New techniques in computer program verification

William C. Mair

New Techniques in Computer

Anyone who has watched moonshots or other space-exploration programs on television has undoubtedly heard of Murphy's Law—at least as applied to the complex systems associated with space missions. In briefest terms, Murphy's Law holds that a system which can fail, will.

Probabilities of failure depend on degrees of complexity of the systems involved. This applies just as logically to a business data-processing system as to the systems that support a space mission. The big difference, of course, is that the stakes are different. If an astronaut encounters an unexpected situation after his vehicle has left the earth, the consequences—and the dangers—are immediately apparent. However, in a business data-processing system, major problems can exist which are not apparent for a long time. This is where the auditor comes in. The auditor must satisfy himself that accounting principles are properly and uniformly applied. This holds true whether accounting records are kept on computers or with pencil and paper. The auditor's responsibility applies in either case. Accuracy cannot be assumed.

In a business data-processing system, many management people and auditors alike are lulled into complaisance by the automatic checking and verification features built into computer hardware. Computers are quite mechanically and functionally reliable. Therefore, there is a tendency to assume that data produced by computers are also automatically reliable. This is simply not true.

A modern business data-processing system is a combination of elements, including equipment, administrative procedures, and processing programs. As has already been mentioned, features are incorporated into most computers that provide a high degree of equipment reliability without further examination by an auditor. Administrative procedures associated with computer systems can normally be examined through the application of traditional audit techniques. The greatest challenge to the auditor requiring new techniques lies in the verification of the computer programs that process the financial data.

The programs used within business data-processing systems are often referred to as "software." This term distinguishes the functional instructions for the executing of computer operations, provided by programs, from the equipment portion of a computer installation, referred to as "hardware." In general terms, there are two

Program Verification

by William C. Mair

types of software: The first type consists of programs provided by the manufacturer of the computer equipment or by specialized software suppliers. The second type consists of application programs developed or acquired by the user for his own business applications. Programs directly associated with the functioning of computer equipment, the first of the categories identified above, are considered beyond the scope of the audit examination. The manufacturer's software—consisting of microprogramming routines, language processors, utility routines, and operating systems—are far too complex technically to fall within the capabilities of audit personnel.

However, the reliability and performance of systems and programs are very much within the responsibility of the auditor. This, specifically, is the area where Murphy's Law applies to the conduct of the audit. User programs are the error-prone area of computer systems. User programs are the area where anything which can happen will. User programs, in the final analysis, fall directly under the auditor's responsibility as stated in the third standard of field work of the AICPA. The auditor is clearly responsible for eliminating reasonable doubt that material financial errors or improprieties can be generated by or derived from "bugs" in application programs used by the organization.

TECHNIQUES FOR PROGRAM VERIFICATION

In meeting his responsibility to verify the accuracy of data processed by computers, the auditor has a variety of tools and techniques available. In general, these fall into five categories:

1. Auditing around the computer.
2. Program code checking.
3. Flowchart verification.
4. Test decks.
5. Parallel simulation.

Verification by Auditing Around the Computer. One method that has been used extensively in the past is to treat the computer as a "black box" and audit around it. Results of computer processing may be manually verified against source data entered into the computer.

This type of verification can be done either on a sampling basis or through a comparison of balances. External verification—either through sampling or comparison of balances—is frequently effective. Further, where such external verification can be used, it is usually efficient.

However, this approach may be impractical, or simply not available to the auditor because the audit trail is lost in the course of computer processing. In increasing numbers of cases, business data-processing systems are so complex that the original identity of data is lost for manual verification purposes.

In other situations, systems are so gigantic that normal verification approaches using sampling techniques are simply not effective. A case in point occurred with the discovery of a situation which has become a classic example of computer-centered fraud. A programmer for a large savings institution designed an interest-computation application program under which interest applied to depositor accounts was rounded to the lower penny and all fractions were deposited to his account. Routine sampling did not identify any exceptions because the auditors were simply not looking for fractions of pennies and never happened to test the single account where the fractional cents were being deposited.

The point is that, where very large or complex computer systems are involved, conventional sampling techniques will often fail to detect either fraud or unusual-exception situations.

Program Code Checking. Another examination technique for verifying the reliability of client computer applications is to perform detailed analyses of program code listings. Under this approach, a member of the audit team reads and analyzes the detail application coding written by the programmers. In the course of this examination, the audit staff member must identify and analyze any potential errors which can be generated by the program. Obviously, this technique requires the services of a person trained in the principles of auditing and also extremely skilled in programming, with detailed knowledge of the specific programming language and hardware being used. Furthermore, the logic of most computer programs is very difficult to understand in the form of an instruction listing, and a reviewer is quite apt to overlook obscure situations unless he knows exactly what to look for. Therefore, this approach is appropriate only in circumstances where a qualified person is available, and where the auditor has a definite idea of the types of situations or conditions he is looking for. Because of these limitations, program code checking is of little value as an examination technique.

Program Flowchart Verification. Program verification can also be done through examination of logic processing flowcharts. This approach seeks to verify reliability of computer processing by reviewing the logic rather than step-by-step coding lists. In effect, a processing flowchart provides a graphic view of the processing that takes place, instead of a listing of the source program language. An advantage of verification through flowcharts is that it is easier to check the logic of the program than it is with a straight program language listing.

Most computers now accept software routines which will generate processing flowcharts mechanically. This approach also assures the auditor that the flowcharts he examines will be current—reflecting processing as it is currently being done on the computer This capability can be important because manually drawn flowcharts are seldom up to date in working computer installations.

As with the case of coding verification, a review of flowcharts still requires a person expert in both auditing and data processing. This technique, too, can be applied effectively only if the auditor knows what problems to look for.

Test Decks. The term "test decks" refers back to the early days in business data processing, when it was common to enter all system test data into decks of punched cards which were entered into the computer to "exercise" the system. Today, test decks of data can be prepared on magnetic tape or discs, or generated by the computer itself through the use of software. The idea, however, has remained consistent. The ideal test data should present the program under examination with every possible combination of transactions, master-record situations, values, or processing logic which could be encountered in business data-processing operations, and thereby produce output to verify that the programs are functioning properly.

For many years, test decks have been widely used in program verifications for audit engagements. One of the advantages of the test deck approach is that such data can usually be prepared by persons with less technical background than those needed for program code checking or flowchart analysis. However, a person preparing a test deck must still be highly familiar with the logic of the system under examination and with the specific controls within the programs.

The major problem encountered with the use of test decks lies in determining the variety of situations and conditions to be actually included in the test data. It is practically impossible for a test deck designer to anticipate all circumstances which can develop in the processing of a computer application. This is true even when test-generator software is applied, though this special-purpose software represents an improvement over manual design of test decks.

Another drawback to the test-deck approach lies in the fact that it is rarely used to test a complete business data-processing system. Generally, a test deck is applied to individual programs or small related groups of programs. However, in modern business-data processing applications, a single system can frequently involve 100 or more separate computer programs or modules. Although no theoretical limitation exists, test decks, in practice, are seldom used to test systems of this magnitude on an integrated basis. Therefore, it is possible for test-deck verification to be either incomplete or inconclusive—even though detailed testing is done on a major segment of a system. Further, it is frequently necessary to create very extensive master files for the test transactions to be processed against, adding expense to this audit approach.

The biggest single shortcoming of the test-deck approach is that it is limited to the testing of preconceived situations. The design of a test deck usually follows the design logic of the program being tested. Therefore, it is likely that the same "bugs" or loopholes will exist in the testing procedures that exist in the programs.

For example, a test deck was designed to verify the exception-reporting provisions of an installment loan application at a commercial bank. The test deck verified that all edit features of the computer program were functioning as specified. However, a separate analysis uncovered the existence of a number of negative balances for accounts in the installment loan file—one in the amount of $30,000. In this case, since negative balances are improbable for installment loans, no tests had been built into the program to report such situations. The test-deck approach lacked the broad perspective necessary for an effective audit examination.

Parallel Simulation. This approach calls for the preparation of separate programs, independent of those used for day-to-day application processing, which accept the same input as the application programs, use the same files, and attempt to produce the same results. These results are then matched with the results from the "live" program verification through comparison. Although parallel simulation can be done with any programming language, the auditor is best served by general-purpose audit software which makes it possible to create the parallel programs with minimum effort by nontechnical people.

The situations and techniques to be cited in this article have actually been performed in real audit situations utilizing a general-purpose audit software system known as "STRATA" (System by Touche Ross for Audit Technical Assistance). Under this approach, a staff auditor with only minimum knowledge of electronic data processing can describe the records to be processed and the functions to be performed in general terms through the use of structured specification sheets. The computer, with the STRATA software directing it, then calls on functional routines which write their own application programs as the auditor's instructions are interpreted.*

One approach for using STRATA is referred to as "parallel simulation" because the auditor can create a new system of programs which process data *in parallel* with the regular system. The *simulation* designation applies because the program created through the use of the general-purpose audit software performs the same processing functions as the regular-user programs but through a different means. The computer processing is not always as efficient using general-purpose software as is necessary for regularly used applications; however, it is much more efficient to prepare. After the same files and transactions have been processed by both systems, the results should be identical and directly comparable with respect to the financially material areas selected for parallel simulation. That is, the parallel-simulation technique need not seek to reproduce the systems in full detail. The auditor may select application areas on the basis of materiality and processes data independently to validate the results of those specific functions of the client systems.

The important characteristic of parallel simulation as an audit tool is that independent processing of relevant data takes place. This processing need be done only to a level which is sufficient to validate the financial results of the system. The basic concept is the same as with auditing around the computer. The end product is a comparison of results. Where the scale and scope of a system are beyond the capabilities of manually recomputing the results, general-purpose audit software can mechanize the process.

This approach serves to test for errors or exceptions in the critical area of application programs. By paralleling the programs, audit simulation performs an independent verification of results by reproducing the process under which the results are obtained.

* For further description of STRATA, see *Tempo*, Winter 1970; and *The Journal of Accountancy*, July 1971.

The remainder of this article will deal with the concepts and applications of independent audit software as applied through the use of STRATA.

THE ROLE OF PARALLEL SIMULATION

Within the context of an independent audit engagement, parallel simulation can be used for either complete balance verification or for the limited testing of the programs. The use to which parallel simulation is applied depends largely on the nature and scope of company operations. For example, in auditing payroll for a large company, parallel simulation would be used to test the reliability of processing and internal control by recalculating the payroll for selected pay periods. However, in auditing depreciation of capital equipment, all calculations involved in depreciation for the year could be performed to affect a complete audit of this account on an annual basis.

The basic determination of whether parallel simulation is applicable occurs when a computer system is created to generate significant accounting information regarding the firm's revenues or expenses, or to maintain records covering a significant portion of its assets or liabilities. If the auditor relies on the results of the computer processing, either due to necessity or convenience, he must acquire some evidence that his reliance is justified.

As a further condition, the complexity or scope of the computer application should be beyond reach of conventional external balancing techniques. For example, if the organization is using straight-line depreciation, it would be relatively simple for the auditor to verify balances using a desk calculator. However, if depreciation is being calculated on a more complex basis, such as sum-of-the-year's-digits or double-declining-balances, annual balance verification through manual techniques may be impractical. The auditor, then, is faced with a choice between sampling or computer recalculation. Recalculation on a computer is far more reliable.

The choice between using parallel simulation for balance verification or for evaluation of internal control depends also on the individual situation. For example, the computer processing demands of a depreciation account would be small enough to warrant a year-end balance approach. However, it would generally be impractical to rerun all of a company's payrolls for the entire year. So, in the case of payroll, it is necessary to establish the reliability of the systems of internal control. Under parallel simulation, this is done by processing batches of data on an interim basis. Where computerized systems are involved, internal control can be consistently reliable because established computer programs can be depended upon to perform the same functions the same way each time they are used under the same circumstances. Each time they are modified, however, their reliability must be redetermined.

Therefore, in audit engagements involving extensive computerized accounting operations, parallel simulation can serve as a broad, general purpose audit tool which fits conveniently into the working schedule of both the auditor and the audited organization.

THE SYSTEM CONCEPT OF PARALLEL SIMULATION

The functional relationships between computer applications and parallel simulation are represented in the flow diagram in Figure 1. This flowchart dramatizes the direct parallel nature of simulation through the use of general-audit software. Like the "live" application, the simulation software uses the actual computer master file and actual transactions input to the system. There is

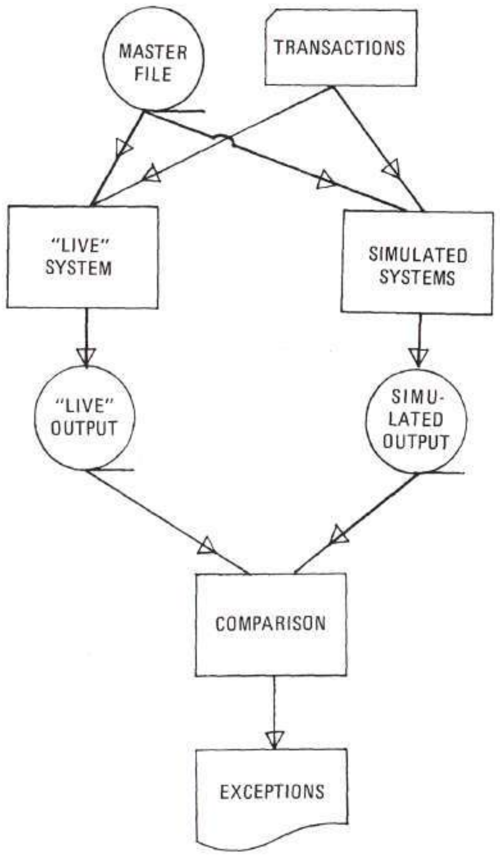


Fig. 1

an additional need, of course, that the auditor determine that the transactions processed under simulation are representative of the transactions which will be encountered by the system for the period under audit. The selection and screening of transactions will be dealt with later in this article.

Under the technique outlined in Figure 1, the STRATA simulation processes transactions against file data, creating its own output files and comparing these with files generated by the "live" programs. The STRATA application can include machine comparison between data produced by the "live" system and that produced by the STRATA application. In such a case, the report delivered to the auditor includes only items representing exceptions.

From an auditing standpoint, the obvious benefit of this approach is that it is more complete and more thorough. The auditor is not restricted to minimum sample transactions as is necessary when manual methods are employed. Rather, the computer can be used to examine and test extensive files of data. Then, because their results can be confirmed, the application programs under which the company processes these transactions are validated.

In terms of audit costs, parallel simulation through the use of general-purpose audit software can usually be accomplished for less expense than other applicable audit techniques, particularly in light of the fact that results may be more conclusive. Programming the parallel simulation through the use of a system like STRATA provides application software at a fraction of the cost which would be involved through conventional programming languages. This is because much of the functional "housekeeping" normally associated with the development of an EDP system is prefabricated within the STRATA technique. This difference is important. The auditor using STRATA does not write individual programs. Rather, he prepares instructions to the computer, which build applications from the functional modules within the STRATA system as processing takes place. This ability to operate at a functional, rather than a detail, level makes it possible for a staff auditor to become proficient in the use of EDP audit techniques after training in STRATA for just one week.

This is not to say that STRATA is "idiotproof." The auditor must thoroughly understand the functions required in computing a payroll, calculating depreciation, or whatever else he wants to do. No method exists that can *anticipate* the procedures an auditor may want to employ. General-purpose audit software is not going to replace any auditor, but it can free him from busy work and allow him to be more effective.

Returning to the list of five techniques for company program verification listed early in this article, we find that parallel simulation, through the use of software like STRATA, is generally accomplished in less staff time and at far lower expense than is incurred using test decks, flowchart certification, or program code checking. As pointed out earlier, verification by auditing around the computer is usually the method that is lowest in cost when it can be appropriately applied. However, where mechanized program checking is necessary, experience on hundreds of audit engagements has indicated that parallel simulation using techniques like those discussed here produces the most reliable results at the lowest costs

DESIGNING A PARALLEL-SIMULATION APPLICATION

Preparation of a parallel-simulation application through the use of software like STRATA is a six-step process:

Step 1. *The auditor defines his problem.* This is usually documented in an informal memo incorporated in the audit work papers. The auditor describes, in simple terms, which functions of the company system are essential to the accurate reporting of financial information.

As a rule of thumb, there are two types of function which warrant verification. One is a direct processing function, such as the calculation of payroll withholding rates, depreciation calculations, and so on. The other type is the control function. Examples include reporting of overdrawn checking accounts with a bank, control totals on the values of files, edit reports on unacceptable input records, and so on.

The auditor examines the record layouts for the systems and, usually through conversation, gains a knowledge of the data processed, the controls applied, and the accounting records created by the system. This need not be a detailed examination by the auditor. For example, an auditor can use simulation effectively if he knows no more about the company system than that it processes payrolls, maintains property and depreciation records, accounts for receivables within a retail store, etc. Based on the auditor's background and experience, a basic application description is often enough to tell him what he should expect from a system and to define the problems for purposes of simulation development. Beyond this, the auditor learns enough about the system so that he will be able to evaluate results of simulation and, particularly, the exceptions reported.

In general, the better acquainted an auditor is with the system, the more accurately he will be able to define the calculations and controls that should be created in the parallel simulation. Conversely, the less an auditor knows about the system, the more time he will have to spend checking out reported differences which may not represent actual exceptions at all. Depending on the nature of the application and the complexity of the system, the auditor must strike a balance between the time spent in studying the system prior to the simulation design and the time which will be necessary for examining and validating results delivered.

This step is usually the most time-consuming phase of the entire examination, at least in the first year it is attempted. Many alternatives are available to the auditor and careful selection of the most effective approach is usually well worth the time involved.

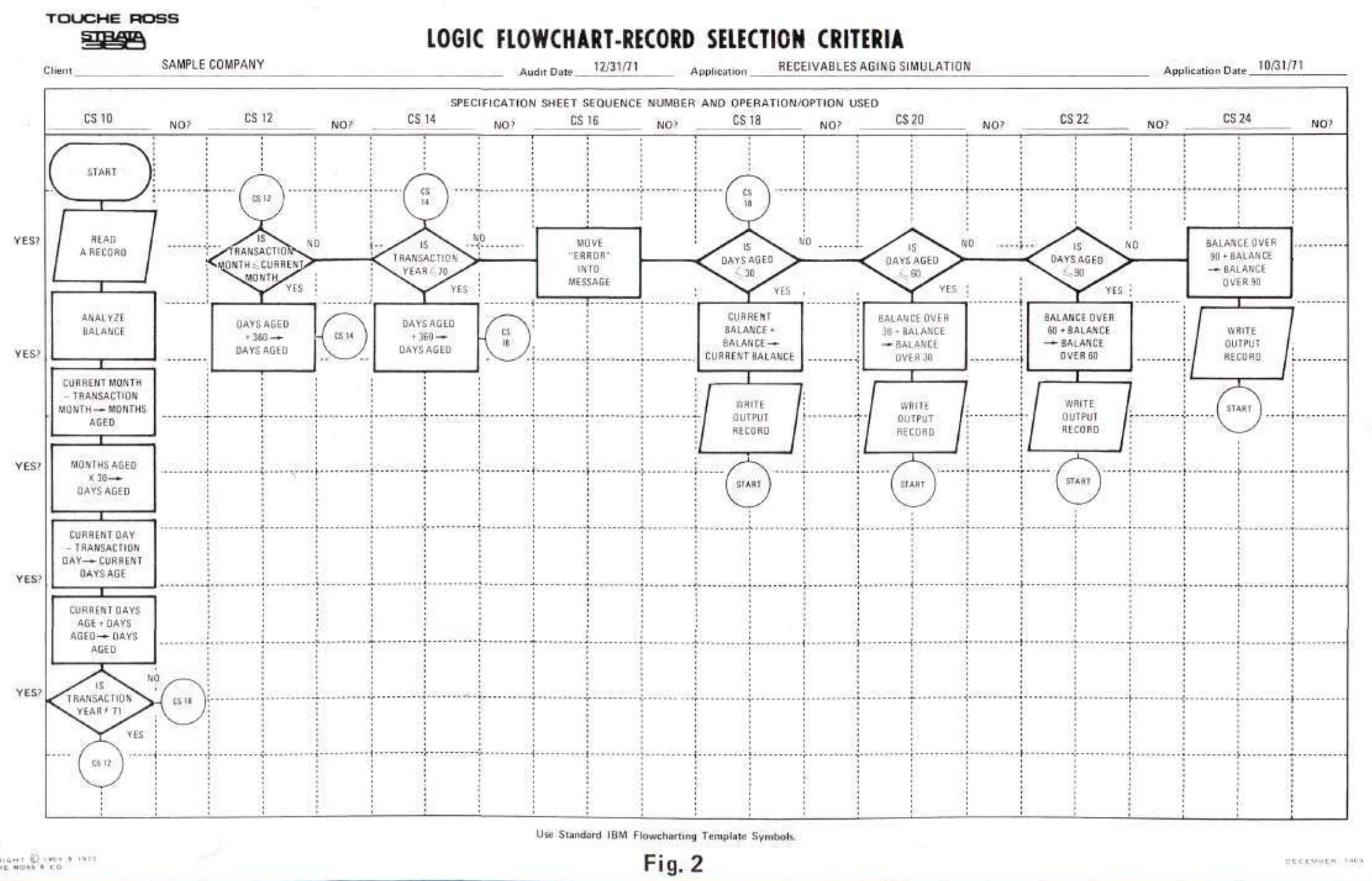
Step 2. *The auditor specifies the logic to be followed in the parallel simulation application.* This is normally done with flowcharts which sequence the functional operations to be performed within the simulation application. Under a system like STRATA, flowcharting is handled quickly. Most highly complex applications can be flowcharted in a maximum of two hours. Flowcharts for simulation applications of less complex systems might be completed in as little as fifteen minutes. This process is illustrated in Figure 2, which contains a logic flowchart for a relatively simple parallel simulation program to be executed under STRATA.

Step 3. *Instructions are coded using STRATA specification sheets.* Specification sheets are unique to the functions performed within the STRATA software. This minimizes the writing necessary by the auditor. The auditor simply enters abbreviated descriptions of the files to be processed and the functions to be performed.

For the purposes of illustration, Figure 3 contains a specification sheet for the data field-selection function of STRATA and Figure 4 contains a specification sheet for the "calculate-stratify" function.

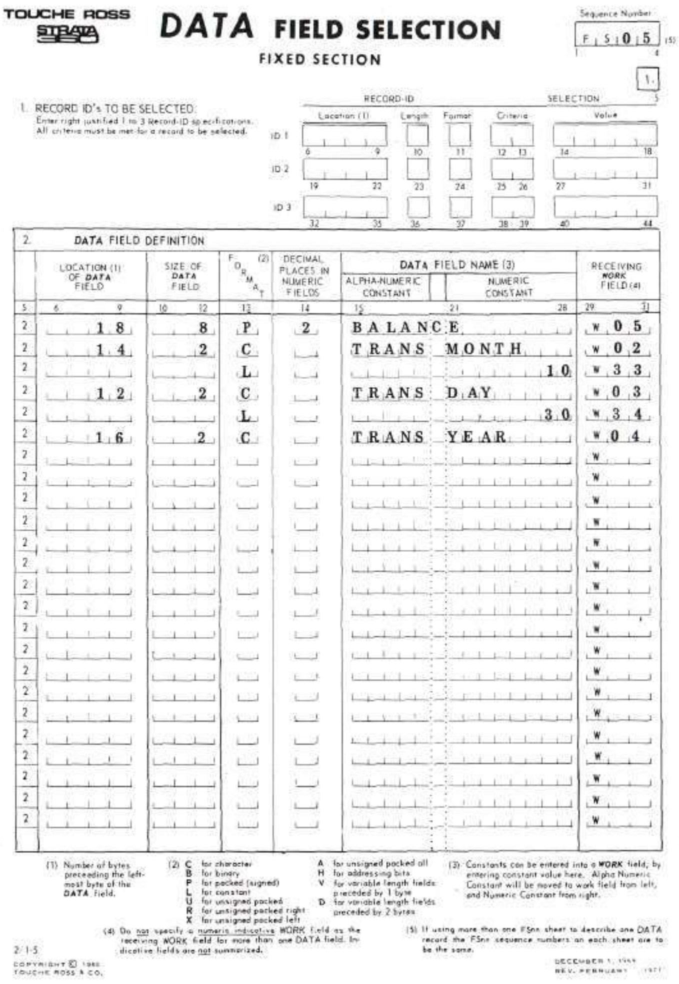
Detailed description of the execution of these forms is beyond the scope of this article. However, the significance of this coding technique can be summarized by indicating that experience has proved that parallel simulation programs can usually be coded under STRATA in less than 10 percent of the time required to prepare a comparable COBOL program for a parallel simulation application.

Step 4. *The parallel simulation application is "debugged."* "Debugging" is an EDP term which recognizes that most computer applications or programs have some flaws in coding or logic when they are originally written. These may arise during the transcription of the specifications to machine-readable punch cards, or



TOUCHE ROSS
STRATA
LOGIC FLOWCHART-RECORD SELECTION CRITERIA
Client SAMPLE COMPANY
Audit Date 12/31/71
Application RECEIVABLES AGING SIMULATION
Application Date 10/31/71
SPECIFICATION SHEET SEQUENCE NUMBER AND OPERATION/OPTION USED
CS 10
NO?
CS 12
CS 14
CS 16
CS 18
CS 20
CS 22
CS 24
YES?
START
READ A RECORD
ANALYZE BALANCE
CURRENT MONTH – TRANSACTION MONTH → MONTHS AGED
MONTHS AGED X 30 → DAYS AGED
CURRENT DAY – TRANSACTION DAY → CURRENT DAYS AGE
CURRENT DAYS AGE + DAYS AGED → DAYS AGED
IS TRANSACTION YEAR ≠ 71
NO
YES
CS 18
CS 12
IS TRANSACTION MONTH ≤ CURRENT MONTH
DAYS AGED + 360 → DAYS AGED
CS 14
IS TRANSACTION YEAR ≤ 70
MOVE "ERROR" INTO MESSAGE
IS DAYS AGED ≤ 30
CURRENT BALANCE + BALANCE → CURRENT BALANCE
WRITE OUTPUT RECORD
IS DAYS AGED ≤ 60
BALANCE OVER 30 + BALANCE → BALANCE OVER 30
IS DAYS AGED ≤ 90
BALANCE OVER 60 + BALANCE → BALANCE OVER 60
BALANCE OVER 90 + BALANCE → BALANCE OVER 90
Use Standard IBM Flowcharting Template Symbols.
9/9-13
COPYRIGHT © 1969 & 1970 TOUCHE ROSS & CO.
DECEMBER, 1969

Fig. 2


TOUCHE ROSS STRATA 360

DATA FIELD SELECTION

FIXED SECTION

Sequence Number F S 0 5 (5) — 1 4

1.

1. RECORD ID's TO BE SELECTED:
Enter right justified 1 to 3 Record-ID specifications. All criteria must be met for a record to be selected.

	RECORD-ID Location (1)	Length	Format	Criteria	SELECTION Value
ID 1	6–9	10	11	12 13	14–18
ID 2	19–22	23	24	25 26	27–31
ID 3	32–35	36	37	38 39	40–44

2. DATA FIELD DEFINITION

	LOCATION (1) OF DATA FIELD	SIZE OF DATA FIELD	FORMAT (2)	DECIMAL PLACES IN NUMERIC FIELDS	DATA FIELD NAME (3) ALPHA-NUMERIC CONSTANT	NUMERIC CONSTANT	RECEIVING WORK FIELD (4)
5	6–9	10–12	13	14	15–21	21–28	29–31
2	18	8	P	2	BALANCE		W05
2	14	2	C		TRANS MONTH		W02
2			L			10	W33
2	12	2	C		TRANS DAY		W03
2			L			30	W34
2	16	2	C		TRANS YEAR		W04
2							W
2							W
2							W
2							W
2							W
2							W
2							W
2							W
2							W
2							W
2							W
2							W
2							W
2							W
2							W
2							W
2							W
2							W

(1) Number of bytes preceeding the left-most byte of the DATA field.

(2) C for character; B for binary; P for packed (signed); L for constant; U for unsigned packed; R for unsigned packed right; X for unsigned packed left; A for unsigned packed all; H for addressing bits; V for variable length fields preceded by 1 byte; D for variable length fields preceded by 2 bytes

(3) Constants can be entered into a WORK field, by entering constant value here. Alpha Numeric Constant will be moved to work field from left, and Numeric Constant from right.

(4) Do not specify a numeric indicative WORK field as the receiving WORK field for more than one DATA field. Indicative fields are not summarized.

(5) If using more than one FSnn sheet to describe one DATA record the FSnn sequence numbers on each sheet are to be the same.

2/1-5

COPYRIGHT © 1969 TOUCHE ROSS & CO.

DECEMBER 1, 1969 REV. FEBRUARY 1971

Fig. 3

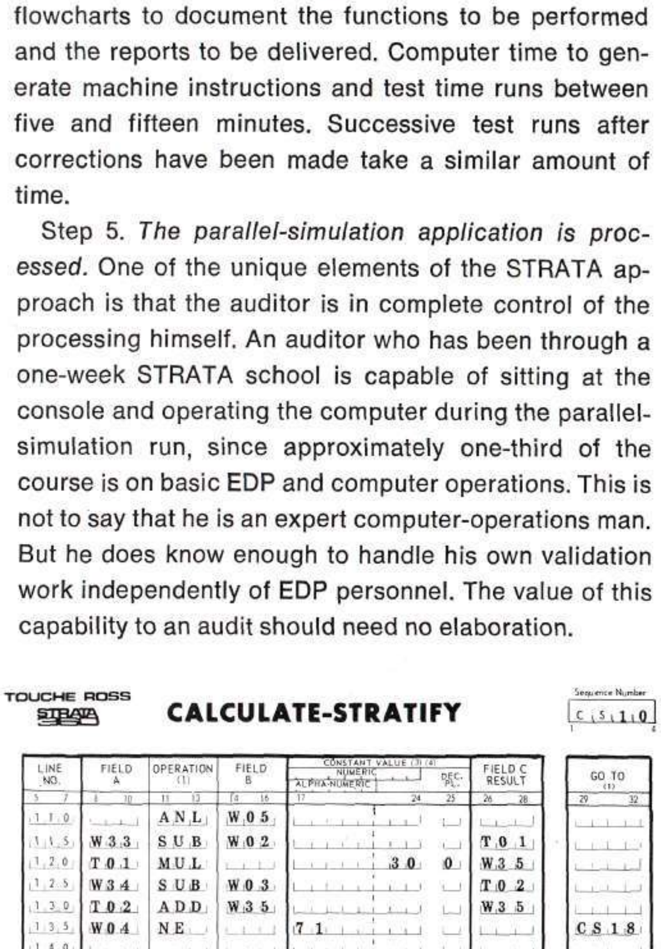
through carelessness or lack of thought during preparation of the logic or specifications. So, all applications should be put through a trial run on the computer to identify such bugs. From this computer test run, the auditor may analyze and correct any mistakes before any large amounts of time are wasted in running full applications.

Debugging is facilitated under the STRATA system through use of a feature which makes it possible to take a segment of a live file and treat it as a complete file for test purposes, without requiring generation of separate test files.

During the debugging run on the computer, STRATA also documents itself—diagnosing and identifying errors or questionable items in the specifications. For each specification sheet completed by the auditor, the computer prints out an easily readable narrative description of the processing performed and the files involved. Where errors in the data descriptions are identified during the test run, messages are also generated by the computer.

When all careless mistakes have been eliminated, the STRATA system designs its application, establishing processing sequences and printing out complete system flowcharts to document the functions to be performed and the reports to be delivered. Computer time to generate machine instructions and test time runs between five and fifteen minutes. Successive test runs after corrections have been made take a similar amount of time.

Step 5. *The parallel-simulation application is processed.* One of the unique elements of the STRATA approach is that the auditor is in complete control of the processing himself. An auditor who has been through a one-week STRATA school is capable of sitting at the console and operating the computer during the parallel-simulation run, since approximately one-third of the course is on basic EDP and computer operations. This is not to say that he is an expert computer-operations man. But he does know enough to handle his own validation work independently of EDP personnel. The value of this capability to an audit should need no elaboration.


TOUCHE ROSS STRATA 360

CALCULATE-STRATIFY

Sequence Number C S 1 0 — 1 4

LINE NO.	FIELD A	OPERATION (1)	FIELD B	CONSTANT VALUE (3) (4) ALPHA-NUMERIC / NUMERIC	DEC. PL.	FIELD C RESULT	GO TO (1)
5–7	8–10	11–13	14–16	17–24	25	26–28	29–32
110		ANL	W05				
115	W33	SUB	W02			T01	
120	T01	MUL		30	0	W35	
125	W34	SUB	W03			T02	
130	T02	ADD	W35			W35	
135	W04	NE		71			CS18
140							
145							
150							
155							
160							
165							
170							
175							
180							
185							
190							
195							

CONDITIONAL OPERATIONS (1)

L - A less than B
LE - A less than or Equal to B
E - A equal to B
NE - A not equal to B
G - A greater than B
GE - A greater than or equal to B
RS Random Selection of nn%. nn is a numeric constant from 01 to 99. For nn% of the records chosen at random, the condition is met.(3)
SC Tests STRATA Code field of the WORK record. If equal to the two-digit Alpha-Numeric constant, the condition is met.(4)

MATHEMATICAL OPERATIONS (2)

ADD A + B → C
SUB A − B → C
MUL A × B → C
DIV A ÷ B → C

OTHER OPERATIONS (2)

ANL Analyze Field B (5)
MOV B → A. Moves Alpha-Numeric field B or Constant to Field A.
COD Enters any two digit Alpha-Numeric Constant in the STRATA code field of the WORK record.(4)

END OF PASS OPERATION

EOP Enter EOP to stop all processing in that STRATA Pass.

GO TO OPTIONS (1)

END pass current WORK record to next STRATA function. (After Calculate/Stratify.)
READ reject current WORK record from any further STRATA function by getting the next record.
CSnn branch to page number nn.
EX CSnn Perform steps on page nn and return. Put EX in "Operation Field" and CSnn in "Go To" field.
blank if any conditional operation fails, a blank "Go To" will cause a branch to the next page.

(1) In Conditional Operations
- if the tested condition is met the next sequential operation is performed.
- if the tested condition fails, STRATA takes "Go To" option.

(2) After performing a mathematical or other operation (except for EOP) STRATA performs the next sequential operation unless the "Go To" field is non blank.

(3) Enter Numeric Constants and Random Selection percent right justified in Constant Value field.

(4) Enter Alpha-Numeric Constants and two digit STRATA Codes left justified in Constant Value field.

(5) Optional – To obtain "Variance" of Field B with Analyze results, enter ANL in "Operation" column and "V" in column 17. Available in 65K STRATA only.

2 1-8

COPYRIGHT © 1969 TOUCHE ROSS & CO.

DECEMBER 1, 1969 REV. FEBRUARY 1971

Fig. 4

The time required for actual processing is directly dependent on the quantity of data to be examined, the size and speed of the equipment being used, and the number of functions being performed. In cases where comparison has been made to fairly complex COBOL programs performing the same operations, STRATA has operated at speeds comparable to the COBOL programs. Typical applications may require anywhere from one-half hour to several hours.

As indicated previously in Figure 1, the typical STRATA parallel-simulation run delivers a computer printout of exceptions identified according to the auditor's specifications.

Step 6. *The auditor resolves exceptions reported during processing.* The reports delivered following the running of the STRATA application should contain all data necessary for the auditor to evaluate and resolve apparent exceptions. For example, the program may have calculated depreciation on an expense basis while the auditor's simulation may compute the total allowance balance. In such instances, there may be round-off differences which are not significant—and which indicate that there are no problems in the program. The simulation processing may also report items that are not true exceptions, but rather are reflections of specific types of special handling situations which are processed properly in accordance with the overall application, but which were not considered by the auditor when the simulation program was designed.

An example of this type of situation occurred when employees requested a company to withhold pay in amount in excess of legal minimums. The auditor's program tested for the normal deduction percentages without being aware of the exception cases. Such exceptions must be resolved, but clearly do not affect internal control reliability.

On another occasion, a STRATA simulation of a manufacturer's payroll program revealed that paychecks had actually been prepared for a number of employees whose identification codes indicated they had been laid-off. Resolution of this exception showed that the program, in fact, did not have a test of employee status before paychecks were generated by the computer.

LIMITATIONS AND PRECAUTIONS

Some of the same general problems and drawbacks described earlier in connection with the use of test decks also must be observed in parallel simulation through the use of general-purpose audit software. Specifically:

1. Special care must be taken to be sure that the data used in the simulation are representative of the total activity in the affected application area for the organization, for the period under examination. If the full year is being reprocessed the problem cannot exist. It can occur, however, when the programs are tested using selected transaction periods.

2. The test data used in parallel simulation must include any unusual types of transactions which may be significant and which may be encountered infrequently in the routine course of the firm's business. This too will not be a problem when an entire year is reprocessed.

3. A large corporation may conceivably have business applications which exceed the capacity of STRATA or other general-purpose audit software. This would be true particularly in multi-application systems using massive table-storage capabilities with a large-scale computer system. For example, a large manufacturing company uses a massive table stored in the main memory of the computer to look up applicable health-insurance deduction rates as part of its payroll system. This table contains hundreds of separate medical coverage plans, each with its own rate breakdowns for family size and other factors. Examination has shown that this table exceeds the capacity of STRATA to duplicate the processing, although it can simulate the processing by using an alernative approach.

SOLVING SPECIAL PROBLEMS

In overcoming the first two of the problems listed above—the need for representative data and for data which include unusual transactions—the auditor may use an approach that combines the test-deck and parallel-simulation techniques. The sample actual company transactions normally processed under parallel simulation may be augmented by additional test-deck data designed to include both representative routine transactions that might not occur in the selected sampling, plus unusual transactions of significance that might not be included. Where such test decks are developed, they may be balanced with routine "live" transactions in order to give the auditor a more realistic basis for appraising client exposure to the possibility of unusual transactions that may not be processed according to specifications.

In dealing with the third situation described above—systems with lookup tables residing in the main memory which are so large that audit software cannot be accommodated—simulation applications can be subdivided or changed so that client data normally housed in main memory can be introduced in more digestible segments through an auditor-created file having the information in the table.

OTHER APPLICATIONS FOR AUDIT SOFTWARE

To keep the topic of parallel simulation in perspective, it should be pointed out that this is just one of several potentially important uses for general-purpose audit software within a public accounting firm. Others include:

1. *Balance examination.* Software applications can be prepared which perform tests of reasonableness or produce listings of selected records for balance verification. A good example of the use of this technique within an audit engagement is in the preparation of confirmations.

2. *File and record adjustments.* STRATA has also been used for both diagnosis and adjustment of computer-maintained files. For example, one computer user had failed to police the correction of errors reported in an edit run of data-processing system. The situation had worsened to a point where file capacity for error listings had been exhausted. A STRATA application was developed to identify offsetting error entries, and entries that could be removed based on other criteria (such as age). The STRATA application then developed the necessary machine-readable input transactions to the computer system to adjust the error file.

3. *Sample selection.* The concept of sampling within an audit engagement changes with the availability of general-purpose audit software. In many applications, for example, it is possible to perform 100 percent examinations of records where this would have been impossible under conventional examination techniques. Where files are so huge or activity rates are so high that full examination is not feasible, general-purpose audit software is used regularly to implement advanced statistical sampling procedures.

4. *Financial modeling.* The auditor can assist his clients with a "what if" approach to their financial applications by simulating an application, but with an alternative method of processing, thus forecasting the implications of the potential decision. For example, different depreciation approaches or financial assumptions can be tested within a computerized information system to evaluate how changing techniques or conditions might affect the company's taxes or financial reporting.

5. *Management services.* Where the consulting arm of a public accounting firm undertakes computer-related engagements, an application system like STRATA can be used as a tool for the economical preparation of one-time programs. For example, one group of consultants was asked by a large retailer to assist with a study of the costs related to the granting of credit. The consultants gathered a large variety of statistics from each of a number of selling locations. Then, using STRATA, the statistics were edited for consistency. Finally, when ample data had been accumulated, the software was used to analyze, distribute, and summarize the data so as to produce meaningful cost information.

In this example, no computer file was involved. Rather, the data were keypunched under the direction of consultants and the software was used to produce a one-time application far more efficiently and economically than could have been done with conventional programming languages.

6. *Management information systems.* Where a company has extensive application files created by computer systems, general-purpose audit software can be used to analyze existing files and to organize files as a basis for developing management reporting systems. In addition, where one-time analyses and reports are needed, general-purpose audit software is frequently the least expensive way to create them.

CONCLUSION

The point of this presentation has been to indicate that general-purpose audit software is an existing, in-place tool ready to assist the auditor in meeting his obligations under the third standard of field work where extensive computerized systems are in use.

Experience has established that testing of computer systems and programs can be done effectively and inexpensively through parallel simulation. Under this approach, live data are processed under applications developed through the use of general-purpose audit software to test, compare, and identify exceptions generated by the company's data-processing applications. General-purpose audit software has proved itself as a more reliable and less expensive method for auditing EDP applications than any other available in situations where systems are too complex for simple verification by auditing around the computer.

New application areas for general-purpose audit software are emerging continually as auditors and companies gain experience with its use. In conclusion, then, general-purpose audit software represents a proven tool for the public accountant, and additional uses are emerging continuously.