentially a division unto itself, while all of the "mainstream" system development work was the province of the System Development Division (SDD) in Poughkeepsie, New York.

Technical Matters

The first project Ed and I worked on was the development of a BSC communications package to exchange data between OS/360-PCP, in a virtual machine of CP-67/CMS, and an IBM 1130. The OS/360 side of the connection was running an associative database for graphics data; the 1130 end had a multi-tasking monitor driving an IBM 2250 Mod 4 display, a Sylvania Data Tablet, and Sketchpad III, a 3-D drawing and display program developed primarily by Dr. Tim Johnson of M.I.T. as an extension of the original Evans & Sutherland Sketchpad concepts.

We had a few difficulties to deal with. OS/360-PCP was not a multi-tasking system, so Ed had to develop a user-level round-robin "scheduler" to support concurrent operation of the database code and the communications code. [The techniques he used were perhaps based on some earlier work he had done at M.I.T. in 1967—*i.e.*, bringing up a version of the SpaceWar game on the Computer Center's S/360-65, which had an IBM 2250 Model 1 (channel attached) vector display.] Secondly, BSC communications were, at that time, far from being well supported or understood. The first attempts at using the OS/360 BTAM support failed because the system macros wouldn't build the correct DCB formats; we ended up using EXCP and writing our own channel control programs. On more than one occasion we had to take out the schematics and logic manuals for the 2701, an oscilloscope, and a data-line strip recorder to find out what was

going on. We invented a protocol of our own to handle pseudo-duplex exchange of multiple unrelated data streams, with what may have been the first use of selective acknowledgements piggy-backed on data blocks.[172]

On the other end of the wire we also had a round-robin multi-tasking monitor in the 1130. The graphics display used the main memory of the 1130 for its display buffer, and there was a large collection of interrupt and command response routines which serviced light pen tracking, function keys, the display keyboard, the system keyboard and printer, a line printer, the comms line, the disks, etc. There was a group of a half dozen or so working on the 1130 code and four or five working on the OS/360 code. I ended up doing the comms code on both ends, some work on the PL/I database code, and several of the device service routines on the 1130. We eventually did get it all working, sometime towards the end of summer, 1969.[173]

The work that we did at that time seems to have had a lot of longer-term impact. The 1130 code and the OS/360 virtual machine code were made available to Brown University in a joint development agreement, and they became the substructure of the Brown HyperText system. The OS/PCP system in a virtual machine was also the basis for ONLINE/OS, a single-user interactive monitor much like CMS, except that it had the full OS/360 data management capability and online support for all of the OS compilers and utilities. It was never released outside of IBM, but we did sneak it into the Spring Joint Computer Conference in Boston in 1969 as "SJCC/DEMO", a name which conveniently had exactly the same number of characters.[174]

Similarly, the multi-tasking and multi-thread communications experience, and a version of the 1130 code which turned it into a remote spooling "workstation", led fairly directly to the original CPREMOTE, which used the BSC protocol we had developed and an Ed Hendricks' special 4K standalone monitor. The CP-67 modifications which allowed CPREMOTE to read spool file blocks *via* DIAGNOSE were also developed somewhere along the way.

---

[172] E.C. Hendricks and D.B. Tuttle, *Notes on Design Objectives and Implementation under OS/360 of a General Purpose Binary Synchronous Telecommunications Package for Multi-Programmed Applications in OS/360,* IBM Cambridge Scientific Center Report 320-2047, August, 1969.

[173] E.C. Hendricks and D.B. Tuttle, *HOTLINE: A Binary Synchronous Access Method*, IBM Technical Disclosure Bulletin WA8-70-0091, September, 1970. Tuttle, D.B., *BSCCA: An Interrupt Service Subroutine for Binary Synchronous Operation of the IBM 1130 Synchronous Communications Adapter*, IBM Cambridge Scientific Center Report ZZ20-2096, October, 1969.

[174] E.C. Hendricks, C.I. Johnson, R.D. Seawright, and D.B. Tuttle, *Introduction to ONLINE/OS* and *ONLINE/OS User's Guide*, IBM Cambridge Scientific Center Reports 320-2036, 320-2037, March, 1969.

"Environmental" Factors

The "great unbundling", in June of 1969, constituted a real threat to the Scientific Centers and the IM&D groups. The new way of doing business created a lot of confusion for the IBM organizations which were not a part of the mainstream System Development Division (SDD), because it was no longer clear how non-product software could be made available outside of IBM. The first releases of CP-67/CMS were by way of the Type III Library. The release of CP-67 Version 2 almost never happened; the submittal missed the deadline for freezing the Type III library, prior to the June 23rd announcement, and it took a lot of high-level arguing to actually get it accepted.

Following the June 23 unbundling announcement, the CP-67/CMS Development Group, at that time under Dick Meyer, split off from the Cambridge Scientific Center (almost literally; they took over some office space on the fourth floor which had been part of CSC, and put up a wall between the two areas. The door was seldom locked, but IBM required the separation.) Dick Meyer had taken over management of the group when Rip Parmelee left for an 18-month assignment at the Grenoble Scientific Center, around the time of the SJCC. CSC stayed in the DPD-HQ chain, while the CP-67/CMS group (11 people, at the time) switched over to DPD-IM&D. I stayed with the CSC Graphics group, at least for the moment.

More Technical Matters

In the fall of 1969 I started working on a new editor. People generally knew that the CMS editor was somewhat limited and clumsy to use, and we wanted an editor that could also be ported to ONLINE/OS. The eventual result, in March, 1970, was called Ned ("New EDitor", of course!), and it quickly became very popular as a replacement editor for CP-67/CMS within IBM. Some of the important new features were an imbedded macro language and macro processing, external descriptor files for filetype-dependent defaults, generic "target" capability for a wide range of editing commands, a CHANGE command with support for compound, repetitive, and recursive changes, and the ability to edit files which were up to 10 times as large as available virtual memory. What it did not have was a screen interface (the 3270 was announced in 1972, I think, and 2260s were never very common) or the ability to generate UPDATE files; otherwise you might recognize it as the immediate father of Xedit.[175]

The popularity of Ned within IBM was the cause of my first experience with SHARE or GUIDE meetings. I was invited to SHARE XXXV in Montreal in the fall of 1970, as the result of a behind-the-scenes nudge by Nat Rochester, of IBM's ATS/360 group. I drove up to Montreal in my orange-and-black, used-to-be-a-Public-Works truck, named "Truck", of course, only to discover that I would be staying in the brand new Hotel Bonaventure—costumed Oriental bell-boys, lobby on the tenth floor, roof gardens, and all. I survived that first meeting week, and it turned out to be only the first of my many SHARE meetings and, later, GUIDE meetings.

My first GUIDE meeting was, coincidentally, GUIDE XXXV, also in Montreal. I was somewhat unprepared for the difference in style between SHARE and GUIDE—the VM/370 group had to buy me a necktie so that I could talk at one of the DP Management sessions; I hadn't brought one along! (Many years later, after a lapse of eight years or more, I attended a GUIDE meeting in Los Angeles as a representative for GTE Telenet. Within 15 minutes of appearing at one of the

---

[175] D.B. Tuttle, *Context Editors, Part II: EDIT II—A Non-System-Specific Context Editor*, IBM Cambridge Scientific Center Report 320-2048, March, 1970.

VM/370 sessions, I had been recognized, given a list of all of the former IBM VM people who were attending the meeting, and reminded that the group still had the Dave Tuttle memorial necktie!)

On another occasion there was a GUIDE meeting in Boston, which gave more of the VM/370 crew a chance to listen to and talk with the user community. As one of the regular GUIDE attendees, I didn't get to go to many of the sessions, but I was expected to show up at least once or twice at SCIDS. I caused something of a sensation one night when I walked into SCIDS in a jacket and tie—and a full Scottish kilt, knee socks, sporran, and my IBM badge. I'm not very much of a Scot, but there is supposedly a sixteenth or so of Royal Stewart in my ancestry, and I didn't have a clean suit that fit.

The Early Days of VM/370 Release 1

I don't know from my own experience exactly when the planning began for VM/370. Late in 1970 I transferred from the CSC Graphics group to the Systems group, still in the Scientific Center, around the time that Rip Parmelee returned from his assignment in France. The CSC Systems group was working on the series of CP-67/CMS modifications which would eventually support a real S/370. The sequence, as I remember it, was approximately as follows:

1. CP-67 modified to provide a virtual S/360-67 (3rd quarter 1969?). This system became the production system at CSC to support further experimentation. At one point in early 1970 or 1971, IBM Yorktown Research ran some tests to verify the successful "virtualization" of the full S/360-67 function. They ran out of terminals and time when they were loading CMS in a virtual machine 17 levels deep! (CP under CP, under CP, ...—1 real CP, 15 virtual CPs).

2. Modified again, to provide a limited S/360-67 Multi-Processor in a pair of virtual machines (late 1970; I was involved). This system supported development of real MP support for CP-67, which was brought up on the CSC machine pair sometime in 1971.

3. Modified separately, to provide a virtual machine with simulated S/370 privileged instructions, control registers, and one version of the S/370 dynamic address translation (DAT) architecture (4K pages, 1M segments, as supported by the S/360-67 hardware). CP-67/CMS never supported 2K pages in a meaningful fashion, because the S/360-67 did not have the necessary hardware support.

4. Modified again, to support operation in a S/370 virtual machine (little if any machine check recovery, no support for S/370-specific DASD, still 4K pages, 1M segments). This was ready about the middle of 1971, before we had a real S/370 to play with. Almost all of the work was done using the IBM Confidential "Red Book" architecture documents.

I joined the CP Development Group on August 1, 1971, in exchange for Lynn Wheeler. He took my slot at CSC, I took his in the CP group. The first work I did was to spend two frantic weeks writing "Alpha Test" functional and logic documentation for Ned, which was intended to be the CMS Editor for the first release. The first approval stage of the VM/370 project was to get a conditional go-ahead for the planning and design efforts; that was done in late Spring, 1971. The next hurdle was to prepare a complete set of design and function specifications, to standards set by SDD, and get them approved for implementation. When I joined the group, we were almost ready for the first submittal of the document set.

The first submittal failed, almost predictably. It took a total of, I think, four tries to secure the Alpha Test approval. One incident that was not pleasant involved my editor, Ned. It was one of the first CMS components to be approved (second submittal, I think), but a decision was made between the third and fourth submittals to pull it back, substituting a slightly updated version of the existing CP-67/CMS editor. Ned was judged to be "too sophisticated for the average user", and it was substantially more code—potentially a problem in small systems. To make matters worse, I was not told by local management that the decision had been made—I heard it from one of my friends at Yorktown Research, almost accidentally, after the final Alpha Test approval.

The Alpha Test process was a series of negotiations both of function and of scale. The original plan was to completely rewrite both CP and CMS, with a lot of new functions and improvements in both components. The plan which finally was approved involved a much reduced CMS effort, essentially a "port" of CMS to the S/370 environment, rather than a complete restructuring. The CP plans came through more or less intact, except for the planned multi-path I/O and later multi-processor support. We knew that we were working toward a deadline of the "big splash" Advanced Function announcement, but nobody knew yet exactly when that would be.

After the first few weeks of documentation, working with Bob Downs, by the way, I was put to work with Paul Tardif on further modifications to the CP-67 "I" system—the version which actually ran on S/370 hardware. Some time in the late summer of 1971 a 512K (nominal) S/370-145 was installed in our machine room on the third floor. For security reasons, it was moved in the wee hours of one morning all in one piece, by crane, through one of the windows which had been removed temporarily. "Everybody" knew that the 545 Tech Square IBM groups worked only on virtual storage systems, so it would have been a pre-announcement of the S/370 capability if anyone knew we had the machine.

Some of the problems we had to deal with in the early development were interesting. Every other month or so, we would get a new version of the loadable microcode for the 145, and more often than not there would be some impact on the privileged instruction set. There were three generations of slightly different DAT control instructions, and one instruction which had two different opcode assignments. One of the things that Paul and I had to deal with was support for the new DASD devices, the 3330 and the 2305, and the new Block Multiplexor channels.

The first step was to modify the CP-67 I-System to support the 3330 and 2305 as devices attached to a virtual machine. That provided an environment for us to develop an I-System version that would support them as residence and paging devices. It was in that step that Dick Newson and I almost lost everything, early one morning... The system which supported attached 3330 and 2305 devices was not particularly stable, and the 2314s which we were using as production drives were not very fast. Consequently, we were in a bit of a hurry to move over to the 3330s. Also, the nominal 512K memory of the S/370-145 was actually quite a bit less; the loadable microcode support for DAT, the advanced timers, PER, etc., used about 20,000 bytes of main memory in addition to the regular control storage. The nominal 524,288 bytes of memory was reduced to about 504,000 bytes—not quite enough to load the CP nucleus from tape or cards.

I've forgotten the exact sequence which got us into trouble, but Dick and I spent about half an hour in mild panic around 7:30 in the morning. We knew that people were going to be arriving soon, but we had no running system and no way to load a new nucleus onto any of the disks. We escaped finally because I was able to IPL the system in Instruction Step mode, make some in-storage patches from the console, then bring up one of the unstable systems just long enough to rebuild a CP nucleus of the "production" I-System. Only the two of us knew how close we had come to a major setback!

The CP development team, reporting to Dick Newson, was an interesting mixture of people—one of the things which undoubtedly was a factor in the success of the implementation. We were a mixture of hot-shot "kids", former academic/research types, and long-service IBM Field Engineers. The detailed implementation approach was based on the new control block structure which Dick Newson (and possibly others—I'm not sure) had designed for CP, along with a set of register usage conventions, command scanning routines, and module linkage macros which Dick Newson and Carl Young had developed. Each of us took some pieces of the old CP-67 "I" code and either recoded it to the new control blocks, register usage, and module naming conventions or, where the S/370 architecture encouraged it, redesigned the function entirely. The people and pieces that I remember were something like the following:

- Dick Newson - DMKSCN, DMKRIO, control blocks, high-level design
- Carl Young - DMKDSP, DMKSCH, DMKPAG, DMKPTR, DMKIOS
- John Seymour - DMKFRE, DMKCCW, DMKPTR, DMKPRG, DMKPSA, DMKTRC, DMKTRA
- Ed Murray - DMKCKP, DMKVSP, all unit-record spooling
- Charlie Weagle and Ray Grein - DMKMCH, DMKCCH, model-specific support, DMKFMT, DMKLDR, DMKDIR
- Clyde Wildes - DMKQCN, DMKCNS, DMKVCN, terminal control commands
- Larry Estelle - DMKVIO, DMKVDB, DMKLNK, minidisk I/O, disk I/O recovery
- Dave Tuttle - DMKPRV, DMKVAT, DMKHVC, DMKHVD, DMKDIA, DMKDRD, DMKPER, DMKDEF, VMFxxxx utilities

Undoubtedly I have forgotten some of the details, but I'm pretty sure that there were just the nine of us in the immediate management group. There was some trading of modules back and forth as we went along. I did some work in DMKPRG to support the interfaces to DMKPRV, DMKVAT, and, later, to DMKPER. I also did the Attach, Detach, Define, and Dial commands in their entirety, after Larry Estelle had done most of the basic minidisk management commands in DMKVDB. Over a period of time each of us worked in a lot of different areas of CP, but there was generally a recognized expert for each major sub-system or set of functions.

The separation of functions into modules was a fluid thing; our goal was to keep every module within range of a single base register—4096 bytes—even within the resident portion of the nucleus. Modules in the non-resident portion had to be kept smaller than about 3700 bytes—one 4K page minus the 10% margin required by SDD standards. For example, when Dick Newson gave me the task of rewriting the privileged instruction simulation and virtual DAT functions, which had been all in one module in CP-67, I went back to him a week or so later and told him that it would take at least three separate modules. As we incrementally added the planned functions, it grew into a cluster of five or six modules, some resident and some pageable.

For several months, from the arrival of the S/370-145 until February, 1972, everybody used the CP I-System as a production time-sharing system. CMS development, tech pubs, and the stand-alone utilities (LDR, DIR, FMT, etc.) could all proceed without much difficulty. It wasn't until sometime in February, however, that we had completed enough of the CP work to think that we might have a runnable system—*i.e.*, able to support a CMS virtual machine and accomplish

some work in it. As soon as the basic functions were in place and somewhat shaken down, we went into an interesting and intensive period of trying to use the system for everyday work, at the same time that we were working hard on filling out the CP capabilities.

From the middle of February until sometime in early June, we were on a schedule something like the following:

1. Using the production I-System, collect the latest group of fixes, new functions, and/or new modules, and build a VM-CP nucleus.

2. With 15 minutes notice to the users, shut down the I-System and bring everyone up on VM. One of the earliest functions running was the fast dump and restart logic.

3. After we had accumulated three to five CP dumps on tape, a period which ranged from as little as 45 minutes in the beginning to as much as three or four hours, take down VM and bring up the I-System again.

4. Print off the CP dumps. In the machine room (visible in the photo of Ed, Larry, and myself) was a work table which was the repository of the most current listings of each of the CP modules. Each day one or two of us had "triage" duty—scanning the dumps to roughly localize the problem, then passing it on to the person working on that area of the system. Carl Young and I frequently had the duty, because we each had a broad overview of the system.

The "Triage" Desk
Larry Estelle, Dave Tuttle, and Ed Christiansen

5. When we had identified fixes for each of the dumps and reassembled the appropriate modules, usually within a few hours, we would update the listing rack and start over from step 1.

In February and March we had up to four different VM-CP systems every day, as we were shaking out some of the more obvious problems. It slowed down substantially as time went on, but the problems became not so easy to find nor quick to fix! Through it all, the CMS group and the Tech Pubs group put up with a lot a disruption, but we all ended up with a system we were very proud of.

Anecdotes

The story of Program Event Recording in VM/370 would probably interest you, because we had a chance to support it in Release 1.0 but had to pull it out at the last minute. One of my explicit roles in the CP group was to be the explorer of new devices and new features, hence my earlier involvement with the 3330 and 2305 support. As part of the DMKPRV work I had to support both the control registers used by the PER feature and the interrupts which would be generated by a virtual machine system which enabled PER. Along the way I also wrote a full-function version of DMKPER and a companion command module, modelled after the old standby TRACE command, which used the PER hardware to provide non-intrusive and selective instruction tracing, fetch/store storage traps, single-step execution, etc.

Unfortunately, debugging the TRAPPER command (abbreviated TRAP, of course) uncovered some subtle problems with the PER microcode in the Model 145. There was an interaction between Extended Control Mode, Dynamic Address Translation, and Program Event Recording which resulted in a failure to suppress PER interrupts when you went from EC-mode, DAT, PER to BC-mode, non-translate—*i.e.*, whenever you re-entered the CP nucleus! On a couple of occasions I observed an "impossible" console trace of program events in the CP handling of a virtual machine privileged instruction. To this day I don't know how the system continued to operate. When we tried to communicate with the microcode group (Kingston?), I learned the real meaning of "jargon". While I could describe the problem situation very clearly, I couldn't make any sense out of the questions I got in reply, so I couldn't help the microcode people enough to get a quick fix. This was sometime in the late spring of 1972 and there was a lot of schedule pressure, both on us and on the hardware development groups. As a result, the full TRAPPER support was replaced with a "stub" DMKPER module for the first release.

Another anecdote from that amazing year arose when we did our first testing on a S/370 Model 135, again in the spring of 1972. I don't remember exactly who was with me, but two of us gathered up a set of early VM/370 tapes and a disk pack or two with a pre-built system on it and travelled to the hardware development labs in Kingston. We arrived in the truly giant system assembly building there and were led on a long walk to a tiny room somewhere in the bowels of the building. In the room was the "C Test" 135 system that we were to test on. This was a machine that had not yet been announced, with some of IBM's latest technology in it—but it looked like it had been used hard for years! The paint had worn off around many of the console controls, the system was dirty, the rotary knobs were loose, and one of the console switches was broken. The hardware "C Test", for mechanical and electrical reliability, was apparently pretty rough!

Our first attempt at bringing up the VM system on the 135 met with very little success. After an hour and a half of poking around, we discovered that there were some serious interrupt problems with the advanced timers (CPU Timer and Clock Comparator, both new features in S/370 Advanced Function). In most of the S/370 models these timers were implemented at least partially in hardware. The smaller, less expensive Model 135 had implemented them entirely in microcode. They had tested extensively with the mainstream OS/VS1 and DOS/VS systems, but VM used the timers much differently—the other systems used it for time-slicing, while we used it

primarily for usage accounting. We were lucky; on hand for the test were two of the microcode authors from the IBM labs in England. While we took a break for lunch, they rewrote the timer microcode on the fly from the system console. After lunch we tried the system again, and it worked with no problems at all!

The saga continued several weeks later, when it came time to test the "official" version of the new microcode timer support. Once more, a couple of us travelled to Kingston for the test, but we had just a set of dump tapes with us and they had given us a different set of directions. Instead of the huge system assembly building, we had been directed to the parking lot of the IBM Kingston Recreation Center. After some driving around, we arrived to find a little shack on a concrete slab in the corner of a field—the kind of pre-fab building you would put up in your back yard for storage. Inside was a complete S/370-135 system, set up for environmental (noise, heat, etc.) testing, with not another building for close to half a mile in any direction. Trying to be professional about things, we dutifully sat down at the system, formatted a disk pack or two, and set about loading the tapes onto disk. As I was sitting at the console waiting, the building started to fill up with electrical smoke—one the disk drives had caught fire! I resisted the urge to use the "Emergency Power Off" switch, but it was hard—how many people have ever had a legitimate occasion to do it? I simply hit System Reset, then Power Off, as all of us quickly cleared the building.

After a few minutes with the windows and doors open, we were able to get back in and use the big pedestal fans to clear out the smoke. The hardware people cut the affected drive out of the 2344 group—it wasn't one of the drives we had been using—and we were able to finish loading the system. This time the VM/370 testing went without a hitch. As we packed up our material and headed out, we were treated to a scene right out of the "Twilight Zone". It was at the end of dusk, in the middle of essentially nowhere; a dark sky full of purple clouds and stars looked down on a tiny building with light pouring out of the doors and windows—with a full-scale computer system and hardware engineers working away, but not another soul as far as you could see!

More Tales of VM/370 Release 1

Many of the aspects of the early VM/370 development are best appreciated in retrospect. For the entire period we were working with very little margin for error. If we had any serious hardware failure, there were no backup systems closer than Poughkeepsie or Kingston, New York; we took the normal once or twice weekly disk backups, but a few days worth of work lost to a disk failure would have been a serious setback. There was some contingency margin in the schedules, but most of that "margin" could be realized only by reduced system testing or by reduced function at First Customer Ship (FCS). In the more careful software development environment of the 1980's, nobody in their right mind would accept the same level of risk.

The lack of a backup system almost caught us when it came time to try running OS/VS1 under VM, in the late spring of 1971. The VS1 group was in Endicott at the time, I think, and they had been working on the virtual memory extensions to OS/MFT longer than we had been working on VM/370. Historically we had a somewhat better relationship with the Endicott people than we did with the Poughkeepsie OS/MVT-TSO-OS/VS2 people. The OS/MFT-VS1 people were "second class citizens", viewed as being squeezed out between the success of DOS/360 at the low end and the grandiose capability of MVT-TSO-VS2 for mid-sized and larger systems. The VM/370 people, not even in SDD, were generally viewed as at best a "dark horse" possibility. There had been some cooperation between the OS/MFT group and the CP-67/CMS group, along with some experimentation to improve the operation of MFT in a virtual machine—but I'm not at all sure when it began. Over a period of a couple of years we forged an alliance with the VS1

people against the TSO-VS2 "bandwagon". VM-CP and CMS provided an interactive programming environment to complement the OS/360-compatible batch capabilities of OS/VS1, as an alternative to the OS/VS2-TSO combination.

When we first had the opportunity to test OS/VS1 under VM/370, the Endicott group sent two of their experienced system people to Cambridge. Schedules were such that we had no more than one or two days to accomplish as much as we could, prior to a development freeze of some kind for VS1. Naturally, we had been having some machine trouble—the weather had been warming up fast, the machine room was in the south-east corner of the third floor, and we had had to shut down the system once or twice because of memory overheating. The day that the VS1 people arrived, we had to shut down the system mid-morning with the memory overheated, only to discover that two cooling fans were not running in one of the memory arrays. The S/370 systems were still quite new, and the IBM repair people could only find one replacement fan in the Boston area. They assured us that we could "probably run OK" with just one fan! Rather than just sit around while the CE went to pick up the spare part, I took the old fans over to workbench in the machine room and stripped one of them down to see if I could find the problem. The overheating had apparently congealed the bearing grease, so I cleaned it out, found a can of lithium grease in the CE cabinet, and repacked the bearings. By the time the CE came back from the main IBM office outside of Harvard Square, I had a roughly-rebuilt fan which we installed along with the one new one. The system held up well during an afternoon of successful testing, and the Endicott people went home with yet another tale of "those crazy people in Cambridge".

Personnel Notes

As I mentioned previously, the VM/370 group in the summer of 1971 and onward was an interesting mix of people. Somewhere in the assembly process we picked up most of the CPS/360 (CPSS?) people from the Boston Programming Center and a significant number of "refugees" from the New York Time-Life Development Center, which was closed down sometime in 1971 or 1972, I think. While the VM/370 development was proceeding in secret, the group was still responsible for supporting CP-67/CMS, CPS/360, and one or two semi-products which had been handled by the Time-Life group. The collective responsibility for several "orphaned" products, each with its own set of dedicated users, and the fact that the group was still in the DP Division were contributing factors to the pro-user/contra-SDD attitude which generally prevailed.

One area of contention we always had to deal with was IBM product standards. We were deliberately trying to produce a mainstream System Control Program (SCP), but we didn't always agree (!) with the standards established by SDD for such things as source control, command scanning and interpretation, error message numbering and coding, documentation library structure and content, etc. Every time we wanted to do things our own way, it required another round of negotiation or justification or, in one case, a threatened programmer revolt. The administrative and marketing people in the VM group deserve a lot of credit for succeeding as often as they did in dealing with the IBM "establishment".

The "programmer revolt" resulted from an SDD standard that would have required us to reject any command which was entered with too many arguments—even for commands with a fixed or known maximum number of arguments. The "tradition" for CP-67 and CMS was to accept commands if the required arguments were present and valid, and to ignore as comments anything else which might have been entered. One afternoon Charlie Weagle and I "explained" to Dick Newson that the additional logic to scan past the last valid argument was simply too much code—it would not fit within the 4K-byte limit for the pageable CP command modules. "Trust

us, Dick. If you make us put it in, it won't fit! We will find a way, even if the current module is only 200 bytes long...", or words to that effect.

Another group of unsung heroes was the documentation group, though one or two did get IBM awards for the Release 1 effort. When we started the documentation effort in early 1972, the original CP-67/CMS crew of four or five writers and support people was far from adequate to the task. The eventual answer was to bring in a group of temporary office people, ten or eleven in all, from one of the regular Boston agencies. Over the nine months or so of the initial documentation cycle, those "temporary" workers learned CMS, Script, the Ned editor, and were instrumental in producing over 3000 pages of IBM product documentation! In spite of this heroic effort, we almost lost the people in an idiotic administrative scramble which followed FCS in November, 1972.

Along the road to the VM/370 inclusion in the August, 1972, S/370 Advanced Function announcement, plans emerged to incorporate the VM/370 group into the System Development Division. I don't know whether it was a concession we were forced to make or a known condition from the start, but we did transfer from DPD-IM&D to SDD on January 1, 1973. Prior to the transfer, in the late fall of 1972, some of the group had physically moved to the IBM Service Bureau Corp. building in Burlington, Mass. (24 New England Executive Park or "NEEP", several hundred yards behind the huge Burlington Mall on Route 128). We were definitely outgrowing the space in 545 Technology Square, both for systems and for people. One of the first things added in Burlington was a second S/370 system, a 155-II, while we were still sharing the building with SBC. By February, 1973, I think, all of the VM/370 group had relocated to Burlington.

Somewhere in this process the "supplemental" people working both in documentation and in the system operations group were transferred from DPD-IM&D to Service Bureau Corp.—the employment policies for SDD apparently didn't provide for supplementals! We almost lost some of the people on the spot; they wanted to work with us, but at least two or three of them had had bad experiences working for SBC in the past as temporary help. In the next few months things got worse; all of SBC was "traded" to Control Data Corp. in the settlement of one of long-standing anti-trust suits! That provoked some relatively frantic scrambling to "rescue" our people from the trade and bring them on board, finally, as full-time IBM employees with the benefits that they deserved.

Adding people to the group over the next several years was another source of occasional tales. We were one of the very few (only?) major development groups which was located in a competitive job market. IBM's unwritten policy was to set up its large facilities just far enough away from major metropolitan areas to escape heavy competition for personnel, then pay their people a little more than the going rate. There was also a concern, inherited from the late 1950's and Tom Watson, Sr., I think, to make sure that the big plants were outside the nuclear blast radius of known strategic targets—reportedly the original reason for the production move to Poughkeepsie and Kingston. As a result, our Boston area location gave us an advantage in attracting top-level people. One of the "jewels" we attracted, in late 1973 or early 1974, I think, was George Saunders. He was a refugee from IBM Kingston, and we put him to work on the remote 3270 support.

From the beginning, George adapted to the system and to the environment very well—he and I shared an office, so he didn't have much choice but to succeed (: -). Some of the other notable additions that I worked closely with were Charlie Johnson and Mark Dunn. Mark was one of our early converts; he had worked as an operator at the Cambridge Scientific Center as a Co-Op student at Northeastern University.

On one other occasion (Spring 1975?), we tried to bring in a staff-level programmer from the Kingston communications group, to work with Charlie and Mark and myself on native SNA support, but we failed for an odd set of reasons. I don't remember the man's name; he came to us with a very good background in IBM systems development and communications expertise that we needed, but no prior experience with VM/370 or CP-67. He spent two weeks with us learning the system from the outside in, then went back to Kingston! As he explained to me, he could not believe that VM/370 was possible; he understood what was going on in CP, he could see it work, but he could not make himself accept it. The details of intercepting program interrupts, simulating instructions, translating I/O programs, managing virtual and real storage—all the basic things which create a virtual machine—had to require more code than could possibly be executed in the time available! VM/370 obviously worked, but he couldn't understand how it was ever possible to support more than two or three virtual machines at a time. To him, it felt like magic, and he had no confidence in his ability to work on the system. (Of course, that wasn't the only factor in his decision; housing in Boston was a lot more expensive than in Kingston, New York, but that sort of ruins the story!)

Personal Glimpses

John Seymour was one of the foundations of the early VM-CP group. He was both one of the older people around and one of the long-time CP-CMS people. John was also an extremely careful programmer. You could tell immediately if the module listing you took out of the rack was one which John had worked on most recently; every line of code, including macro expansions, had a ball-point check mark next to it from his final inspection. Difficult or critical areas of the code often had two check marks on each line. As a result, John was the local champion in fewest defects per 1000 lines of code, with a measured rating of 1.1 to 1.2, in spite of the fact that most of his work was in "simple" areas such as DMKCCW, DMKFRE, DMKPTR, etc.

Not too long after VM/370 FCS, an APAR came in which required a fix to DMKFRE, the CP storage allocation module. John found the problem and fixed it, then remarked, "There's one more."—DMKFRE was a little bit more than 1000 lines of code. About six months later, another DMKFRE problem showed up. After that second problem was fixed, DMKFRE ran without failure from 1973 until well after the group broke up in late summer 1976—executing two or three hundred times a second in hundreds of different systems! There had been just two bugs, as he expected.

John Xenakis, also known as "Captain Midnight", came to us from the Time-Life Development Center, I think. He earned his nickname in the "stretch run" from Spring 1972 until the announcement in August. John was seldom around the office during normal business hours, unless you happened to catch him on his way out in the morning or on his way in about supper time. He maintained a supply of fresh fruit and soda in his office, and he had an electric popcorn popper which he used to make rice (!), all to support his marathon work sessions. It was common to arrive in the morning and find a note on your door explaining which new features had been added to CMS since the day before. Tom Rosato, then manager of the CMS group, was probably the originator of the "Captain Midnight" name—Tom was routinely exasperated about losing control of the group, but he needed the extraordinary productivity. Not long after one of Tom's tirades, John started to use the nickname as a signature on his morning notes.

Paul Tardif was another very good person to have around, though he was only with us for six or eight months. Paul brought a certain cosmopolitan flair that helped balance the prevailing fanaticism and workaholic approach. One of his chosen social roles was to arrange our lunch trips so that we went out every day, but we never went to the same restaurant twice in one month. As I remember, he succeeded month after month in completely natural fashion, seldom letting us know even a day in advance what to expect.

Paul Tardif

Eating lunch with Dick Newson was occasionally a startling thing. I have never been a particularly fast or particularly slow eater; my mother always had just about the right amount of food, so my two older brothers and I never developed a large-family competitive eating style. Newson ate like he had grown up in an orphanage right out of *Oliver Twist*. I remember vividly one time when just the two of us went out together. Our sandwiches arrived at the same time, in the middle of a technical discussion, and Dick and I kept right on talking. By the time I had added some salt and mustard and picked up my sandwich for the first bite, I looked over to see Dick cleaning his plate!

Another one of the constructive contrasts within the CP group was between the general group of crazies and the long-service IBM FEs—Charlie Weagle, Ray Grein, Larry Estelle, and Ed Murray. For several months in the beginning, we (the crazies, of course!) had fun convincing them to unbend a little, come to work without a necktie, wear a colored shirt, or be a bit late once in a while. I think it was Ray Grein who finally explained—he had been an IBM field person for so long that he didn't *own* anything except white shirts and dark suits! The only other clothes he had were ragged jeans with paint on them, sweatshirts, shorts, T-shirts, and a bathing suit.

The 545 Tech Square building had a concentration of "personality" people, going well back to my early days at the Scientific Center. Don Hatfield was the only person I ever knew who had not a single hair on his head and owned something like two dozen white silk Nehru shirts—all the same except for different embroidery patterns in the trim or subtle damask patterns in the fabric. John Ravin was one of the easy-going younger crew at CSC, and he would occasionally bring in Bumble, his 300-pound Saint Bernard "puppy". The hazard was that Bumble loved to be friendly—by leaning on you as you patted him! I spent a good fifteen minutes one weekend afternoon pinned to the door of John's office when Bumble was feeling particularly neglected. Ed Hendricks and I spent some time trying to find a really comfortable yet durable pair of leather sandals to wear around the office in the summer.[176]

---

[176] Eventually we found a new leather goods shop on the outskirts of Harvard Square which made custom-fitted sandals for about 25 dollars a pair. By a completely unlikely coincidence, the owner of that store, which went out of business in the early 1970's, is now my brother-in-law.

There were also several incidents that could only have happened in the environment of Cambridge, Mass., and the Viet Nam War situation. One afternoon I was sitting in the CSC reception area reading a newspaper when a stereotype walked in the door—a medium tall man in a nondescript suit wearing a trenchcoat. He looked around furtively, then stepped up to the receptionist's desk, pulled out a wallet and said something like, "Good afternoon, I'm so-and-so from the FBI and I'd like to ask a few questions," as he flashed his badge! At the time our receptionist was Lily, a tall black woman from Haiti with a mischievous sense of humor. She watched the scene with a twinkle in her eye, but she couldn't stand it very long—she broke up laughing before she could say a word in response. It happened to be a completely innocent visit; one of the women who had worked at CSC for a while had applied for a job which required a security clearance, and the FBI was merely checking her references.

On another occasion we almost had an in-house protest. Among the early users of CP-67/CMS were both the National Security Agency and the CIA; the fact that the DAT hardware isolated each user in his own address space was viewed as a powerful system security feature. One time in 1970, I think, the CIA sent two of their people to Cambridge to talk about something that Ed Hendricks had developed or was working on. In the atmosphere of the time, none of the technical people at CSC, especially Ed, wanted to talk to them at all! Ed stormed around the halls muttering "damned spooks!" for half an hour or more before Craig Johnson and Norm Rasmussen were able to coerce him into the meeting. Even more amazing is that they *were* spooks; there was a man and a woman, both of slightly below-average height, average build, average everything! You could stand and talk directly to them or study them for five minutes or more, but if you turned around there was nothing to remember and nothing to describe; they were effectively invisible.

As time went on, there were other hazards associated with that era. The IBM Branch Office near Harvard Square was bombed once or twice, luckily causing little more than broken glass and some outside structural damage. One afternoon in the spring or summer of 1972 (or late 1971, I'm not sure), I happened to look out our third-floor window at 545 Tech Square, only to see the whole Project MAC and MIT AI Lab crew, from the eighth and ninth floors, walking out into the parking lot and turning around to look back at the building. I quickly notified Dick Newson and Tom Rosato, and all of the IBM people very shortly joined everybody else outside. Project MAC had received a bomb threat, but nobody had thought to notify the other tenants of the building!

Tales of the IBM 3704/3705

My first run-in with the IBM 3704/3705 Programmable Communication Control Units (PCCUs) was sometime towards the end of 1970, I think. The CSC work on S/360 communications was apparently recognized within the rest of IBM, since the controller hardware group in Raleigh, North Carolina, sent a couple of people up to Cambridge to talk to us about their plans for a new front-end controller product. They described it as a state-of-the-art, multi-line communications control unit, capable of supporting many different line types and protocols, "programmable" through user-specified parameters. We called it "the great step forward into the past!"

Some of the earliest IBM data communications work had been done next door at MIT as part of the CTSS development, using a beast known as the IBM 7750—a programmable communications control unit so difficult to handle that there were reportedly fewer than a dozen people who had ever programmed it successfully! Ed Hendricks, Craig Johnson, and I had all talked, at one time or another, to the one person at MIT who ever did 7750 programming. We all considered the S/360 2701, 2702, and 2703 hard-wired controllers a real benefit; they made it possible for highly intelligent people to write communications programs. The IBM 7750 and its brethren required a genius specialist.

What goes around comes around, I guess. In the summer of 1973, after finishing up the virtual and real channel-to-channel adapter (CTCA) support, my new device support role landed me the task of supporting the 3704 and 3705 controllers in VM/370-CP. When the 370X PCCUs were first announced, VM/370 would support them in 270X Emulation Program (EP) mode only. In addition, there was no EP utility support in CP or CMS, so it was necessary to run a DOS/VS or OS/VS system under CP to build the 370X program image and load it into the controller. This was awkward, at best; if the EP was not running when VM/370 first came up, the hardware would not respond to the individual device addresses for the emulated lines, so CP would mark all of them OFFLINE. Bringing up the EP initially, or restarting it after a CP reIPL, was a major pain for the system operators.

The first problem to be solved was what to do about the Network Control Program (NCP) and Partitioned Emulation Program (PEP) modes. One of the stranger presentations I ever made was an "exploratory" discussion of possible VM support for the 370X, to one of the SHARE project groups at the Fall 1973 meeting (in Denver, I think). I still felt a bit out of place in front of a large group, at the ripe old age of not yet 25, and I was expected to somehow talk about future VM/370 support possibilities without pre-announcing anything! I had prepared a brief overview of the various 3704/5 program modes, some material on the problems unique to VM/370, and a set of four or five alternatives for the VM-CP support (all but one of the alternatives were "smoke"; we had already decided to go for full support in CP and nothing extra in CMS).

Like a trooper I got up in front of a full meeting room and worked my way through the pitch. Every leading question or plea for a response was met with dead silence. Even at the end, fifteen minutes or so ahead of schedule, I got just one question—on some basic matter which I had touched on briefly in the first five minutes! Everyone but me seemed to be satisfied with the session—I was a nervous wreck. That was one of my first experiences with what you might call "developer's time warp": you're planning things for two years out, working on things which won't be available until next year, supporting features that were shipped last month, and helping the users take advantage of features that were new last year, with user programs that were written two or three years ago.

The actual development of the 3704/5 support was also something of an adventure. At the beginning of the project I had estimated, fairly accurately, how much work would be required to complete it. Unfortunately I had very little experience in translating the amount of work into the corresponding amount of elapsed time. The support was originally aimed for VM/370 Release 2.0, but we didn't get it out until Release 2, PLC 05.

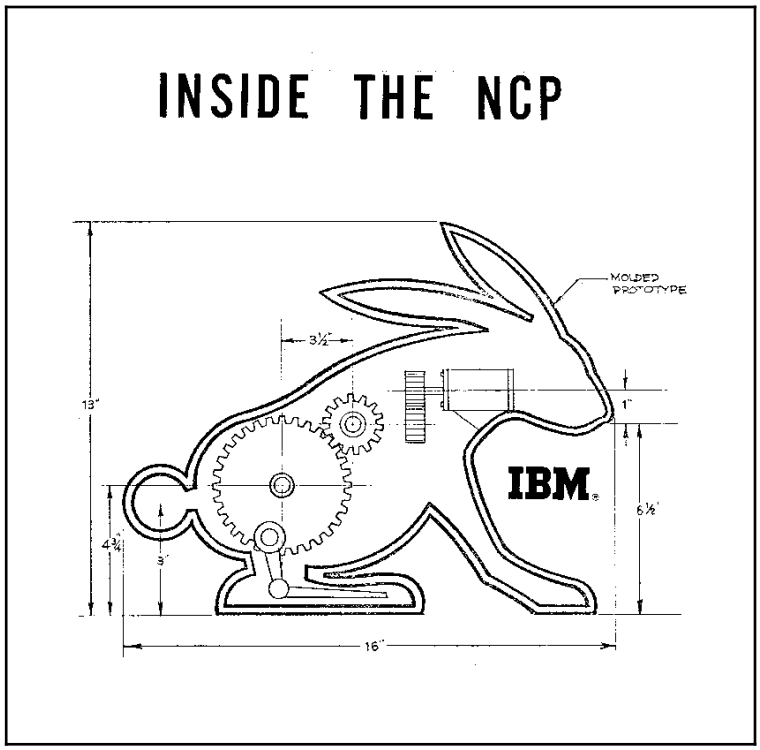
Along the way we had to develop some cooperation with the EP/NCP software and the 3704/5 hardware groups in Raleigh, since we needed to include VM/370-specific installation files on the standard distribution tape for the 370X utilities. We also enlisted two or three people from the mid-Hudson valley (or White Plains, I'm not sure) to do the bulk of the 370X generation support (EP/NCP/PEP program generation, porting the 370X macro assembler and macro library from OS/VS, etc.) in CMS. Bob Downs had to put together a limited version of the OS/VS Linkage Editor and get it running well enough in CMS to build the 370X load images, despite the fact that CMS didn't support all of the OS features which were required. I spent my time restructuring the VM-CP console and terminal support routines to provide a common device interface, trying to understand how to "speak NCP" from within CP, and doing the 370X program load and dump support.

At one stage of the project I discovered that it was possible to put an IBM 2741 terminal into a state where it would gleefully ignore all characters sent to it from the host, without reporting an error or causing any detectable line event. On several other occasions I was able to take down the

S/370-145 multiplexor channel, running either a pure NCP or a PEP in the 370X. That one uncovered both a bug in the model 145 channel microcode and a problem in the design of the 3705 Type 1 Channel Adapter. By that time, fortunately, we had established an unusual amount of credibility with the hardware development people. After an initial visit to verify that there was a problem, they stopped the production line in Raleigh, assembled two channel adapter cards with a probable fix, and flew them up to Burlington in the hands of one of the design engineers. An afternoon of weekend testing verified the fix, and they let us keep the new cards while they "worked the process" to release the fix as an Engineering Change (EC). For several months there was a sign taped to the inside of the 3705, warning the CEs that there were two cards that could not be replaced!

We did eventually finish the support, including the pre-SNA support for NCP and PEP mode, but I caused a lot of trouble for management by missing the schedule so badly. During the fall months of 1973, I worked a total of eight weeks of "scheduled overtime" at premium pay, a rare thing for exempt employees. IBM policy was that you should be able to do your job well working the regular hours of 8:45 AM to 5:15 PM; a healthy family life was necessary for good job performance. Any project which required too many extra hours was considered a planning failure, and the pressure was put not on the troops but on the managers!

After the 3704/5 support was released, I had the opportunity to go back to the SHARE VM/370 Project (August, 1974, in Denver?) and explain what it was that we had done. I've included a copy of the presentation material in the "CARE package" of VM/370 memorabilia.



Once again I misjudged the audience, but the presentation still felt better than the first one. That session was the scene of another coincidence; during the once-around-the-room introductions at the beginning, I discovered Ron Zeilinger, one of the small crew of MIT Computer Center people that Ed Hendricks and I had worked with in 1967!

Answers to some questions you've asked

PER and VMTRACE: There were several factors involved in the decision not to ship the support for Program Event Recording, in addition to the problem with the Model 145 microcode. The prototype that I developed was aimed toward natural support of the PER hardware features, rather than toward a well-rounded virtual machine debugging or trace facility. Many of the basic PER capabilities were functionally equivalent to the old CP-67/CMS TRACE command features, but the CP-67 TRACE capability offered some things which PER did not, and vice versa. In the best of all possible worlds we would have used the PER hardware features for many of the basic functions, used code or techniques from the CP-67 TRACE facility for some extended functions, and added new functions based on intelligent manipulation of the PER features. Unfortunately we had neither the calendar time nor the resources to do the complete job. We reluctantly made the decision to ship only the "traditional" TRACE functions. Releasing both capabilities would have created a very difficult documentation task, at best, and a lot of user confusion.

The VMTRACE capability that appeared in Release 4 is not something that I had any direct experience with, but it is vaguely familiar. Dick Jensen is almost certainly the author. The problem that it was intended to address started with the 3704/5 support in Release 2 (PLC 05?) and was aggravated with the advent of remote 3270 support—the VM/370-CP internal trace table became very difficult to interpret. There were a lot of new event codes introduced to handle both the NCP Version 2 support and the BSC remote 3270 support. At the same time, VM/370 was being used in larger system configurations to support much larger user populations, thus creating more trace entries per second. Even with the very large trace tables which were available in large-memory systems, the real clock time history recorded in the trace table was seldom more than a few minutes.

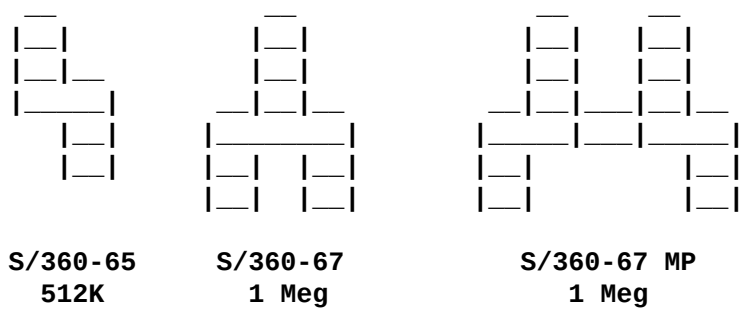
You may or may not remember that the CP internal trace table was originally released as an optional feature. We strongly recommended that users include it, but it was possible to remove the function when you were running in a small system (256K-512K nominal). I was one of the unfortunates who had to try to find a CP problem in a dump without a trace table—very close to being a lost cause, even for VM/370 Release 1. I don't remember exactly when we gave up on the optional status, but it was relatively soon after the initial release; the IBM Field Support organization let it be known that VM/370-CP without the internal trace table was not serviceable. Nonetheless, the facilities existed within CP to run without creating trace table entries. These hooks were first used internally on an experimental basis, in cases where we needed a longer history of selected events rather than a full record of CP activity. The DMKVMT module was most likely developed as an internal system test or development test tool, then extended to include the print, data reduction, and statistics support.

Systems at the CSC: When I first started working at CSC in the Fall of 1968, we had a 512K S/360-67 in a machine room which occupied the northwest corner of the third floor at 545 Tech Square. The small machine room which was in the center of the CSC (west) half of the fourth floor contained the IBM 1130/2250 Model 4 combination. Sometime in 1969, probably as a collateral event to the separation of the CP-67 support group, the CSC machine was moved down to a new machine room which occupied the west half of the second floor, along with some office space for the operations people and for Fritz Giesin, our "modelling engineer" (more on Fritz, later). We needed the extra room to expand the disk storage, add a paging drum or two, and add extra terminal lines. Sometime in 1970, I think, we also picked up a second S/360-67 CPU and some more memory—it may have been the machine used by Lincoln Lab, but I'm not sure.

The second CPU was not very useful at first; we did not have the extra channels and I/O equipment to set up two full systems. The first thing we did was to add some of the memory to

the original CPU, while Fritz tried to figure out how to make a CP-67/MP system out of two uniprocessors. If you ordered an MP system from IBM, it came with special logic in the storage bus controllers and with a "left" and "right" CPU. The original CSC system was MP-capable as far as the bus logic was concerned, but both systems were "left"-handed. According to standard IBM practice, it was not possible to put the two CPUs together in an MP configuration. As was typical of the CSC crew at the time, Fritz was not one to turn away from an "it can't be done" challenge.

A standard S/360-65 consisted of two CPU cabinets, a "crossbar" cabinet with bus cables and power control, and one to four cabinets worth of core memory. The S/360-67 uniprocessor was similar, but it had an extra CPU cabinet for the DAT hardware and a lot more internal logic to support 32-bit addressing in virtual mode. The physical layout of the systems was something like this:



The maximum memory configuration was 1 Megabyte, I think, due to cable length and timing limitations. When Fritz had finished his investigations and constructions, we had a fully operational S/360-67/MP system that was somewhat unconventional. Instead of a standard IBM cabinet as the crossbar of the "H", there was a 6-foot by 3-foot panel of 1/4" Masonite supporting a neat array of IBM-standard internal cable guides, dozens of flat cables interconnecting the two bus controllers, and the necessary array of relays and switches for power sequence control, memory unit control, and I/O interfaces. Essentially all of the interconnecting hardware was ordered as spare parts and put together according to the basic MP schematics, with some unofficial help from the Boston-area IBM CE specialists. The MP system was ready long before we had the CP-67 support for it; for a long time we ran the collection as two separate systems.

Virtual 67s and Such Things: Alain's recollections are probably more reliable than mine for the period he's describing. He was the major contributor to the virtual 67 work, but Charlie Salisbury and Bob Adair were involved at least on a regular consulting basis. At the time, Dick Newson and Lynn Wheeler were both in the CP-67/CMS Support Group and not working directly in the CSC, though the separation between the groups was more administrative than it was technical or physical.

The system version which Alain referred to as CP-67VE was probably called the "E-System" internally, leading to the lettering nomenclature which eventually generated a "G-System", "H-System", and "I-System". As I remember it, the lettering started because you had to be able to tell which CP you were talking to from the terminal. When you had a "stack" of virtual systems, you had to work your way down to the correct CP or CMS level by multiple hits on the old 2741 "ATTN" key. Each system we put together was given a different version of the "CP" prompt so that it was possible to tell where in the stack you were. Not all of the CP-67 versions were in the same sequence; the virtual MP support was based on the virtual 67 branch and never had any of the virtual S/370 support.

The first release of the VM/370-CP product was a rewrite which contained 5% or less of unmodified code from any of the CP-67 versions. My memory is a bit vague, but I think that the last system Alain produced was "CP-67H", which ran on the System/370 machines but did not support any of the newer I/O devices. Paul Tardif started the I-System work, and I joined in a little bit later. The remarks on 2K pages and 64K segments I'm not sure about; it is only marginally possible to support 2K paging on a S/360- 67, but I do have some crazy recollection about an attempt to do so. Alain may have been referring to the need to interpret the additional control register bits which were defined in the S/370 DAT architecture.

As an optional feature the S/360 Model 67 would support 32-bit addressing in virtual mode, a feature which was required for TSS/360 if I remember right. The virtual 67 support in CP-67 included the 32-bit option, but neither CP nor CMS were ever modified to take advantage of it. The S/360 physical memory address bus was never more than 24 bits. The early definition of the S/370 DAT architecture, on the other hand, provided for both 24-bit and 31-bit addressing in virtual and real mode. One of the problems I had to deal with in designing the VM/370-CP DMKVAT support was the possibility of eight different combinations of page size, segment size, and page table format. After some studying of the actual bit positions and values, I came up with a table-driven scheme to handle all of the possible combinations without a performance penalty. There was one section of code which I labelled "TORNADO" because it dealt with the situation of an "in-flight" change in the page/segment/page table formats—it was necessary to throw away all of the existing shadow tables and rebuild them in the new format.

When the S/370 Advanced Function announcement was finally made, the 31-bit addressing capability was no longer included. Nonetheless, all of VM/370-CP was perfectly capable of running 31-bit virtual machines as long as the real storage was no larger than 16 Megabytes. We had also been very careful in the CP control blocks to allow eventual extension to 31-bit real addressing; the real page frame table and the I/O CCW formats were the only areas that would have required changes. The other hooks and holes we left were for multi-processor support and full multi-path I/O support—more things that might have been done much earlier and, perhaps, much better.

Of Editors and Ned: I'm not familiar enough with the realm of UNIX[177] add-ons and add-ins to know whether the UNIX "ned" is related to mine. The CMS version of Ned was completed in the first half of 1970, and I never had enough free time to go back to it and do any major extensions. One of its early virtues was reentrancy; it would run happily in a shared segment. I'm pretty sure that the VM/370-CMS version of it was included with the "July 5 System" distribution, which continued the spread of Ned within IBM. For quite a few years I was subject to periodic double-takes or attacks of paranoia when I would answer the telephone and hear something like "Hello, this is so-and-so in Corporate Legal in Armonk...", only to find out that they were avid users of Ned and needed some support. One thing that it apparently did better than any of the other CMS editors was to handle text editing, rather than program editing.

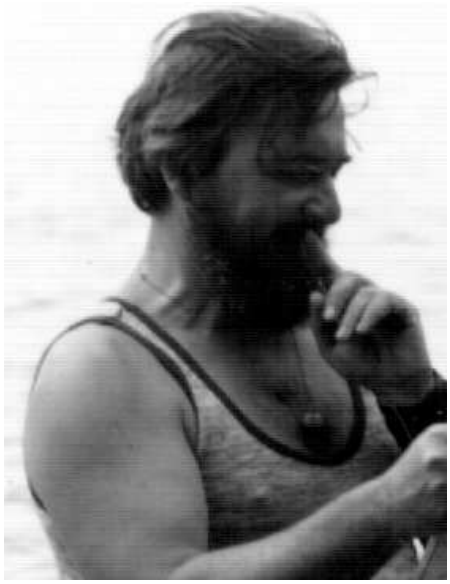
On another occasion I was flabbergasted by Stu Greenberg, sometime late in 1970. He came to me with "a little Ned macro problem", then showed me a 2400-line macro program which took an all-upper-case file of word occurrences, six or eight words per line, and reformatted it into a one-word-per-line, leading caps index in alphabetical order. This for my poor little in-storage macro processor with a lousy syntax and "buggy" conditional handling which was less than 4000 bytes of code! One lesson which I learned well from Ned was, "Don't make assumptions about usage." The users of anything you produce will almost instantaneously discover five or more

[177] UNIX is a trademark of AT&T Bell Laboratories.

things which it was never intended to do, which almost work, and/or they will immediately use it at two or more times the intended maximum load or configuration.

More Personal Glimpses

Fritz Giesin (I hope I have the spelling right) was one of the lesser-known heroes of the Cambridge Scientific Center. His title was something like "Models Engineer", and he was sometimes known as the only CE in IBM who had his own budget. Fritz was the hardware person who actually assembled the DAT box for the S/360 Model 40 which was used to develop CP-40. For Ed Hendricks and I, Fritz designed and put together two or three generations of synchronous modem eliminators to support our BSC communications work. He was also the creator of the interface between the Sylvania Data Tablet and the 2250 Model 4 scope which allowed SketchPad III to support high-resolution free-hand drawing. (The standard 2250 supported a light pen, but it was necessary to generate a "tracking cross" on the screen to find out where the light pen was; its position sensing was based on comparison with the position of the CRT display beam.)



Fritz Giesin

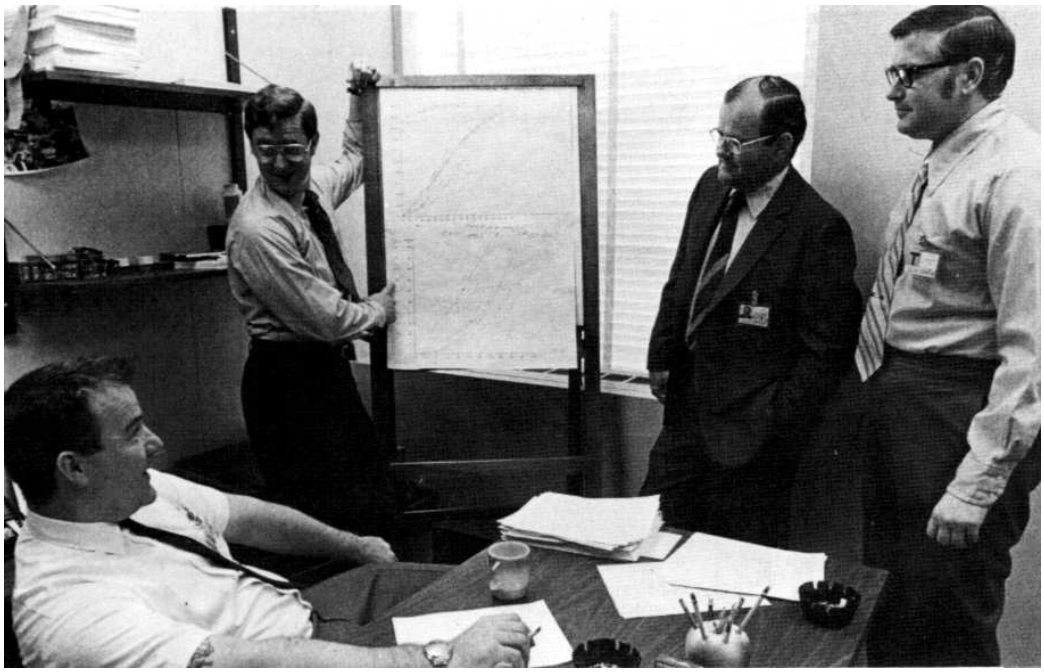
Over the years Fritz contributed a lot of different gadgets and simple things to support both the CSC systems and, later on, the S/370 systems used by the VM Development Group. As we expanded the amount of disk storage on the CP-67 machine, it became more and more difficult to tell somebody where a disk drive was. We had five banks of 2314s, nine drives per bank, and it was possible to switch the controllers between two different channels. The solution was color coding; IBM offered two different color schemes for S/360 equipment, but that clearly wasn't enough. One weekend Fritz and a helper took all of the exterior panels from three of the 2314 disk banks to an auto-body paint shop over in Somerville and had them painted in three different colors! From then on it was easy to explain—"Go over to the Orange bank...", or red, green, blue, grey!

When it was necessary to put together some special piece of gear, be it a coaxial cable patch panel for local 3270s or the complete arithmetic unit which he once added to a "surplus" 2250 Model 1, Fritz was more likely to go down the street to Eli Heffron's Solid State Sales (the local electronics surplus/discount house) or to stop at the local Radio Shack than he was to order from the normal distributor channels. For mechanical assemblies or high-power bus bars, Fritz had some arrangement with MIT which let him use the machine tools and power equipment in the hobby shop. He enjoyed learning new techniques and attacking new problems, and he always found some way to do what we wanted—just the right sort of magic to provide engineering "backup" for the CSC, CP-67, and VM/370 software people!

Philosophical Matters

Development, distribution, and support for VM/370, in the Cambridge and Burlington days, was handled in a variety of ways which were very unusual for IBM products. Some of the procedures and attitudes grew out of the group's organizational history (DPD-HQ and DPD-IM&D), but many of them were the result of individual people's advocacy and obstinacy. It's a little tricky discussing specific things, because I am not at all sure that I know exactly who was involved in a lot of the important decisions. In several areas I was one of the torch bearers over time, but I really don't know if I was ever responsible for "lighting the fire". The areas I'm referring to include these:

Source Level Distribution: This was probably a follow-on from the days of the Type III Library and CP-67/CMS, because we were aware of the many valuable contributions and extensions from other groups within IBM and from the user community. As you have mentioned, it was an important factor in the acceptance and eventual success of VM/370. Selection of standard S/370 Assembly Language was also a factor; Dick Newson, Tom Hennesey, and Dick Meyer had to resist some fairly strong pressure towards using PL/S as the development language.

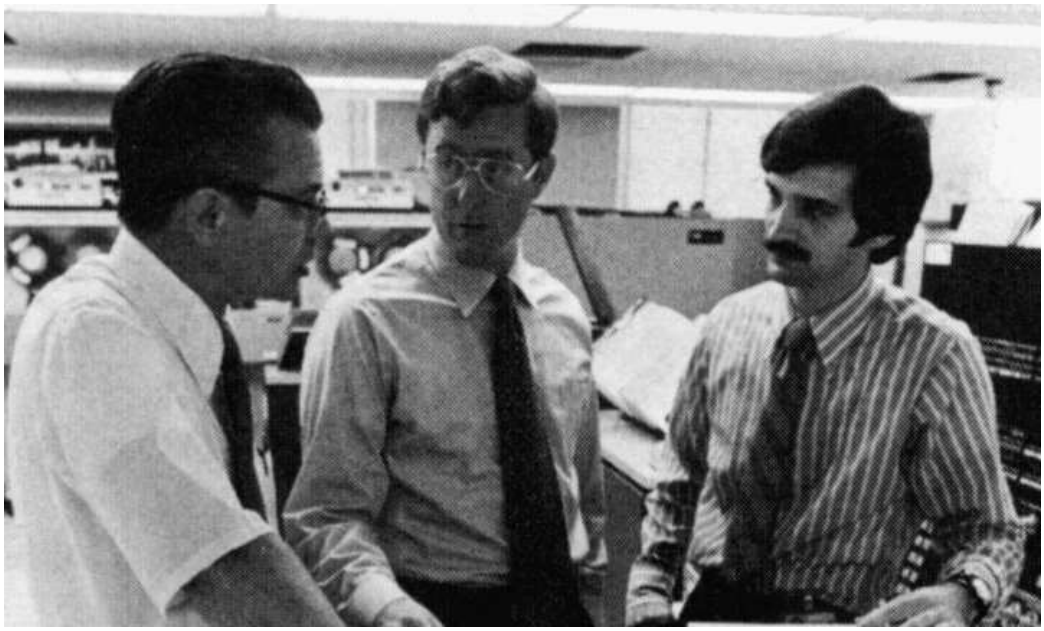


Ed Christiansen, Larry O'Dell, Tom Hennesey, and Ben Pettersson

Distribution of the Development Tools: By this I mean both the UPDATE facilities, VMFASM, VMFLOAD, VMFMAC, etc., and the load lists and control files necessary to build the VM/370 components. The decision to ship the system with all of the tools was somewhat separate from the decision to ship the source code. The alternative which was discussed was to ship only "finished" source modules—*i.e.*, source code with all of the updates applied to the base. It would have been possible to add changes on top of the basic source, but it would have been much more difficult for users to make changes and keep track of them.

By the way, the multi-level UPDATE which I originally coded was written to the conventions of CP-67/CMS, not VM/370-CMS. John Xenakis took the "tool" version and did some fairly extensive modifications to fit it into the new CMS structure of command line handling, file system macros, error codes and message handling.

Frequent and Cumulative Update Releases: The Program Level Change (PLC) procedure was completely alien to the normal SDD way of supporting products. It was either invented by the VM/370 support people, with Dom Lacava, Ron Snell, and/or Larry O'Dell as the major advocates, or it was an adaptation of some similar procedure from their Time/Life or BPC DPD-IM&D background. It amounted to a serious commitment of system test and integration resources with the deliberate goal of quick service turn-around. We routinely had as many as four or five different PLC levels of VM/370 "active" at the same time, something which would not have been possible without both source-level distribution and the distribution of the development tool set. The original goal was one PLC per month, with a target APAR turn-around of no more than two PLC cycles (submit a problem on PLC "N", have an official fix incorporated no later than PLC "N+2"). The cycle stretched out after the first year or so only because the number of fixes went down!



Ron Snell, Larry O'Dell, and Dom LaCava

Development/Release Process for "Minor Enhancements": The minor enhancement process is something that may or may not have been apparent to the user community, and I don't remember exactly when it started or how long it survived. The concept was to provide an official mechanism for adding small features or extensions to VM/370, outside of the main development planning, specification, approval, etc., cycle. The guidelines as I remember them were no more than one man-week of development effort and no more than one man-month release effort (documentation, test, etc.). For each PLC cycle there was a limit of something like five minor enhancements, varying based on the amount of normal-cycle fixes and new function included. The CP "SLEEP with Timeout" extensions that I developed were one example, and I think the CMS in-storage UPDATE support was another case. The same process was used, on a number of

occasions, to incorporate features which had been developed outside of the Burlington VM/370 group.

High Availability / High Integrity Approach: The basic architecture of a virtual machine system gave us many advantages in achieving a highly reliable system, but there were some early "traditions" that contributed as well. The separation of function between CP and CMS allowed us to develop CP entirely as a "closed" environment; there was no user code in CP and there was no possibility of user code access to the real system address space. This led to the CP principle of "fail early, fail often"—now recognized as one of the classic approaches to achieving high availability but somewhat radical at the time. The underlying principles were:

1. Nothing which happens in a virtual machine can be allowed to result in a CP failure;

2. No recoverable error in the real machine realm can be allowed to result in corrupted data;

3. Anything which looks like an internal problem (control block inconsistency, "can't happen" code path, pointer errors, etc.) should result in an immediate CP abend.

Part of the trick was to stop CP as close to the failure as possible. The internal CP trace table contained enough data for us to completely reconstruct the code paths leading up to any failure, as long as we didn't cover up the trail by trying to recover.

Another result of the CP approach was that CMS and its file system had to be built to survive a CP failure at any point without corrupting the disk structure. Jim Walsh was the primary file system wizard, if I remember right. He worked on the minidisk I/O and file system code until the only exposure was interrupted execution of a channel program. It was possible to lose data, but it was not possible to lose a minidisk unless the channel or control unit failed badly.

Fix the Problem, Not the Symptoms: This has always been one of my personal commandments, and it was one of the things which was surprisingly hard to teach. Essentially all of CP was critical path code in one sense or another, since all but a few instructions in the dispatcher ran with interrupts disabled. We couldn't afford a lot of special-case checking or exception handling—the structure and basic logic had to fit the problem. As we added support over the first few years, we had to be willing to change the existing structure when new function or fixes required it.

On more than one occasion I had to be quite stubborn to avoid the "minimum fix" approach which was the more normal IBM way of doing business. One of the fairly early PLCs on Release 1 contained an "update" to DMKVAT which replaced the entire module, because early development testing of OS/VS2-MVS had uncovered a basic misinterpretation of the DAT validity checking. Later the same year I ended up rewriting almost all of DMKCNS and modifying all of the CP console I/O modules to make a place for the 3704/5 EP and NCP support and the subsequent remote 3270 support.

Another aspect of the same principle could be expressed as "Don't be afraid of the big change." We started running into a problem with Release 2 because there was a limit on the number of real I/O devices which could be configured in DMKRIO, based on the initial use of 16-bit offset fields in the control block definitions. As people started using local 3270s on very large VM/370 systems, it became more and more of a problem. I came up with a technically simple fix—using the 16-bit offset as a count of double-words instead of a count of bytes—and it only required updating every CP module which referenced the real I/O blocks!

Use the System You Ship, and Ship What You Use: With the exception of the Ned editor and a few IBM proprietary tools, the system which we used every day in the VM/370 group was exactly the same as the released (or soon-to-be-released) VM/370 product. We did not use the elaborate CLEAR/CASTOR source control system which required proprietary terminal hardware. We avoided using PL/S, IBM's proprietary "structured assembly" language. We did, for a little while, use an internal tool called "FL/1" which generated flow charts based on statements embedded in the module source, but we were able eventually to talk our way out of the requirement for flowcharts in the program logic manuals. Most of the diagnostic aids and performance monitoring or measurement tools had to be developed, tested, and documented at close to product quality because of the number of internal IBM sites depending on VM/370 for their own development. Extensions such as the VNET code in RSCS perhaps took a while to get out in product form, but even that made it eventually.

Measured Performance, Measured Reliability, Measured Integrity: These aspects were in line with IBM's stated goals, but the VM/370 structure made the goals more than usually achievable. There is enough of a tale associated with each one to justify discussing them separately.

1. ***Reliability:*** We started monitoring the system reliability even before we shipped Release 1, and simple good luck had a lot to do with VM's early reputation. Sometime shortly after the internal release of the "July 5th System" in 1972, we started numbering the development versions of VM/370-CP and keeping track of the mean time between failures. The code which we shipped as VM/370-CP Release 1 was "System 54"; there had been that many or more different systems between mid-July and the code freeze for FCS (October?). The normal range of reliability for CP systems from System 40 or so onward was 3-1/2 to 5-1/2 hours between crashes—one or two crashes per working day, on average. CP System 53 was on the low end of the range at about 3.6 hours MTBF, but we knew that we only had time for one more system build prior to FCS. VM-CP System 54 was built in exactly the same way as the previous versions, with more or less the same number of fixes. Miraculously its measured reliability was something over 38 hours MTBF, an improvement of a full order of magnitude!

   Both the measured performance and the measured reliability of CP were accepted by the group as requirements for ongoing releases. The system test procedures for the first couple of years included a full regression test and MTBF determination, even for monthly PLC tapes. Sometime after the first 18 months or so, the MTBF testing was dropped from a requirement to a goal; it was seldom possible to measure it because the system wouldn't fail in a full week of continuous testing. The move to Burlington in 1973 and the second S/370 system allowed us to do "non-intrusive" testing; we used one machine as a terminal simulator and measurement vehicle so that we could run a completely standard VM/370 system in the other.

2. ***Performance:*** The ongoing research work in the Cambridge Scientific Center and IBM Yorktown Research was a significant benefit in the pursuit of VM/370 performance. Even before the VM/370 effort started, Don Hatfield had developed a "profiler" capability which would measure and record the address references of a virtual machine during execution. By plotting the results over time against the load map of the program, he could identify opportunities for rearranging the program to improve working set characteristics. As you might imagine, a full run of the profiler was a significant effort; the CalComp plotter on the 1130 system would run for several hours to produce a single output map. During the VM/370 Release 1 period we profiled VM-CMS several times and adjusted the module loading sequence to improve its locality of reference. We also profiled VM-CP, running in a virtual machine, to identify which code paths were the most critical. About 11% of the CP resident nucleus accounted for a large portion of the CP execution time; many of those code paths were the target of Carl Young's first scheduler update.

The well-known scheduler work of Lynn Wheeler was another example of successful "applied research"; his triumph was a remarkable balance between sophisticated algorithms and the code execution time needed to do the analyses. Many of the algorithms which perform best in theory cannot be implemented in practice—the system time needed to execute the algorithm is greater than the time saved by greater precision. This principle was demonstrated quite well by OS/VS2-SVS; their code for real page frame management was much more sophisticated than anything we had in VM/370, but we could sustain a paging rate more than 10 times their maximum! The MVS people took advantage of our experience, but they were saddled with a much less efficient real I/O path.

3. ***Integrity:*** Dick Jensen, our "resident malicious user", should get a lot of credit for VM/370's resistance to penetration. His self-assigned goal (at least originally; it became offical later on) was to find ways of either crashing CP or gaining access to system resources which were outside of his virtual machine definition, using only the Class G "general user" privileges. He came armed with a full set of the CP listings and a wicked imagination. As he found the problems, we had to find ways to fix them. I ended up with the unlovely task of reworking the ISAM self-modifying channel program support to close one of the better-known "windows" into CP. Dick was also involved, I think, when IBM commissioned the Systems Research Institute in California to do a formal penetration study of the VM/370 system. Their study turned up about 15 areas which needed fixes or improvements. Over a period of a couple of years we were able to fix everything except the ones which fell into the "denial of service" category. There was an IBM Research version of VM/370 which handled even that problem, but it was based on the principle of delivering each user a specified level of performance—no more as well as no less, even if you were the only user on the system. The notion of running slowly as a single user on a S/370-168 limited the popularity of that system!

Things That Might Have Been

Through the early years of VM/370 there were a number of projects which either never got to the drawing board, never got truly started, or never got finished. Some fall into the "wouldn't it have been nice" category, while others raised stronger feelings of hunger or desire. I surely can't remember all of the ideas and notions, but I can tell tales of a few of the larger ones. The prototype support for Program Event Recording (PER) is a tale you've already heard; here are a few others.

Remote 3270-BSC Support for NCP Version 2: VM/370 support for display terminals was a round-about development which started with Dick Newson. After VM/370 Release 1 was shipped, he retreated (*via* a promotion to Advisory Programmer) from managing the CP Development group to a technical position in the newly-created CP Advanced Development group. One of his first tasks was to develop the system console support for the new S/370-158 and S/370-168 systems, both of which came with display-mode-only consoles. He created the infamous VM READ/CP READ/RUNNING paradigm for the 3066 console of the S/370-168, then adjusted the display size parameters for the 3158 console.

[Let me digress for a cute tale.] The first time that Dick had a chance to test on a real Model 168, he came back from Poughkeepsie with wonder and greed in his eyes. The 168 was at least three times as fast as any of the machines which had ever run CP-67 or VM/370. Tasks like running VMFLOAD to gather the CP object files into a load deck, which ran for several minutes on our machine, completed so fast that Dick found himself double-checking the results—he couldn't believe the speed! [Back to the main tale.]

After Dick Newson had the display support working for the system consoles, Charlie Weagle (CP Development) took it over and added support for local (channel attached) 3277s. The next step, this time by George Saunders, was to add the BSC communications to support remote 3270s. George started working on the remote support some time after I had started on the 3704/5 EP, PEP, and NCP support, and we shared both an office and an architecture issue. The 3704/5 NCP, even in its pre-SNA Version 2 days, multiplexed the I/O activity of many lines over a single channel address. The existing control block structure for real I/O had to be extended to provide both another level in the device hierarchy and a different kind of device inter-dependency. The status of devices controlled by the 3704/5 was dependent on the status of the controller program, but the 3704/5 "native subchannel" address could be any convenient value. There was no automatic correlation of device and control unit addresses.

George Saunders had much the same kind of problem with the remote 3270 support, since each BSC line could support one or more 3272 cluster controllers, and each controller could support up to 32 display terminals, printers, or both. Somewhere in the shuffle we got only part of it right. The remote 3270 support took advantage of the four-digit DEV references which I created for the 3704/5 NCP support, but we didn't solve the problem of handling a really large terminal population. The four-digit reference used the first hexit (four bits) to identify which 3704/5 or which BSC line the device belonged to. Unfortunately, that limited us to a maximum of 15 or 16 lines and/or 370Xs, and it made it very difficult to combine the two capabilities—*i.e.*, to support BSC 3270s through the 370X NCP. The CP overhead of BSC polling has been a significant performance/response time factor ever since, and it is one thing that could certainly be done better by a smart comms front end.

Direct channel programming of synchronous data communications is messy; it requires a lot of I/O operations, a lot of interrupts, and it is dependent on real-time response. CPREMOTE and all of its successors existed partially to keep the mess out of the CP nucleus. One consequence of that goal was the need for some very "fault tolerant" I/O programming in the service virtual machines, bearing little resemblance to IBM's recommended error detection and recovery techniques. The 3270 remote display support, unfortunately, needed to be within the scope of CP. Having the system kernel depend on the operation of a user-level virtual machine was something that we did not (and I still don't!) believe in. There might have been an alternative, however...

The SYSTEM Virtual Machine: One of the things which never got past the "blue-sky" stage was a notion to actually implement the SYSTEM virtual machine in VM/370-CP. There were a variety of functions which belonged in CP architecturally but did not really belong in the interrupts-disabled, critical path, real address environment of the basic kernel support. Unit record spooling and terminal I/O (distinguished, perhaps, from system console I/O) were the first likely candidates, along with some subset of the CP user commands. Later opportunities might have included SNA support for virtual machines, disk volume management for the 3850 MSS, page migration, shared minidisk support, a file system for CP, etc.

The underlying notion was to create a virtual address space which included the resident CP nucleus and free storage areas, but was otherwise pageable and runnable like a virtual machine. With not too much internal finagling, it should have been possible both to reduce the size of the resident nucleus and to make more of the CPU time available for dispatch competition. The modular structure of CP, and the regular use of an SVC-based calling convention, should have made it possible to get the best of both worlds—a small, fast, highly reliable kernel with a lot of extended services for virtual machine systems.

True "Warm Start" After CP Abend: The possibility of a "selective" restart after certain kinds of CP failure was another idea which didn't make it much past the discussion stage. I should remember the one or two people who were the most vocal advocates, but I can't be sure if it was Larry Estelle, Ed Murray, or one of the CP support people. The idea came up originally in 1974, I think, and I was one of the major skeptics, perhaps unfairly. At the time CP was fairly reliable and, because of the quick dump-to-disk and restart features, highly available. Nonetheless it was very disruptive to lose everything in the virtual machines, especially for large user populations.

The idea was to add another function to the chain of programs which were invoked by a CP failure. It would first examine the CP Abend code to determine if a "warm start" could be considered. Machine checks, I/O errors which affected the channel activity, and certain internal errors could never be considered due to a loss of system integrity. Some number of failures, however, could probably be isolated to a single virtual machine. In those cases it would be (theoretically) possible to "crash" just the failing user, refresh the CP storage areas modified by the dump process, and restart the system after a very short interruption. Since the warm start would not be attempted until after a full dump was taken, it should still have been possible to track down the original problem.

It was a glorious idea, but the complexity of an on-the-fly system integrity check scared me then, and it still does. The design coordination required between the warm restart process and any new function would have put even more strain on the system development cycle. We would also have created the possibility of long-term operational degradation. Each failure and successful restart might require "abandoning" some amount of CP free storage (virtual machine control blocks, I/O chains, etc.) which could not be safely recovered. The more successful the implementation, the more time would likely elapse between "cold" starts, and we might have had to deal with some very interesting long-term effects! (Much of this is perhaps self justification; I wanted it to be possible but I couldn't believe it. Ironically the system which I use regularly now, PRIMOS on the Prime 50-Series, has incorporated a similar "warm start" capability, successfully, for many years.)

Native SNA Support in VM/370-CP: This is perhaps one of the tales which you have been most waiting for, due to the "wonders" of the original VCNA and the subsequent GCS-VTAM support. The original attempt at SNA support started in 1974, as a natural follow-on to the initial 3704/5 EP and pre-SNA NCP support. I spent the middle of 1974 learning about SNA and some of IBM's other communications futures—some which never saw the light of day, others which developed into such things as the Twinax interface and the IBM Token Ring.

As a personal note, that period included my first contact with Jim Cannavino.[178] He was working with the VTAM communications group in Kingston as an educator/trainer, and he was one of the primary internal teachers for SNA and the other advanced communications architectures. His energy and obvious enthusiasm made him a dynamic speaker, and he had a very good grasp of both the subjects and his audiences. We got together on a few other occasions on a personal level, trading "tall tales" of our early days in the industry and with IBM, but our career paths led off in two very different directions. I think IBM is very lucky to have kept him.

---

[178] Editor's note: Dave has confused Jim Cannavino and Jim Cavet here. See his February, 1991, addendum. [MWV].

Overlapping the technical exploration and evaluation that I was doing, others within the VM/370 organization (Ed Christiansen, certainly, and probably Dick Newson as well) were fighting the "company line" which mandated (at the time) that there would be only one SNA access method—VTAM. Since a stripped-down version of VTAM required a working set which was larger than all of CP and most of CMS, combined, we were less than thrilled about the prospects of porting it into CP. Nonetheless, we were convinced that the support had to be in CP, for architectural reasons, rather than in a virtual machine. We got the reluctant go-ahead to develop a "native" SNA capability in VM/370-CP sometime late in 1974 or early in 1975. The combination of VTAM's size, its dependence on an underlying DOS/VS or OS/VS system, and the fact that it was coded in PL/S, all were factors somewhat in our favor.

Along the way, in October, 1974, I started attending the internal SNA Architectural Maintenance Board (the "AMB") as the official VM/370 representative. The AMB consisted of a chairman with dictatorial power and a changing group of people who were empowered only to "assist" his decision making. If discussion could not resolve an issue, the chairman could, by fiat. The board met every three weeks or so for two or three days, generating an inch or more of minutes, action items, status, and correspondence from each meeting. Attendance was seldom as few as 20 and occasionally as many as 35, consisting of several SNA Architecture people (Jim Gray was one of the original architecture "heavies") and one or more representatives from each development group, hardware and software, which was working on SNA-related projects.

The purpose of the AMB was to make decisions and compromises on a timely basis. Any member could surface an issue—architectural, implementation, or a "variance" request—with a guaranteed resolution period of about two months. New issues or requests would be included in the minutes for meeting "N", presented at meeting "N+1", discussed at meeting "N+2", and a decision was guaranteed in the minutes for that meeting. It was a demanding experience, but it was also an invaluable connection to the other activities within IBM. The meetings alternated between Kingston, New York, and Raleigh, North Carolina, with a once-yearly excursion to one of the other IBM locations doing SNA work. The AMB regulars came to be very familiar companions; at least half of the group was staying in some nearby hotel, regardless of the meeting location.

The SNA support which we had mapped out for VM/370 included native CP support for all of the necessary System Services Control Point (SSCP) functions, SNA terminal support for the 3270, 3767, and 3770 (in 3767/LU-1 mode), and a DIAGNOSE interface for virtual machine access to NCP-SNA resources. My five-minute, off-the-cuff estimate was 25,000 lines of code (assembler) for the CP effort; Ed Murray and Mark Dunn's module-by-module detailed analysis pegged the effort at something like 25,154 lines, added, modified, or deleted. The vast majority of the "Utility" support—NCP configuration, generation, image build, downline load, etc.—could be carried over intact from the earlier NCP-2 support. [Remember that we were dealing with an SNA which was significantly simpler than it is today; VTAM had not yet been shipped, TCAM SNA support was just getting started (also in defiance of the one-and-only-one SNA principle!), and there was not yet any real allowance for multi-host SNA networks.]

While I was spending a lot of time on SNA architectural issues and keeping track of the rest of implementation efforts, the other four on the project team (Ed Murray, Mark Dunn, Charlie Johnson, and one other whose name I can't recall) were working hard on the specifications, design, and coding. Through the first half of 1975 we completed a full set of functional and preliminary design specifications, validated our sizing estimates, and got approval to continue the project. Along the way I had to educate a fair number of the AMB people about the unusual problems or "viewpoint" arising from a multi-system system like VM/370. They hadn't yet considered the separation of the SSCP functions from the data-stream functions, dynamic

reassignment of SNA resources between subsystems, or the addressing problems inherent in a large SNA network. For almost all of the "standard" SNA host support it was "one terminal, one application, forever and ever" in the minds of the developers.

In July of 1975, after a month or so of actual coding with no technical surprises, the whole project was cancelled. It was one of the cyclical periods when IBM was under pressure to show quarterly earnings growth, and they had been investing heavily in the infamous "Future System" or "FS" projects. Many of the FS efforts were in trouble; they were targeting very ambitious technology and it was not always possible to do "invention on demand". As a result, a significant amount of the VM/370 group's resources were "temporarily" redirected to helping out with the next generation of machine and system architecture.

In January of 1975 I had missed the AMB meeting in Boca Raton, Florida, because I was called into a two-week task force in Palo Alto, California. The task force was commissioned to come up with an advanced interactive computing capability for office automation, ideally incorporating the best features of CMS, TSO, and ATS (IBM's mainframe-based "OA" offering). During the second half of 1975 and the first part of 1976, many of the "Virtual People" were busy on a series of task forces and architectural reviews. IBM was trying to salvage what it could of the FS technology and hammer it into some implementable products.

The good news was that they recognized that a virtual machine capability would be required in any new system architecture, and that the virtual machine expertise was in the VM/370 arena. Part of the bad news was that we couldn't be doing two or three things at once; VM suffered from the diversion of effort. The other bad news was that the Poughkeepsie technical and management establishment were not particularly comfortable depending on a small group in Burlington, Massachusetts, which preferred to do things their own way, and there was a lot of travel expense involved in our working together. To a large degree our own success was one of the elements which lead to the Burlington closing in the summer of 1976.

MHCP—The M/H Control Program: One of the exciting things to come out of the task force efforts in late 1975 and early 1976 was a dream called the "MHCP"—a ground-up combination of the best of OS/VS2-MVS, VM/370-CP, and VM/370-CMS. Its name came from the internal code-names for the next machine generation; IBM envisioned a series of entry-level "E" machines, mid-range "M" machines, and high-end "H" machines. The E machines were to be the province of an extended DOS/VS and an appropriate VM capability; the M and H machines were the province of the "big boys"—OS/MVS and some form of VM/CMS, VM/TSO, MVS/TSO, MVS with virtual machines, etc.

The E machines were released eventually as the IBM 4300 series. The initial design of the system did not include a S/370 compatibility mode at all. The first pass at the architecture called for a dramatic extension of the Virtual Machine Assist (VMA) concept to a fully microcode-controlled single virtual address space, such that the DOS/VS system would never "see" the real memory space. I personally led two brief task forces which were asked the question "Can you build a VM system for a machine like this?" The first examination took about a week; the second, with a slightly modified E-System architecture, took us less than a day. The answer in both cases was a simple "No." Thus the 4300 systems came out, some time later, with a full S/370 capability in addition to the DOS/VSE ECPS mode.

For the mid-range and high-end systems, IBM was searching for opportunities to take a step beyond the basic S/360 and S/370 architecture. The very high density packaging technology of the Liquid Cooling Module (LCM) and the continuing decrease in semiconductor memory costs were leading quickly to hardware capabilities that could not be exploited by existing hardware

architectures and software. At the same time, users were reaching the limits of the S/370; a sizeable VTAM network on OS/VS2-MVS would use up a full 16M-byte logical address space, while the combination of large "disk farms" and multi-CPU configurations were pushing the bounds of the I/O address space. I don't remember exactly how many different task forces we participated in; there were several passes taken at evaluating different virtual address space control architectures, system software designs, and I/O control architectures.

During this period, many of the inter-group rivalries were set aside or ignored, but we could not escape all of the organizational distinctions. We were able to work directly with the top people from the Poughkeepsie and Endicott system software groups, but there was little if any direct give-and-take between the software people and the primary hardware architecture groups. The hardware design versus software design issues were generally handled by an exchange of documents rather than working discussions. Not too surprisingly, the first version of the DAT architecture for the M/H machines looked like it was designed to run OS/VS2-MVS and not much of anything else.

We were given two questions to start with: "Can you build a VM system on this machine which runs S/370 virtual machines?", and "Can this architecture create a virtual M/H machine?" The first question came out as a qualified "Yes", but the second one was very tough to determine. I remember some active talks with Charlie Salisbury, Bob Adair, and others where we really couldn't decide if the software required to virtualize the M/H DAT architecture would work. The detailed hardware design was based on assumptions about the way it would be used—much like the S/370-135 timer microcode—and prototyping the entire system in our heads or on paper, in the context of a two- or three-week task force, was a major stretch even for the best of us.

After a couple of inconclusive attempts at the problem, another group was convened to come up with a better picture of the operating system which we wanted to run on the "big" machines. The necessary ingredients would include an interactive environment like CMS or TSO, a dedicated application environment like CICS/VS, compatibility for S/370 batch applications and utilities, S/370 virtual machine capability, and enough "native" virtual machine capability to support development, test and non-disruptive installation of software for the new machines. The usefulness of virtual machine technology was by that time well recognized within IBM. Some form of virtual machine support had been added to TSS/370 on an experimental basis, and there were rumors of a similar experiment in VS2-MVS. The first few rounds were not too productive, with an "exchange" of ideas something like these:

- "We'll just run MVS under VM."
- "No, we'll add one virtual machine to MVS and run VM in it."
- "Why do we need MVS? We'll run CICS naked under VM."
- "Forget VM, we'll run TSO under MVS."
- "Forget TSO, run CMS under MVS."

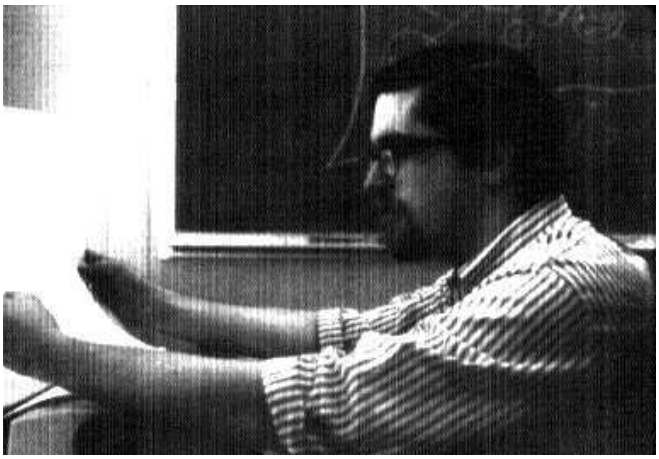
Eventually we settled down to some real work. The idea of the "M/H Control Program" or the MHCP was to apply the separation of function which worked so well for VM/370 to the larger problem of handling multiple address spaces with shared memory, in the style of MVS, in addition to well-defined virtual machines. (VM/370 was great at isolating virtual machines, but it was never very good at letting virtual machines work together. Virtual CTCAs, VMCF, and IUCV are all pretty clumsy and specialized.) Conceptually the idea was to isolate the low-level

I/O and system handling functions into a kernel much like VM-CP, but provide more than one "interface" into that kernel. One interface would be a regular virtual machine, but another would be a dramatic expansion of the DIAGNOSE-style enhanced interface, unconstrained by the hardware architecture definition.

The fairly short period of time we had to work on the high-level design limited the detail we could go into, but everyone involved thought that we could make it work. With the mechanics of system control and I/O device dependencies isolated in the MHCP kernel, it would have been theoretically possible to support new devices, and perhaps even new CPUs, much more easily and more quickly. By the same token the "intelligence" about how many address spaces to create and which areas of storage to share would remain in the MVS control realm. Faint echoes of the MHCP design, I suspect, led to the Group Control System (GCS) component. The original MHCP notion made a lot of converts in the Burlington VM group and in the Poughkeepsie groups, both technical and marketing/strategic. I can't articulate many of the ideas very well, but we were all convinced that there was a real opportunity.

Our timing was less than perfect, unfortunately. Putting the MHCP together would have required a lot of work and a massive coordination effort between several different organizations. The combination of expense and earnings pressure, and the fact that we were (unknowingly) on the eve of the Burlington shutdown, probably contributed to the "not yet" decision. Various pieces of the advanced architecture have since been released as "XA", but I haven't worked with IBM systems closely enough in the past ten years to know what did make it and what didn't. It's always a bit of a thrill to see something which you've worked on or contributed to announced as a product, even if it takes a while to surface.

Conclusion



Dave Tuttle, CSC, February, 1970

Somehow I get a funny feeling when you refer to my "memoirs"—those are written by old people, aren't they? I have been doing computer system development since early in 1967, but I won't be even 41 until next October.

—Dave Tuttle
—July, 1989

Addendum

One of the things which I can probably clear up for good is my earlier confusion between Jim Cannavino and Jim Cavet. Jim Cavet was indeed the SNA/Data Comms training person, working out of Kingston, and we did share several evenings trading tales on the road and at least one very pleasant dinner at his home in Kingston. Jim Cavet might be the one who started with IBM as an 029 keypunch CE in the Chicago or Detroit area and told me a wonderful tale about getting his first manager in trouble by not cashing six months worth of paychecks... but the tone of the tale fits Jim Cannavino just as well if not better.

Jim Cannavino, on the other hand, I met in the early part of 1973 while he was in charge of some aspect of SDD Development (probably related to OS/VS2-MVS, but I'm not too clear on the details) in Poughkeepsie. He was involved in trying to put together a network of the many different VM/370 systems which were being used within SDD to develop OS/VS2. It was typically ironic that the VS2 marketing people were our worst enemies, but the VS2 developers had bet their schedule on using VM's capabilities.

Through some negotiation that I know very little about, Jim had gotten a couple of days of my time made available to work on a version of CP2780 which had been modified, not yet successfully, to provide file transfers between VM systems. The time was late January or early February of 1973. VM/370 had been out in the field for a couple of months. The VM/370 group had transferred from DPD to SDD (January 1, 1973). I had moved from the CP Development Group to the Advanced Development Group, but I was still doing some bug-fixing (major, as it turned out) on the DMKVAT code, and I was working on the DMKCTC virtual CTCA support.

The first trip down to Poughkeepsie was something of an adventure. The initial plan was for me to go down on Wednesday night, work Thursday and Friday, then come back Friday night—I had some crucial personal plans for the weekend. Best-laid plans fell afoul of New England weather—I couldn't get out of Logan until Thursday night. I took Command Airways ("IBM Airlines") from Boston to Poughkeepsie, rented a car, and somehow found my way over to the big Building 701-705 development complex. I'm pretty sure it was my first visit to Poughkeepsie, and I arrived well after normal hours (9:00?). I met Jim there, and he gave me a quick briefing on what they were trying to do, showed me the current state of the code, and gave me a tour of the two-acre machine room at the center of Building 701.

Such strange circumstances beget other strange events; I ran across Bob Wright, an acquaintance from my days as a computer hacker in the MIT Computation Center, running stand-alone on a system in the back corner of the machine room. I also noticed, in passing, a very frustrated programmer trying to get some VS2 testing done in a VM virtual machine. Coincidence, again; he was experiencing a serious bug in the virtual DAT support which showed up only with VS2, and I had—earlier the same day!—just found out what it was and how to get around it. We didn't have a load map of his CP, but I was able to find DMKVAT by going through the Program Interrupt New PSW; I worked up an in-core patch, put it in via STCP, and had him up and running in less than 20 minutes. The fix worked fine, and I remembered it well enough to write it down for him from memory, when the VM system went down for some other reason an hour or so later. (Simple to do, in fact, but I may not have been the only one to tell tales about it.)

[On with the main story.] Those of us who remember CP2780 know that it was a truly worthy example of "spaghetti code". Very few people, even then, knew enough about BSC communications to write a reasonably structured line I/O handler. It was so badly organized, in fact, that I thought I had to straighten it out before I could put in the extra features that Jim Cannavino was looking for. Ever the optimist, as a cocky young engineer, I set about doing just

that. Time was not freely available, however; the full day lost to weather couldn't be recovered. Jim was pushing me to stay and work the weekend, because he was not successful in getting more of my time through official channels. My personal plans were not very flexible; that weekend I was scheduled to take my *fiancée* (Susan, my first wife) to upstate New York to meet her parents for the first time. Little did I appreciate the skills of a wheeler-dealer such as Jim, in the pursuit of his goals.

Through some hidden magic with credit cards and expense accounts, Jim arranged the following scenario: I would stay in Poughkeepsie through Saturday, working on the modified CP2780. Susan, on Sunday, would take a pre-paid taxi to Logan Airport, pick up a pre-paid airline ticket, and fly from Boston to Albany, New York. With the rental car that I was supposed to return to the Poughkeepsie airport on Friday, I would drive to Albany, pick up Susan at the airport, then continue on to Syracuse to meet her family. *Mirabile dictu*, this all came to pass. On Monday, a holiday, Susan and I drove back to Boston in the rental car, which I subsequently turned in at Logan Airport. The AVIS Wizard choked, coughed, and spit up a mild note saying, "Return Station Other than Expected"—off by about 200 miles and three extra days.

The magic with expense accounts came next. Jim had told me to put the rental car on a separate expense form and send it to him. That was fine, and I got reimbursed in the normal course of events. However, our regular expense mechanism couldn't handle the fact that I flew down to Poughkeepsie but didn't (visibly) return; they made me add in 210 miles of personal car expense to explain the return trip. While Jim was hatching this scheme, he told me some tales and tricks about expense account handling from the time when he was an SE or CE working out of New York City. (After a holiday-party-interrupting crisis trip got Jim and a co-worker stranded in a hip-deep snow storm in Connecticut, he managed to hide the cost of a ruined $750 cashmere overcoat in a three-day expense report, even though his manager knew it was there, somewhere...)

There was a second trip down to Poughkeepsie on the same mission, but it was an unauthorized one. Jim successfully hooked my "savior" image, and I drove down on a Friday night to work the weekend. As luck would have it, of course, I got snowed in and didn't get back to work until the following Wednesday. This led, inexorably, to one of the best straight lines, from my manager, which I have ever had to pass up. The Advanced Development Group, at the time, was officially being managed by Ken Brooke. The people in the group, however, included both myself, Dick Newson, and some others who were not very manageable. Ken was generally more concerned about the appearance of control than he was adept at handling the technical issues. When the weather kept me in Poughkeepsie for two extra days, I had to call to tell Ken where I was, and I earned a tongue-lashing when I finally got back to the office. Ken was apparently the one who had refused more help for Jim's project, and he came out with the line, "Why do you think I'm here, just to sign your paychecks?" At the time, it was **very** hard to resist answering his question.

The sad part of the story is that I never did get to finish the cleaned-up, networking version of CP2780. We might have had a VM/370 network of some sort much sooner than we eventually did. The other "unfinished symphony" from my early VM-CP days was a ground-up rewrite of DMKIOS; I worked out the I/O initiation path, but I couldn't free up enough time to figure out a clean interrupt service structure. My fast-path version could take an IOBLOK from DMKIOS entry to the actual Start I/O in about 55 instructions, if the requested path was free.

Ah, the good old days....

—Dave Tuttle
—February, 1991