Online Computer Auditing Through Continuous and Intermittent Simulation*

By: Harvey S. Koch

Abstract

A new computer auditing technique, called Continuous and Intermittent Simulation (CIS), is introduced. It has been specifically designed as a compliance auditing technique for timesharing systems that can be used to audit internal controls. CIS is an auditing technique that simulates the instruction execution of the application at the time the application is processing a transaction. All data and input to the application is accessible by and shared with the simulation. This means that the simulation is notified about each transaction that is entered to the application and accesses to the database by the DBMS.

It is not necessary for all transactions to be audited in order to have the capability of performing online auditing. Before any updates are made to the database, or before any output is returned to the users, the simulation can verify the results by executing the appropriate instructions that evaluate the internal controls of the application. If an inconsistency is found, all pertinent information about the system's status can be put into the exception log. The simulation can then choose to use the results computed by the application or by the simulation, or choose not to use any of the results, as if there was no transaction.

Keywords: Computer auditing, internal auditing, online auditing, internal controls, parallel simulation

ACM Categories: 3.50, 4.6

*This research was supported in part by the Office of Naval Research under Grant CNA SUB N00014-76-C-0001.

The author wishes to thank William H. Meckling, Dean of the Graduate School of Management for financial support, and for creating a superior environment for performing research. Bezalel Gavish provided the motivation for this research in computer auditing, and contributed greatly toward enhancing the model through our many conversations.

Introduction

An important purpose of computer auditing is to evaluate the internal controls in an automated information system and verify the system's processing phases and results. Audits detect and hopefully reduce future processing errors through recommended improvements.

As with any audit, it is not sufficient to just verify the results of processing. The auditor must verify that the internal controls within the computer system are sufficient, consistent with management guidelines, and working properly. These controls cannot always be evaluated by only investigating the output results. Sometimes requirements are expressed in terms of how the system should be designed. Also, requirements may be in terms of how an application should perform its functions, or in terms of data processing concepts, *e.g.,* computer security controls, integrity controls, reliability controls, and recovery controls. It may not be possible for the auditor to test or simulate these control mechanisms.

As in manual applications, two types of audits should be conducted — internal and external. The primary function of the internal auditor is to evaluate controls, verify their implementation, and provide conclusions to management [4]. The internal auditor is typically concerned with evaluating the degree to which control standards set by the management are being satisfied. It should be emphasized that the enforcement of daily operating controls is a function of line management, not the internal auditor. Internal audits should not be made at intervals that would constitute a pattern identifiable by a malfactor. Random audits make it difficult for anyone abusing the system to do so without fear of discovery.

The primary function of the external auditor is limited to the exposures to unacceptable accounting practices and erroneous record keeping [4]. The external auditor is an independent agent who objectively examines the system's performance and the accuracy of its outputs in regard to both the way they were produced and the way they are intended to be used. An external audit should also include an evaluation of the effectiveness and efficiency of the internal auditing.

The objectives of any audit are to identify the vulnerability of the system where it exists, evaluate internal controls, verify implementation, and provide conclusions to management. In this article computer auditing from the perspective of the internal auditor is discussed.

The auditing process of computer applications is generally more encompassing than program testing. Some of the objectives of computer auditing are to 1) evaluate the integrity of the system from both functional and security standpoints, 2) evaluate the degree to which control standards set by management are being satisfied, and 3) assess the accuracy of system output. In general, the auditing process attempts to detect four types of errors: incorrect results, security breaches, inaccurate record keeping, and variations from management control standards. The auditing process occurs more than just once. It should occur on an unpredictable recurring basis. It should not occur only when the system environment changes, *e.g.*, installation of changes to existing software.

Computer auditing techniques can be classified by what is being analyzed or inspected in the audit process. Usually the source code or output from the application is inspected. Summarized below are some of the auditing techniques which can be included in this classification. Further descriptions of the auditing techniques can be found in [1,2,4]; more detailed descriptions of the testing techniques can be found in [3,4].

Inspection of source code

Analyzed by the Auditor

Inspection of the source code by the auditor is limited in its value and almost unlimited in its complexity. Desk-checking a program by its author is a tedious job and limited in value without any test runs. The auditor has additional problems: not familiar with the program, may not know the source language, the program may not be documented well, may not be familiar with the computer system, and the program that is being inspected may not be the program which is being used.

Analyzed Automatically

It is possible to have a program analyze the source code of an application. Automatic analysis has been successfully used to detect programming errors, undesirable program design features, and the completeness of test data. For example, multiple entry points into a subroutine is an undesirable feature that can be automatically detected. The completeness of test data can be determined by having a monitor recognize which portions of the application program have not been executed during the processing of test data. However, there are three factors that limit the extent to which automatic analysis can be applied:

1. **Theoretical limitations** — for instance, it is impossible to determine for an arbitrary program given arbitrary input that an arbitrary statement of the program will be executed. This eliminates the possibility of having a general purpose program analyzer.
2. **Semantic paths cannot be distinguished from syntactic paths.** This means that inspection of the source code is not sufficient to determine how the program will behave at the time of execution. For instance, array pointers cannot be evaluated until execution time.
3. **There is no way to determine whether the program meets operational and management requirements.** Inspection of the source code, for example, will not guarantee that operational procedures are being followed with respect to backup and recovery.

Comparison of results

Auditing Around the Computer

Using this method, the auditor compares the results produced by the application to the results computed by hand. The computer is not used as an aid in the auditing process. The only reason the computer is involved in the auditing process is that the application to be audited is installed in a computer. Since the computer is not used in the auditing analysis, many transactions cannot be analyzed, the frequency of auditing is limited, and only printable output can be audited.

Auditing Through the Computer

Most of the currently used auditing techniques fall into this category. The techniques in this category use the computer to produce the results that are compared to the results produced by the automated application. Some techniques use special test data while others use live data. Parallel simulation is an auditing technique in this category which uses live data.

Parallel Simulation — Capabilities and Limitations

Parallel simulation is one of the most successful auditing techniques in practice today. It generally seems to provide the best balance of reliability, resources, time, cost, and conclusiveness of all the alternative techniques, and has been proclaimed as the best method for detection of fraudulent code [2,4]. The parallel simulation process is shown in Figure 1 [4]. The auditor creates a set of programs that simulate selected processing functions of the automated application or the entire application, and any language can be used. Generalized auditing software is normally used because it provides more auditing functions and is less procedure-oriented than most general purpose programming languages. The simulation reads in the same input transactions, uses the same files as the application it is auditing, and attempts to produce the same results. The results from the application and simulation are compared, and exceptions are given to the auditor for analysis.

Even though parallel simulation is widely used and is generally recognized as the most powerful existing auditing technique, it has several major disadvantages:

1. The type of output that can be audited is new master files created in batch processing mode and reports produced by the application. The auditing of direct access updates in a timesharing environment would not be possible by parallel simulation without the simulation having its own copy of the database(s) that are updated as soon as the update transactions are submitted. Parallel simulation was designed for sequential file processing and batched updates, not for the timesharing environment.

2. When an exception is detected, the only action that can be taken is to write a message to the exception log. Because exceptions are discovered after the application processes a batch of transactions, the failure of internal controls is not discovered at the time of failure. Information about the operational environment of the application or the system at the time that the transaction in question was processed can be lost. That is, certain events would not appear in the log or cannot be traced or duplicated. Two examples of such events appearing at the same time as the questionable transaction being executed are: a) the status of all input/output operations generated by the questionable transaction, and b) the status of all other transactions.

3. High input/output overhead is incurred for each transaction at the time of simulation to produce the simulated output, and to read the output files produced by the application and the simulation to detect exceptions.

4. Since an inconsistency is not detected until sometime after the transaction in question has been processed, the system is weaker from a security point of view. For instance, assume that an inconsistency was caused by an illegal penetration of the system, *e.g.*, an unauthorized withdrawal. Not only was the system penetrated, but also the longer the elapsed time between the penetration and detection of that penetration, the longer the system is vulnerable and the longer the penetrator has to disappear or cover his tracks.

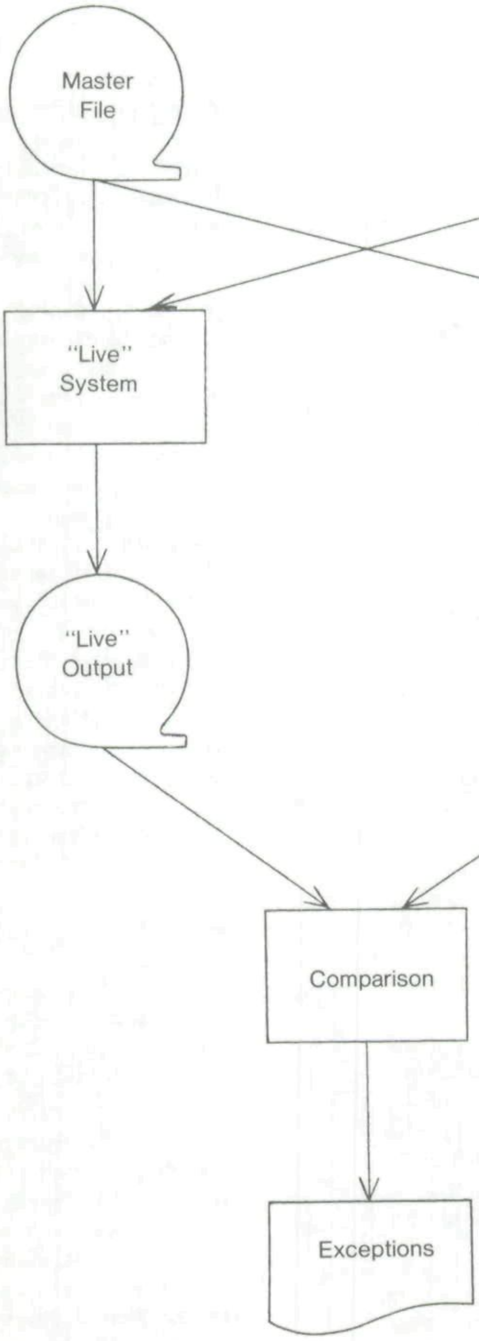

Figure 1. Parallel Simulation Process

Reprinted by special permission from *Computer Control & Audit*, by William C. Mair, Donald R. Wood, and Keagle W. Davis, published by Touche Ross & Co., Copyright 1978, p. 152.

In fact, all of the auditing techniques described by Cash, Bailey, and Whinston [2], which are commonly used to perform computer auditing, suffer from the same major disadvantage — the comparison of results is done offline from the automated application. This means that errors may be caught relatively late, and pertinent status information can be lost. Knowledge of the failure of internal controls at the time of failure would benefit the user, programmers, and the internal audit staff. Immediate knowledge that a result is wrong, and the state of the system at that time can help the internal auditor detect the failure or lack of proper internal controls.

It would be a great improvement to have an auditing technique that would overcome all of the above disadvantages. The next section describes such a technique, called Continuous and Intermittent Simulation (CIS).

Description of Continuous and Intermittent Simulation

Any application implemented on a computer is composed of three elements: the instructions, memory for variables (working-storage), and databases. The purpose of auditing by simulation is to verify that the results produced by an application and the simulation correspond. If they do not, then the discrepancy should be resolved. In a timesharing environment, if a transaction is to be audited, the simulator must check to see that all updates to the database caused by that transaction are correct. For CIS this means that the simulator simulates the instructions of the application that correspond to the internal controls that are being evaluated. The simulation and the application share the same database. While a transaction is being processed, the simulation determines whether the updates are correct. If not, all pertinent information about the operational environment at that time is logged.

In Figure 2 a schematic representation of CIS is provided. The simulation is viewed as an extension of the Database Management System (DBMS) with all input/output of the application, including the reading of transactions and the output of results to users, performed by the DBMS.

The sequence of steps for CIS is:

1. When a transaction is read by the DBMS, the simulator decides whether it wants to audit the application. If not, the simulator will wait for the next transaction. If so, the second step will be executed.
2. Every update to the database caused by this transaction will be checked by the simulator to determine whether there are any inconsistencies. The DBMS can determine each time there will be an update. Before the update occurs, the DMBS will notify the simulator and wait for an acknowledgment that there are no exceptions.
3. Any inconsistencies with respect to the update between the simulator and the application will be logged into the exception log.

It is possible for every transaction to be audited since the simulator will be notified each time that a transaction is entered into the system. In this mode a continuous simulation is performed. Every update to the database will be verified by the simulator. In effect, this is auditing by parallel simulation with the capability of performing the audit online while the transaction is being processed, and before any incorrect values contaminate the database, or are given to the users.

It is not necessary for all transactions to be audited in order to have the capability of performing online auditing. This intermittent simulation is possible because the simulator has access to the same data as the application, and evaluation of an internal control is done at the time the application is processing the transaction, but before the database can be contaminated with inaccurate results.

Any exceptions to one transaction are detected before the next transaction is processed. Therefore, all transactions do not have to be audited for the simulator to determine whether the results the application has produced are correct. For this reason CIS can be used as a test for compliance and as a substantive test. Parallel simulation is only a substantive test.

The selection of transactions to be audited can be based upon sampling, characteristics of the trans-

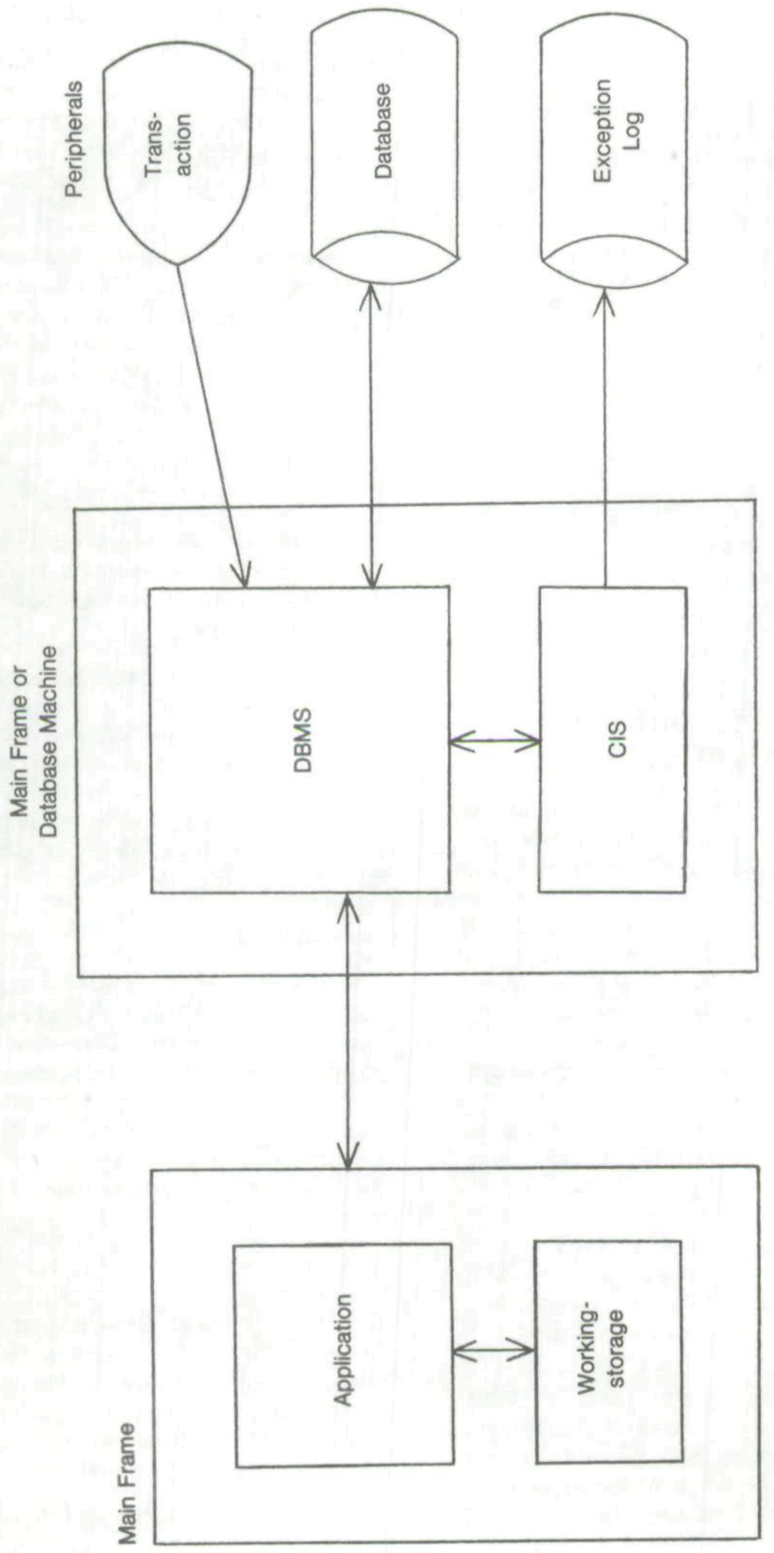

Figure 2. Environment for Continuous and Intermittent Simulation

action, or characteristics of the processing environment. The following are examples of selection criteria that can be used to choose transactions to be audited:

1. Random choice of transactions, *e.g.*, a random number generator can be used to determine whether to audit the current transaction, the next transaction type to be audited, and/or the next time of day that a transaction should be audited.
2. Time of day, *e.g.*, all transactions entered during the third shift should be audited.
3. All transactions submitted at a specific terminal or by a certain individual.
4. Combination of events based upon the system status, *e.g.*, when certain files are open and a certain transaction type is entered.

At this point an example is introduced to explain the role and operation of CIS. The environment for the example is banking where tellers enter transactions through terminals, see Figure 3. There are two types of transactions — deposits and withdrawals. In this example it is assumed that only withdrawals will be audited. The simulator can pick all or some withdrawals to be audited. The selection of withdrawals can be based upon randomness, account number, or teller identification. The steps for the application to process a withdrawal transaction are:

1. verify teller number,
2. verify account number,
3. retrieve account record,
4. verify that the account will not be overdrawn if this withdrawal is allowed,
5. update account record with new balance, and
6. print output at teller's terminal.

Each one of these steps can be simulated and audited, or the entire withdrawal process can be audited. In this example step 1 (verify teller number) and step 5 (verify that the new balance being updated in the account number is correct) are being audited. The sequence of events for the application processing a withdrawal, and the simulator to perform an audit are represented in Figure 4. Note that each time CIS executes it is prompted by a transaction entering the system, output to the terminal, or access to a database. Each time the DBMS receives a request to or from the application, a portion of CIS is executed if the application is being audited.

The major difference between parallel simulation and CIS is that in parallel simulation the results of processing are compared after all of the transactions have been processing, while in CIS the results of processing for a transaction are compared before any results have been output to users and before any files have been updated. This allows CIS to be a much stronger auditing technique than parallel simulation. The added advantages of CIS are:

1. Applications operating in a timesharing environment can be audited for all results that they produce — output to users, and updates to any files. This is possible since any output from the application is audited at the time the DBMS receives it. For instance, in the example updates to the database, in addition to the output to the terminal, were audited.
2. A higher level of integrity is possible since CIS interacts with the DBMS. If there is a discrepancy in the results of processing between the application and CIS, the results by CIS can override the results of the application. In parallel simulation this is not possible. Also, with CIS all pertinent information about the system environment can be sent to the exception log at the time a discrepancy is detected, *e.g.*, the source of the transaction, the time of day, or the users who are interacting with the system.
3. By auditing all changes to the database online, more than just the application can be audited from the security point of view. For instance, if another program attempts to alter a record, this can be detected. Each time the DBMS is requested to access or update a

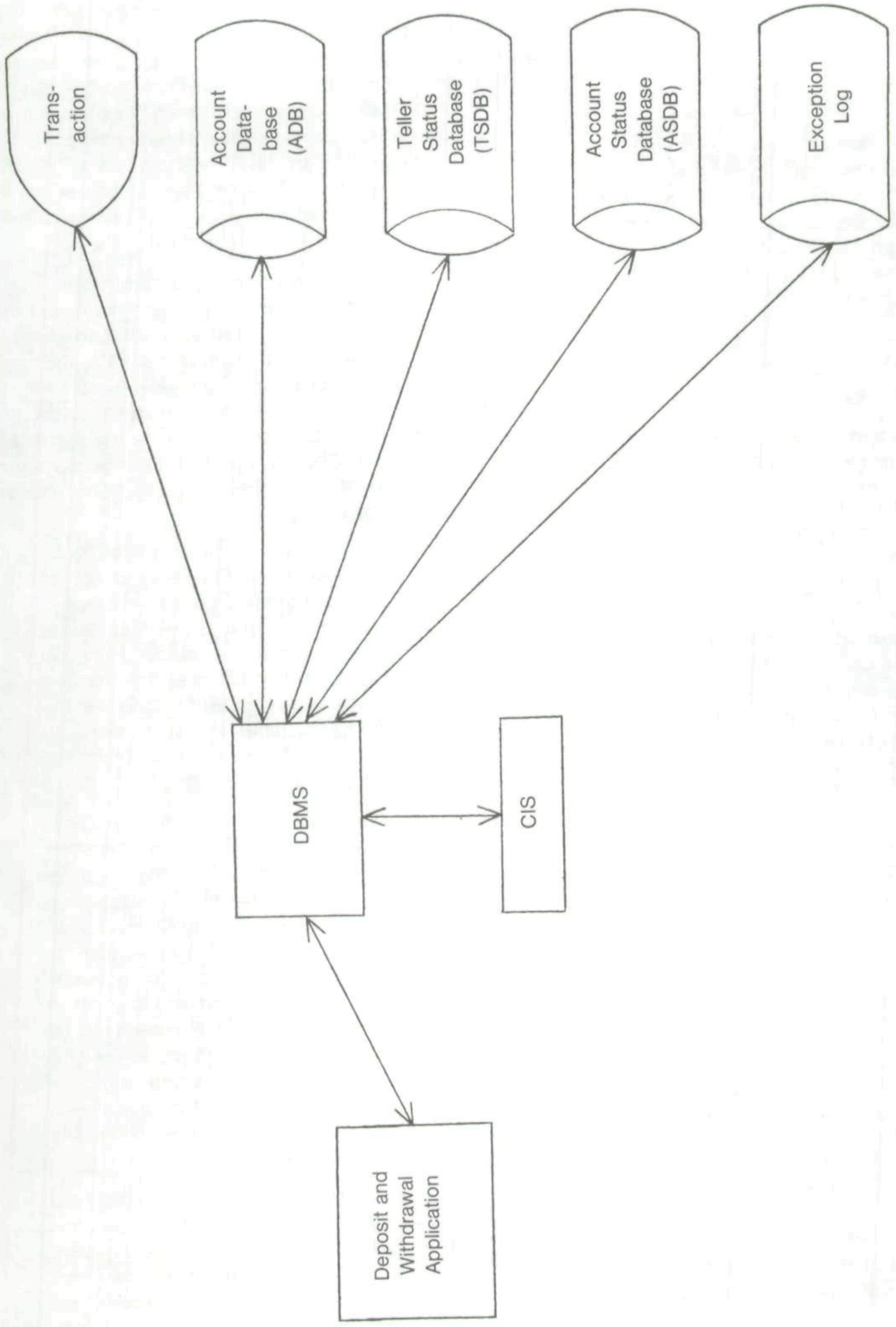

Figure 3. System Structure of Deposit and Withdrawal Application with CIS

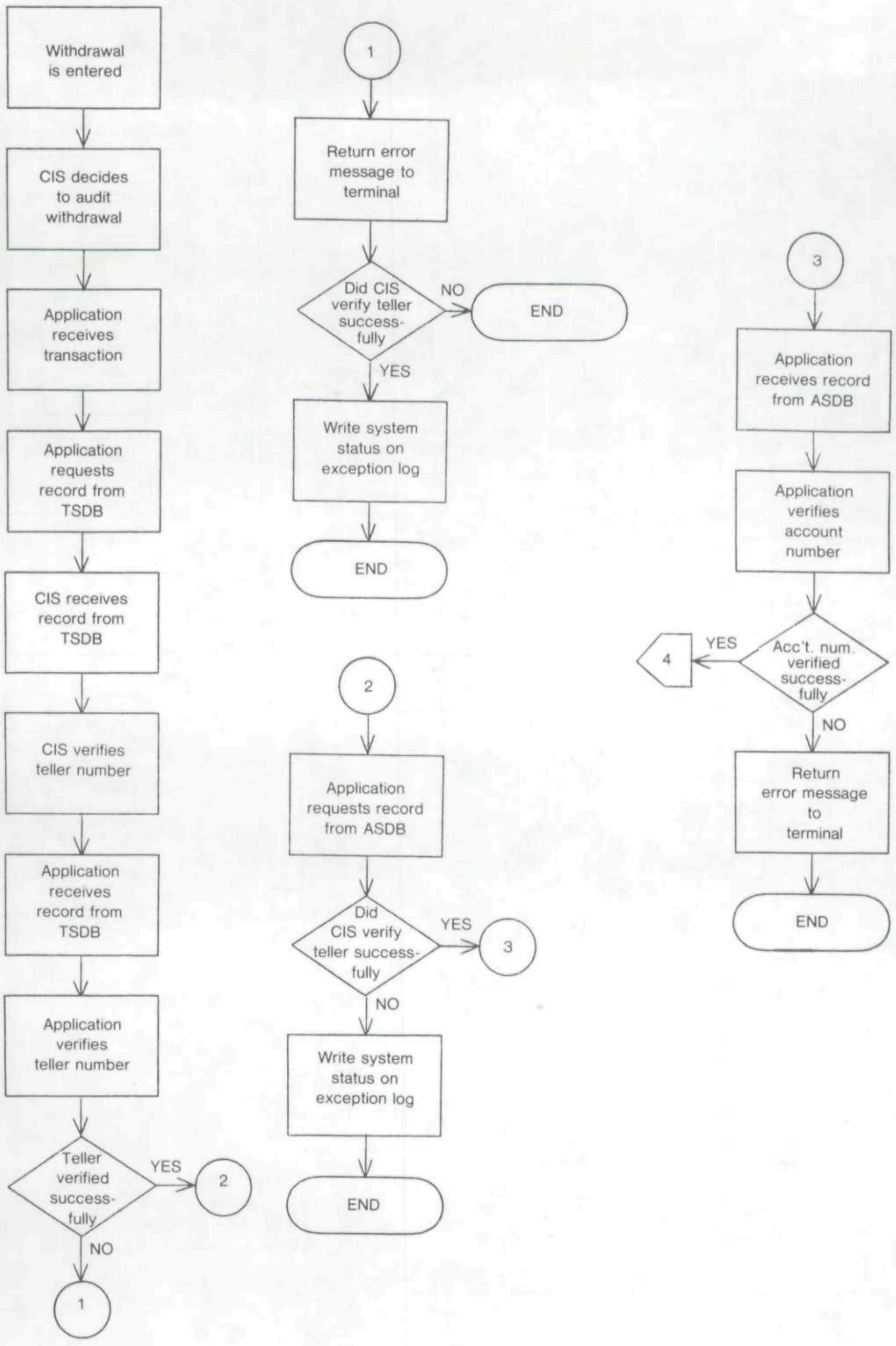
Withdrawal is entered
CIS decides to audit withdrawal
Application receives transaction
Application requests record from TSDB
CIS receives record from TSDB
CIS verifies teller number
Application receives record from TSDB
Application verifies teller number
Teller verified success-fully
YES
NO
1
2
Return error message to terminal
Did CIS verify teller success-fully
NO
YES
END
Write system status on exception log
END
2
Application requests record from ASDB
Did CIS verify teller success-fully
YES
NO
3
Write system status on exception log
END
3
Application receives record from ASDB
Application verifies account number
Acc't. num. verified success-fully
YES
4
NO
Return error message to terminal
END

Figure 4, Part A. Logic Flowchart of Deposit and Withdrawal Application with CIS

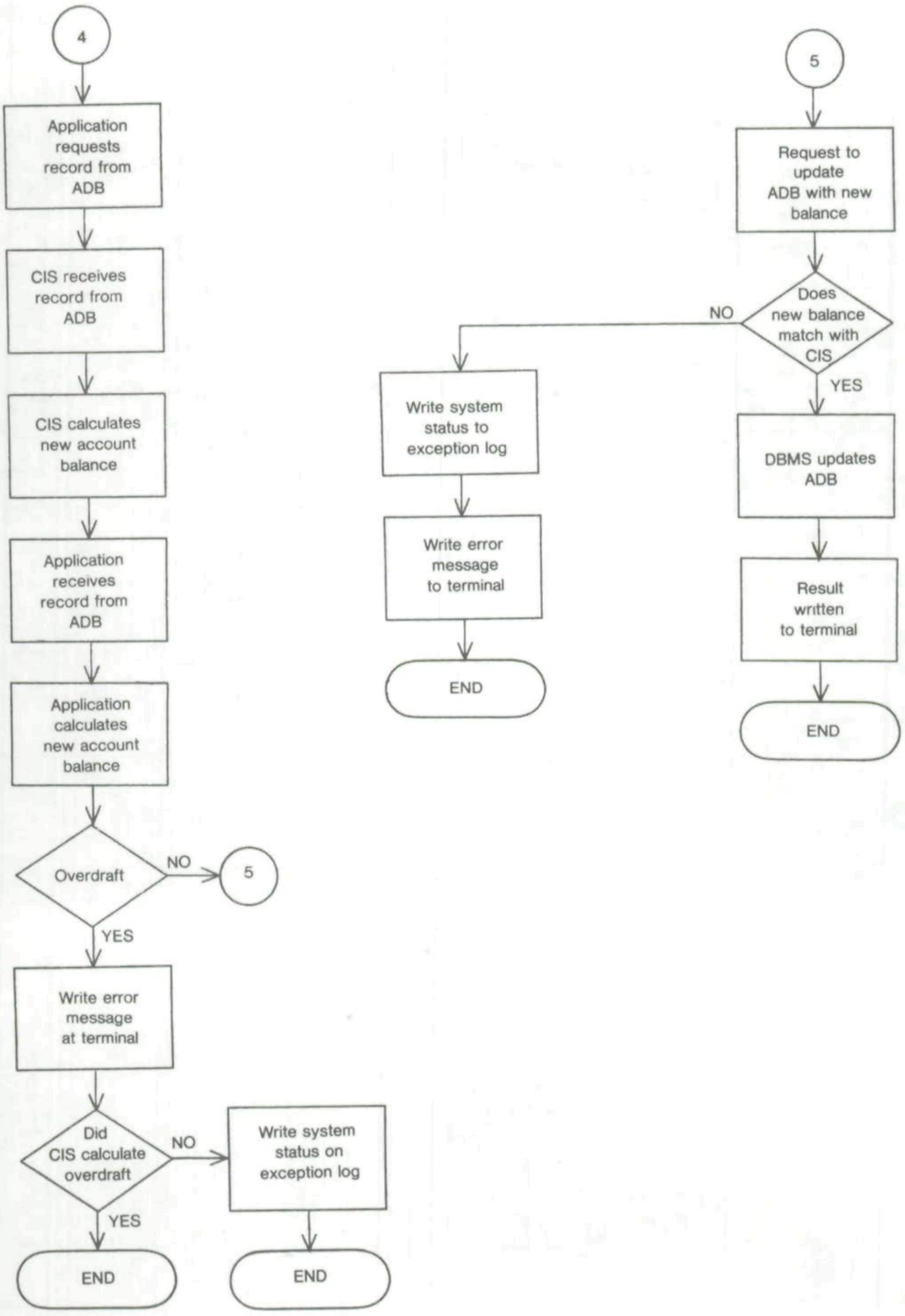
4
Application requests record from ADB
CIS receives record from ADB
CIS calculates new account balance
Application receives record from ADB
Application calculates new account balance
Overdraft
NO
5
YES
Write error message at terminal
Did CIS calculate overdraft
NO
Write system status on exception log
END
YES
END
5
Request to update ADB with new balance
Does new balance match with CIS
NO
Write system status to exception log
Write error message to terminal
END
YES
DBMS updates ADB
Result written to terminal
END

Figure 4, Part B. Logic Flowchart of Deposit and Withdrawal Application with CIS

record, the simulator is notified. The simulator will be able to determine that the attempted update is not warranted because a transaction to initiate this type of update has not been entered to the system through the DBMS. It is not being claimed that all illegal penetrations can be detected. However, it is being claimed that some illegal or unwarranted penetrations via other parts of the system can be detected.

4. Transactions can be audited on an intermittent basis. In the example some of the withdrawals were audited. This is possible because the output results due to a transaction are verified at the time the transaction is being processed by the application. If parallel simulation was used, all deposits and all withdrawals would have to be processed by the simulation. This is because parallel simulation is a substantive audit technique. The output of the application is compared to the output of the parallel simulation after a batch of transactions have been processed. The results will not be the same if the simulation has knowledge of only some of the transactions that the application has processed.

5. There is no output overhead for CIS since no files are created other than the exception log.

A primary concern in evaluating any auditing technique is cost, specifically the cost of implementing the auditing technique, and the overhead incurred in performing an audit. Although a formal or empirical analysis of CIS has not been performed, costs on a relative basis to parallel simulation can be compared.

The cost of implementing an audit simulation for both auditing techniques should be on the same order. The instruction logic of the audit simulation for CIS can be the same as that for parallel simulation. However, the implementation for parallel simulation only needs to work in a batch environment, while that for CIS is for timesharing. This means it must be re-entrant since an application may be operating on several transactions concurrently.

The input/output for these two auditing techniques will be different. CIS needs to communicate with a DBMS, but does no input/output itself other than writing to the exception log. CIS does not request transactions to be read, and performs no input/output with respect to the database. The DBMS gives all necessary input to CIS when the application requests it. For parallel simulation the input/output instructions for reading transactions and database records, and the creation of the simulated output file, must be explicitly supplied in the simulation. This creates overhead solely for the purpose of auditing.

The overhead of parallel simulation to audit a transaction is:

1. execution of simulation code,
2. I/O to produce the simulated output,
3. I/O to read the live output and the simulated output,
4. comparison of live and simulated results, and
5. production of the exception report.

The overhead of CIS to audit a transaction is:

1. execution of simulation code,
2. comparison of live and simulated results,
3. I/O to produce the exception report, and
4. communication between DBMS and CIS.

It is clear that with CIS there is less overhead for I/O. In fact, the only I/O for CIS is sent to the exception report. I/O costs for parallel simulation are expended to 1) produce the simulated output, 2) read the live output and simulated output, and 3) produce the exception report. I/O costs 2 and 3 arise offline from the application. However, the simulated output for parallel simulation is produced at the time that the application is processing live transactions. Since there are usually more simulated output records than exception report records, the amount of I/O overhead that occurs while the transactions are being processed is greater for parallel simulation than CIS.

More CPU overhead at the time that a transaction is being audited is expended for CIS than for parallel simulation. For CIS the extra costs are comparison of live and simulated results—this only occurs for transactions that are being audited, not all transactions—and communication between DBMS and CIS. However, since it is not necessary to simulate every transaction in CIS, total CPU time for simulation for CIS may be less than the total time for simulation of every transaction for parallel simulation. Without analytical or empirical results it is not possible to conclude at this time which method leads to higher CPU overhead.

Future research is necessary to prove the practicality, effectiveness, and cost of using CIS as an auditing technique for various types of applications. CIS interacts very much with the database management software. Thus, an indepth study of the database system requirements is necessary to determine what are the desirable features of current DBMS that support this auditing technique. This should be done from several points of view, *e.g.*, ease of implementing the simulation, or necessary security features within the DBMS in order to support a successful audit. Finally, further analysis of CIS is needed to determine what types of applications CIS is most suitable for in terms of implementation costs, operational costs, and limitations of this auditing technique.

Summary and Conclusions

An auditing technique that audits transactions as they are being processed has been introduced. Concurrent and intermittent simulation is an auditing technique that is based upon parallel simulation. It is similar to parallel simulation in terms of the amount of work and type of code that must be completed by an internal auditor. However, in terms of operation there are substantial differences. Parallel simulation is appropriate for batch processing and sequential files. CIS has capabilities that make it more appropriate for timesharing systems with online update capabilities.

CIS is an auditing technique that simulates the instruction execution of the application at the time the application is processing a transaction. All data and input to the application is accessible by and shared with the simulation. This means that the simulation is notified about each transaction that is entered to the application and accesses to the database by the DBMS. Before any updates are made to the database, or before any output is returned to the users, the simulation can verify the results by executing the appropriate instructions of the simulation that evaluate the internal controls of the application. If an inconsistency is found, all pertinent information about the system status can be put into the exception log. The simulation can then use the results computed by the application or by the simulation, or can choose not to use any of the results, as if there was no transaction.

Because an audit is performed at the same time as the transaction is being processed, and because an exception can be noticed before the transaction has been completed, CIS has the following advantages over parallel simulation:

1. Auditing in a timesharing environment is possible.
2. Applications that update the database can be audited.
3. Not all transactions within a time frame need to be audited. Transactions to be audited can be picked intermittently on any basis — randomness, type of transaction, or transaction source, *etc.*
4. If a discrepancy exists, the result of a transaction can be immediately nullified.
5. The only input/output overhead in the simulation is to produce the exception records.

CIS has great potential since it is the first auditing technique that moves computer auditing from an offline, *ex post facto* process to one that is online and timely. It is not viewed as an auditing technique to be categorized under "auditing through the computer," but as the foundation of a new class of online auditing techniques. This will help bring computer auditing one generation closer to the computer systems that are being used today.

References

[1] Burch, J. and Sardinas, J. *Computer Control and Audit: A Total Systems Approach,* John Wiley & Sons, Inc., New York, New York, 1978.

[2] Cash, J., Bailey, A. and Whinston, A. "A Survey of Techniques for Auditing EDP-Based Accounting Information Systems," *The Accounting Review,* Volume LII, Number 4, October 1977, pp. 813-32.

[3] Fairley, R. "Tutorial: Static Analysis and Dynamic Testing of Computer Software," *Computer,* Volume 11, Number 4, April 1978, pp. 14-23.

[4] Mair, W., Wood, D. and Davis K. *Computer Control and Audit,* Institute of Internal Auditors, Orlando, Florida, 1978.

About the Author

Harvey S. Koch *is an Assistant Professor of Computers and Information Systems in the Graduate School of Management at the University of Rochester. His current research interests are in the areas of modelling and evaluation of methods for computer security and computer auditing, and software reliability. A member of ACM, IEEE, and Sigma Xi, Koch received a Ph.D. in Computer Science in 1973 from the Pennsylvania State University.*