Improving availability of software subsystems through on-line error detection

by L. Koved
G. Waldbaum

A VM/370 program called Auditor detects faults in the operation of computer software subsystems and attempts to restore service as quickly as possible. Through a series of periodic tests, Auditor diagnoses whether these subsystems are operating properly. When faults are detected, service restoration procedures are automatically called, and the persons responsible for the subsystems are notified. The various types of faults are recorded for subsequent analysis.

In large centralized or distributed computing environments, there are many interacting subsystems that provide services, either directly or indirectly, to the users of the systems. Direct services are programs, such as text editors, that interact directly with the user. Indirect services include subsystems to perform functions that are transparent to the user. Such subsystems include computer networking, printing, batch processing, and data archiving. Frequently, these subsystem services are provided unbeknown to the user. To maintain a smooth and efficient operation, a large computing environment must ensure that these subsystems are available and operating when needed.

If for any reason any subsystem fails, many persons may be adversely affected. For example, while waiting for the completion of a batch job or the printing of a document, the users may be totally unaware that the subsystem processing a job has stopped working. Since many persons who use the computing systems are neither programmers nor persons intimately familiar with the details of how the services are performed, they may be unable to find the reason for the delay. It is often assumed that the executives, secretaries, researchers, and others who use the system will learn how to diagnose the subsystem failures. Otherwise, they must depend upon an expert, such as the computer operator, who is trained to diagnose and correct failures. It is preferable for failures to be automatically detected and services quickly restored, with minimal inconvenience to the users.

The detection of failures in computer software subsystems has been largely a manual procedure, requiring detailed knowledge of how each subsystem works. Failures frequently go unnoticed for long periods of time before users complain to the computer operator. The operator must then determine whether the software is indeed malfunctioning; if it is, the operator must then decide how to correct the problem. If a failure remains unde-

tected, the service remains unavailable until the responsible programmer or computer operator notices the problem and corrects it. A problem can last for hours, days, or even months before it is noticed.

Purpose of the Auditor facility

The purpose of the auditing system, known as Auditor, discussed in this paper is to automate and extend the procedures performed by the computer operator. Auditor periodically verifies that each subsystem is

- Present in the system, i.e., has not terminated
- Processing service requests
- Not exhibiting anomalous behavior

A complex computing environment may have many subsystems performing operating-system or end-user services.

Since Auditor is an automated set of procedures, it can rapidly detect problems, take actions to restore service, and notify the proper personnel.

Another purpose of Auditor is to collect data on failures and their manifestations. These statistics may be analyzed to make better, economically justified decisions for future improvements of software and services. Without Auditor, the frequency of software failures and their manifestations would be unknown.

Typical operating environment

Most subsystems are intended to provide services to the users whenever the computing facilities are available, and an objective of a computing center is to maintain high availability of these services. A complex computing environment may be composed of many subsystems, each performing operating-system or end-user services. In the IBM VM/370 Operating System,[1] subsystems are frequently written as separate software systems running in different user IDs or virtual machines.

Services fall into three distinct categories:

- Provided by IBM
- Provided by the computing center
- Provided by the user

IBM-provided subsystems are IBM program products or program offerings that are sold and serviced by IBM. These include computer networking,[2,3] consulting/conferencing,[4] and performance monitoring and control.[5]

Computing center services include those developed and maintained by the local computing center, including file system backup, archiving, system resource monitors, and text-processing services. The functions provided by the computing center may vary from site to site.

User-provided services are those developed by the users of the systems and may be designated for use by specific groups. At the IBM Thomas J. Watson Research Center, such services include special printer services, classroom reservations, and laboratory automation. User-provided services are usually available at the user's own risk, because the owners are responsible for maintaining these services.

Manual operator procedures

In such a computing environment, it is not simple to determine whether a subsystem is operating properly, because of the variety of causes of failure. Failures range from not being present on the system to giving erroneous results. The computer operator has a set of procedures that describe how to detect particular subsystem failures or bottlenecks. If there is a failure, the persons responsible for maintaining the faulty subsystem are located and informed. There may be an involved procedure for locating the responsible person, particularly if there are many subsystems each with a different owner or maintainer. If the proper persons are not found, the operator's orders indicate whether a subsystem is to be restarted. For some

faults, a subsystem should not be restarted without expert assistance.

When the operator detects a failure or restarts a subsystem, the owner or maintainer is informed via electronic mail. Once informed of the problem, the owner or maintainer may determine the specific cause of the failure and correct it.

The use of subsystems in virtual machines has complicated the problem of determining when errors or failures have occurred. Most subsystems available in VM/370 systems have evolved independently of one another, and each subsystem has different operating characteristics. In such a complex environment, manual procedures are subject to operator errors, wrong test procedures, and outdated operation manuals. An objective of Auditor is to automate the operations procedures by maintaining the procedures and lists appropriate to each subsystem monitored.

When a new subsystem is added, tests are selected that best match its desired operating characteristics.

The characteristics of subsystems have been categorized, and tests have been developed to monitor each of the operational characteristics of interest. When a new subsystem is added to the list of those monitored by Auditor, tests are selected that best match its desired operating characteristics. Testing can be divided into two categories: general tests and subsystem-specific tests.

Automatic detection of subsystem failures

Two general tests are applicable to all subsystems:

1. Is the subsystem running on the system? The presence of the subsystem can be verified by testing whether the user ID is logged on to the system.
2. Is the subsystem in a "hung" state due to a software or operating system failure? Subsystems on VM/370 typically operate without a terminal attached. Suppose a subsystem attempts to read data from a terminal, and a person does not log on to the user ID to respond to the read request. The system then logs off the user ID after fifteen minutes. During this fifteen-minute period, the subsystem is unable to perform any processing due to the waiting terminal-read request. A hung subsystem is detected by testing status information maintained by the operating system about the user ID.

The two general tests by themselves are inadequate measures of whether a subsystem is properly and promptly processing service requests. Therefore, it is necessary to test other subsystem characteristics. The tests performed for each subsystem check for one or more of the following:

- *Periodic processing.* A variety of subsystems have been designed to perform their functions on a fixed periodic basis. Auditor checks to see whether these subsystems have done any processing during the past *n* minutes. This test serves as a rough indicator of whether a subsystem has stopped running.
- *Demand processing.* Some subsystems begin processing whenever there is a queued work request. Auditor sends a work request to the subsystem via a message (interprocess communication) and spool files. After 10 to 30 seconds, the subsystem is checked to see whether it has begun processing the request. If not, the subsystem may have stopped running due to an error in processing the current or prior requests.
- *Handshaking.* Other subsystems are able to shake hands. Auditor sends a work request to a subsystem and waits for a reply. The reply may be in the form of a message, spool files, or data written to a disk file. Optionally, once a reply is received, Auditor may analyze it to verify that the result is correct. There is a time-out mechanism for use when a subsystem does not reply or is delayed a long time. The absence of a response typically indicates a subsystem failure. A delayed response may indicate a heavily loaded subsystem or a degraded system due to anomalous conditions or partial failure. Although handshaking uses more computer re-

sources than periodic or demand processing, it provides a better indication of potential problems. Some subsystems that run periodically may process requests for service yet not complete any or all of their work. Auditor uses handshaking to check for this.

Demand processing and handshaking can be combined to form a hybrid testing method. By the hybrid test, handshaking is attempted first. If there is a time-out before a reply is received, there is a test of whether the subsystem has done processing since the handshaking request was issued. The subsystem may be processing other requests and therefore be unable to respond to a handshake request. That is the case for a first-come, first-served batch processor. If a subsystem has not done any processing recently, the problem is probably in the subsystem.

General and specific tests are performed periodically for all subsystems.

A variant of the hybrid method is to test the subsystem to see whether it has done any processing during the past five to ten seconds. If it has, the subsystem is probably busy processing another request. Then handshaking is not necessary, especially for a subsystem such as a batch processor that requires a substantial amount of time to process an average service request. Otherwise, handshaking is performed.

Another subsystem-specific test is to check that files are updated at the proper intervals. Still another such test is to execute locally defined system commands. If these tests fail, a subsystem has probably stopped working.

Automatic subsystem recovery procedures

All of the subsystem-specific tests are implemented as short programs, each performing a requested test. In most cases, the programs are predefined, and Auditor simply passes parameters that describe the expected subsystem behavior. The program returns with an indicator of whether the subsystem appears to be functioning properly. General and specific tests are performed periodically for all subsystems. The intervals between tests are subsystem dependent. The usual testing period for most subsystems is ten minutes. For several services that are critical to the performance and integrity of the system, the testing periods range from one to five minutes.

Whenever a problem is detected, people must be contacted, preferably via computer messages. The Auditor developed at the IBM Research Center automatically informs the responsible programmers and computer operator of problems. For each subsystem, there is a list of primary and alternate persons to be notified of an error or failure. If a problem persists, notification is escalated. The original persons are notified as well as others. This escalation procedure continues until either the problem is resolved or Auditor is ordered to stop monitoring the subsystem. The interval between the time a message is sent and the time at which the problem is escalated is usually ten minutes. This interval may be longer or shorter, depending upon the nature of the subsystem. If people are not available to receive error messages (e.g., not logged on), the messages are automatically sent to the computer operator.

When Auditor restores a service by restarting a subsystem, it sends messages and electronic mail to the persons on the subsystem notification list. If the subsystem is behaving properly during its next testing period, messages are sent to the same persons informing them of the current status. If, however, the subsystem continues malfunctioning, the error messages resume. If appropriate, the subsystem is restarted. Problems and corrective actions are summarized in Appendix A.

One of Auditor's objectives is to assist in maintaining high availability of subsystem services. To achieve this goal, it may be necessary to take actions to restart subsystems that have stopped running. For example, when the user ID running a subsystem is found not logged on to the system, Auditor issues a command to automatically log on that ID. Subsystems that are hung can usually be restarted by several operating system com-

mands. The user ID running the subsystem is logged off and then automatically logged on to the system again.

Once the user ID has been automatically logged on to the system, the subsystem is restarted. These procedures, which are automatically performed by Auditor, are the same as the actions usually taken by a computer operator when such problems arise. In all cases, Auditor sends electronic mail to the owners or maintainers. A sampling of messages to owners, maintainers, and operators is given in Appendix B.

Special detection and recovery procedures

Subsystem-specific procedures to restore services for other types of failures can be added to the collection of programs that perform the subsystem-specific tests. For example, a batch processing subsystem may be delayed because the job is blocked for lack of sufficient disk space. The subsystem-specific testing program detects blocked jobs and issues commands to cancel a current job and allow the next job to initiate.

Where it is not appropriate to automatically restore a subsystem that has stopped working, an expert may have to restore service. In this case, Auditor sends messages to the owners or maintainers, as just described. Then the subsystem may be attached to special or dedicated hardware, and the hardware needs to be reset manually by the expert.

Auditor performs only one restart per subsystem per day. If a subsystem fails twice during the same day, it may require expert intervention to restart it. Repeated attempts by Auditor to restart a failing subsystem are unproductive. Auditor notifies the proper personnel of the problem and indicates that it will not try again to restart the subsystem.

Benefits

Auditor records in journals the subsystem failures and its attempts to restore service. (Summaries of such journals are given in Appendix A.) The Night columns of these journals indicate that there are a significant number of actions taken by Auditor to restore services during the night. Many of these actions were performed on weekends, although they are not shown separately. If there were no Auditor, many of these services would not be available during these times, or they would be delayed. Thus it is reasonable to conclude that Auditor is performing useful services, as shown by the large number of actions it has taken.

Auditor provides these services with minimal resource consumption. Measurements of CPU utilization for accounting purposes for the six-month period from September 10, 1983, through March 10, 1984, show that the average CPU time consumed by Auditor per day during the prime shift (weekdays from 9 a.m. to 5 p.m.) for the three IBM Research Center systems combined was 2.8 minutes of 3081 CPU time. This translates roughly to 1.3 CPU seconds per subsystem per day for each of over 130 subsystems monitored.

The subsystem-specific testing program detects blocked jobs, cancels a current job, and initiates the next job.

Additional auditing functions

Auditor has been successful because it has improved the availability of services. However, it could be further improved in the following ways.

The computer operator or a user should be able to ask Auditor to test a subsystem when the person becomes concerned that the subsystem may not be working properly. Auditor could make the determination and reply to the person concerned. A prototype has been constructed that demonstrates the feasibility of testing subsystems on demand. A command called UP? can be executed by a user to determine whether the network subsystems, the batch processing subsystem, and several locally defined services are functioning.

Auditor should also recognize that subsystems may have particular schedules. Some subsystems

operate during particular hours of the day, whereas others operate only on particular days of the week or on holidays. The current method of handling schedules is to modify a subsystem so that it notifies Auditor when it wants monitoring to begin and end.

Monitoring software subsystems is just one aspect of improving subsystem availability.

Auditor could monitor resource consumption as a way of determining the health of a subsystem. It is a simple procedure to monitor CPU usage and I/O or spooling activity. Resource consumption monitoring works best with subsystems that have known characteristics and predictable minimum or maximum consumption rates. This procedure is done in a primitive manner by the Resource Limiter.[5]

Auditor could place a telephone call to the proper party on detecting a problem. Such messages could be stored in a microcomputer or in the IBM Audio Distribution System.[6] This method of notification is desirable when those who are responsible for the subsystems are away from their terminals.

Once a subsystem has been restarted, Auditor should perform specific tests to ensure that the subsystem has restarted successfully. Additional automatic logons to restore a service can be done per day per subsystem according to specified criteria. Such criteria might include tests that subsystems have been running for more than specified periods of time since the last automatic logon or that subsystems pass functional tests.

Auditor should determine whether a subsystem has stopped working due to insufficiency of disk space or of other resources. Auditor could inform the proper persons or make the additional resources available automatically.

It would be better if subsystems could perform self-diagnostics and report their findings to Auditor. A subsystem is better able to determine whether all of its components are functioning properly than is a general-purpose Auditor. A subsystem has access to all of its internal tables and program interfaces, which are inaccessible from the outside. The diagnostic results returned to Auditor could then be coordinated with the results from other subsystems. Auditor would thus determine possible corrective actions in the case of multiple or interrelated failures.

Other forms of auditing

Monitoring software subsystems is just one aspect of improving subsystem availability by on-line error detection. The principles outlined in this paper can be applied to other computing elements (i.e., processors, memory, I/O devices, etc.), although the methods used to correct the problems or restore services may be quite different.

At the IBM Research Center, the Auditor principles have been applied to the VM/370 I/O subsystem. There is a problem when an I/O device does not complete an I/O operation. Users who are waiting for I/O operations to be completed are unable to continue their work.

When hardware or software for a file migration subsystem[7] fails, users who are trying to access the files are unable to do anything on their user IDs. Since file migration may take up to several minutes, the users are not immediately aware of the problem. A prototype auditing procedure detects the failure and reports the problem. The I/O monitor is also useful when terminals, disks, and tape units have hardware failures.

The ability to monitor teleprocessing communication lines, printer status, and queues appears to be in great demand by both computing center personnel and end users. An auditor could watch communication line I/O counts, I/O rates, and error rates for each line. While the number of line errors is usually recorded by the teleprocessing programs, they usually report neither excessive error rates nor lines that have stopped transmission. The status of a communication line may be critical for

heavily used routes in a communication network. The failure or degradation of any of these I/O devices should be brought to the attention of the computing center staff so that they can remedy the situation. If available, automatic corrective actions are taken.

Printer operation is sometimes troublesome. A user who wants to print a document usually chooses a convenient printer and expects to see the results after a reasonable period of time. Problems arise when there is a large queue for the printer, when the communication link to the printer has failed, or when the printer is not working. Once users realize that the printer is busy or unavailable, they may not know how to reroute the document to a printer that either is working or has a shorter queue. An auditing procedure can detect these problems before the document is sent to the printer and can suggest alternatives. The suggestion can make both the user and the computing resources more productive.

At the operating-system level, auditing can improve the availability and reliability of the system by periodically checking system integrity.[8] Auditing can check for consistency of control blocks and data structures. It can ensure that hardware is operating properly without excessive errors. It can also look for system performance bottlenecks. If a problem is found, appropriate personnel can be notified and corrective actions taken.

Data base systems can also benefit from auditing because their integrity is critical. Auditing is especially important for distributed data base systems.[9] An auditor can determine whether these data bases are synchronized, and it can ensure the integrity of the data. Ensuring the availability of the communication links between data base systems is a major function of an auditor within the data base network.

Auditing is valuable in a distributed computing environment, such as a local area network that contains dedicated server machines. Within VM/370, an auditor is a virtual machine, but in a network an auditor can be implemented as a real machine that communicates with and tests the servers via the communication medium. When an error occurs, the proper persons can be contacted via messages similarly to the way in which they are notified on VM/370.

Some of the auditing objectives mentioned have been achieved in other systems. For example, the No. 1 ESS[10] is an electronic telephone switching system that contains a subsystem that performs audit-like functions. Auditing enables the No. 1 ESS to maintain high availability in spite of hardware or software failures. The No. 1 ESS auditing functions are tailored to the specific needs of a dedicated telephone switching system. Another specialized auditor has been created for improving the availability of a distributed data base system.[9] In yet another case, YES/MVS,[11] a real-time expert system, assists the computer operator of an IBM MVS operating system. YES/MVS monitors job queue space, performance, communication links between computing systems, job scheduling, and hardware and software errors. Expert systems appear to be a new and interesting approach to improving the availability of subsystems and assisting computer operations. Both of these specialized auditors and the Auditor described in this paper are similar in that they have the common basic objective of improving availability. They differ from one another in the goals and objectives of their respective target systems.

Expert systems appear to be an interesting approach to improving the availability of subsystems.

Concluding remarks

Auditor has been used successfully at the IBM Research Center, where it has detected thousands of problems since it was implemented in July 1981. The result has been improved availability of computer subsystems. Usage at other IBM sites has confirmed our findings. If subsystem problems are quickly detected, services can be swiftly restored. On average, Auditor can detect problems much faster than a person using manual techniques. If services are restored before users are aware of mal-

Table 1 Monthly averages for System A averaged over eight months in 1982, twelve months in 1983, and two months in 1984

Year	Logged-Off	Failure	Stopped	Restore2	Restore1	Attempted	Night
1982	26.8	45.3	18.1	17.9	10.3	1.1	22.8
1983	19.7	31.6	22.9	14.6	14.0	2.0	22.8
1984	42.5	52.0	46.5	19.5	17.0	5.0	28.5

Table 2 Monthly averages for System B averaged over eleven months in 1982, twelve months in 1983, and two months in 1984

Year	Logged-Off	Failure	Stopped	Restore2	Restore1	Attempted	Night
1982	36.3	31.5	28.5	15.4	12.0	0.8	5.5
1983	29.6	38.5	36.1	21.3	27.8	4.8	30.9
1984	31.0	55.0	41.5	32.0	36.5	1.0	47.5

Table 3 Monthly averages for System C averaged over one month in 1982, twelve months in 1983, and two months in 1984

Year	Logged-Off	Failure	Stopped	Restore2	Restore1	Attempted	Night
1982	64.0	57.0	43.0	26.0	49.0	1.0	198.0
1983	34.9	54.5	38.2	16.3	32.2	0.6	32.0
1984	69.5	56.5	80.0	22.5	55.0	0.0	50.0

functions, our goal of greater availability has been achieved. Reliability may be improved by quickly notifying responsible programmers of problems. Since a programmer can see a problem more clearly before rather than after a subsystem has failed completely, there is a greater opportunity for determining the cause of failure and correcting the software. Once failure detection procedures are automated, programmers spend less time monitoring their subsystems. If a failure occurs, Auditor detects the malfunction and alerts the programmers. Thus programmers can use their time more efficiently.

With the information about subsystem failures supplied by auditing, proper management decisions are made as to which services need to be improved and how to improve them. The information leads to more cost-effective software development and maintenance and to improved computing center operation.

Acknowledgments

Many individuals have contributed in a variety of ways to Auditor and to this paper. We thank members of the Computing Systems Department, who assisted with information about the operation of VM/370 and the behavior of subsystems. Matt Zekauskas implemented a second version of the I/O subsystem monitor that is used on the IBM Research Center systems. Kevin McCallum wrote a printer monitor that informs the operator when a printer has stopped. Similar line monitors have been implemented by persons elsewhere in IBM. We thank Nancy B. Raisman for reading and editing the revisions of this paper and Judd Rogers for his suggestions for improving its clarity.

Appendix A: Data from Auditor journals

Tables 1, 2, and 3 are three-year summaries of the numbers of problems and corrective actions on three IBM 3081 systems (A, B, and C) running VM/370 at the IBM Research Center. The meanings of the columns are as follows:

- *Logged-Off*—Subsystem user IDs were found not logged on. Auditor took no action to restore service.

Table 4 Sample preformatted messages from Auditor with description of actions taken

XYZ is not logged on, and automatic logging on is not to be done.

```
MSG FROM AUDITOR: XYZ is not logged on.
MSG FROM AUDITOR: AUDITOR will not autolog XYZ.
MSG FROM AUDITOR: Please take corrective action.
MSG FROM AUDITOR: Type AUDITOR ? for info on
MSG FROM AUDITOR: how to communicate with AUDITOR.
```

Auditor issues the command (AUTOLOG) to automatically log on XYZ.

```
MSG FROM AUDITOR: XYZ is not logged on.
MSG FROM AUDITOR: AUDITOR will autolog XYZ at 09:14.
MSG FROM AUDITOR: Type AUDITOR STATE XYZ OFF to prevent
MSG FROM AUDITOR: the autolog.
```

When a user prevents an automatic logon (i.e., the user types AUDITOR STATE XYZ OFF), Auditor informs all concerned persons that XYZ has been logged off.

```
MSG FROM AUDITOR: KOVED stopped AUDITOR from autologging
MSG FROM AUDITOR: XYZ.
```

Auditor performs an automatic logon.

```
MSG FROM AUDITOR: XYZ was not logged on so AUDITOR
MSG FROM AUDITOR: autologged it.
```

Mail is sent to the persons responsible for XYZ, informing them that an automatic logon was performed.

```
To: KOVED
From: AUDITOR 82/02/24 16:14:37

AUDITOR just autologged XYZ
```

XYZ has hung with a terminal read request waiting; Auditor will not attempt to restore service.

```
MSG FROM AUDITOR: XYZ is idle.
MSG FROM AUDITOR: XYZ will be forced off by VM
MSG FROM AUDITOR: in less than 15 minutes.
MSG FROM AUDITOR: Please take corrective action.
```

XYZ was hung; Auditor took actions to restore service—logged off and automatically logged on XYZ.

```
MSG FROM AUDITOR: XYZ was idle.
MSG FROM AUDITOR: AUDITOR forced it off and autologged it.
```

Mail is also sent to the persons responsible for XYZ, informing them that Auditor logged off and automatically logged on XYZ.

```
To: KOVED
From: AUDITOR 82/02/24 16:23:33

AUDITOR just forced off XYZ because it was idle.
AUDITOR then autologged XYZ.
```

A message is sent to responsible personnel when XYZ is logged on the system after it was logged off or hung.

```
MSG FROM AUDITOR: XYZ is now logged on.
```

- *Failure*—Subsystem failures may have occurred, but the subsystem did not log off nor was it hung. Auditor took no action to restore service. This condition is determined through the use of subsystem-specific test programs.
- *Hung*—A subsystem was hung with a terminal read request waiting. The subsystem was automatically logged off within 15 minutes of waiting. Auditor took no action to restore service.
- *Restore1*—Auditor successfully and automatically logged on a subsystem user ID that was found not logged on. This count is separate from the Restore2 count.
- *Restore2*—Auditor terminated the user ID (process) running a subsystem and automatically logged it back on to the system to restore service.
- *Night*—These are Restore1 and Restore2 action data, taking into account actions after 5 p.m. and before 9 a.m.
- *Attempted*—Auditor was unsuccessful in automatically logging on to the system the subsystem user IDs found not logged on. If Auditor had been successful, the action would have been recorded either as Restore1 or Restore2.

The events Logged-Off, Failure, and Hung are mutually exclusive events, as are Restore1, Restore2, and Attempted.

Appendix B: Messages from Auditor

Table 4 gives samples of preformatted messages sent to responsible owners, maintainers, or operators if problems or failures are detected. On VM/370, the information is sent as multiple-line messages in English, rather than programming jargon, to convey the severity of the problem and the Auditor commands. XYZ is the name of the subsystem in the examples of Table 4. There is an AUDITOR command for use in communicating with Auditor from one's user ID. AUTOLOG is the VM/370 command to automatically log on a user ID. FORCE is the command to terminate the running of a user ID.

Cited references

1. *Virtual Machine/System Product General Information, Release 3,* GC20-1838-3, IBM Corporation (June 1983); available through IBM branch offices.
2. *Virtual Machine/System Product Remote Spooling Subsystem Networking Program Reference and Operations Manual,* SH24-5005, IBM Corporation; available through IBM branch offices.
3. N. Mendelsohn, M. H. Linehan, and W. J. Anzick, "Reflections on VM/Pass-Through: A facility for interactive networking," *IBM Systems Journal* **22**, Nos. 1/2, 63–79 (1983).
4. *Cooperative Viewing Facility General Information,* SC34-2151, IBM Corporation; available through IBM branch offices.
5. D. M. Chess and G. Waldbaum, "The VM/370 Resource Limiter," *IBM Systems Journal* **20**, No. 4, 424–437 (1981).
6. J. D. Gould and S. J. Boies, "Speech filing—An office system for principals," *IBM Systems Journal* **23**, No. 1, 65–81 (1984).
7. *IBM 3850 Mass Storage System (MSS) Principles of Operation: Theory,* GA32-0035, IBM Corporation; available through IBM branch offices.
8. G. Waldbaum, *Audit Programs—A Proposal for Improving System Availability,* Research Report RC-2811, IBM Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598.
9. W. Kim, *Auditor: A Framework for High Availability of DB/DC Systems,* Research Report RJ-3512, IBM Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598.
10. R. W. Downing, J. S. Nowak, and L. S. Tuomenoksa, "No. 1 ESS maintenance plan," *The Bell System Technical Journal* **XLIII**, No. 5, Part 1, 1961–2019 (September 1964).
11. J. H. Griesmer et al., *YES/MVS: A Continuous Real Time Expert System,* Research Report RC-10461, IBM Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598.

Lawrence Koved *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598.* Since 1982, Lawrence Koved has been a member of the Computing Systems Department at the IBM Research Center. Mr. Koved is the designer and developer of several prototype interactive conferencing systems that have become the Cooperative Viewing (CVIEW) facility. He is also one of the major contributors to the design and development of the IBM PC XT/370 computer that runs VM/PC and communicates with a VM/370 host. For each of these projects, Mr. Koved has received Outstanding Technical Achievement Awards. He is currently a Ph.D. candidate in the Computer Science Department at the University of Maryland, College Park. Mr. Koved received his B.S. in computer science from Union College, Schenectady, New York, in 1981, and his M.S. from the University of Maryland in 1985.

Gerald Waldbaum *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598.* Dr. Waldbaum is a senior manager responsible for advanced workstation projects and services. His development organization prototyped and helped develop several products, including PC/VM Bond, the Cooperative Viewing (CVIEW) facility, and the VM/370 Resource Limiter (RESLIM). His organization assisted in the development of VM/PC for the XT/370. His service organization provides consulting, internal PC hardware and software sales, and assembly and repair services to workstation users at the Research Center. Dr. Waldbaum also helps manage a company-wide disk and conferencing facility for sharing IBM personal computer information and internally developed software. From 1959 until 1966, when he joined the IBM Research Division, he was a

member of the technical staff at Bell Telephone Laboratories. There he worked on the design and development of the No. 1 Electronic Switching System. Dr. Waldbaum has taught computer science at Iona College Graduate School of Business; he is coauthor of the book *Electronic Switching Theory and Circuits*. He received his B.A. and B.S.E.E. degrees from Columbia University and his M.S.E.E. and Ph.D. in operations research from New York University.

Reprint Order No. G321-5265.