Bureau of Business Research
Graduate School of Business Administration
University of Michigan

December, 1970

BUREAU OF BUSINESS RESEARCH WORKING PAPER
NO. 15

PROGRAM AUDITING

by

Andrew M. McCosh
Associate Professor of Accounting
University of Michigan

BACKGROUND OF THIS PAPER

This paper describes the possible auditing technique entitled Program Auditing. The technique involves examination by an auditor of the text of client computer programs. Experimental use of the method is suggested, and circumstances where it would apply are defined.

The purpose of this article is to suggest a method of audit review which may be of value in certain circumstances. The method is called program auditing. It involves the study, by programming personnel on the auditor's staff, of the text of computer programs used in data processing by client companies. A variety of other auditing techniques exist for dealing with computerized systems, and each is valuable in the right circumstances. If the auditor believes, however, that a program deserves unusually careful study for any reason, program auditing may be the best answer.

We shall first consider the relevant standards of auditing, and then look at the objectives of a system review. We shall next consider how the available techniques, including program auditing, meet these objectives. Some suggestions on the methods which might be used in carrying out a program audit are then offered, and the economic feasibility of the technique is reviewed.

Applicable Audit Standards

The second standard of field work requires the auditor to test the internal control system, so that he can decide how much reliance to place on the system and how to limit his other procedures. To make this evaluation, the auditor must acquire "knowledge and understanding of the procedures and methods prescribed and a reasonable degree of assurance that they are in use and operating as planned."[1]

When confronted with a modern computerized accounting system, the auditor must decide how best to acquire the required knowledge and understanding

---

[1] AICPA, Statements on Auditing Procedure #33 "Auditing Standards and Procedures" NY, AICPA, 1963, Page 32.

of the procedures. The procedures, of course, include the organizational and administrative controls[1] set up to assure the independence and coordination of the operating departments. However, a significant proportion of the processing in a modern system occurs within the computer. Many of the accounting records of the concern may be maintained on tape or on disc. To acquire knowledge of the procedures in this part of the system, the auditor must become familiar with the steps the computer has been programmed to take.

The auditor may use a variety of techniques in attaining this familiarity. Most auditors will undoubtedly prefer to use several different methods, depending on the particular circumstances. In fact, more than one technique may be used on a single job. The following list of approaches, therefore, is not intended to be mutually exclusive.

The auditor may study the procedures which are carried out within the computer by requiring a full or partial printout of the files. He may then perform his review in exactly the same manner as if the system were manual.

Alternatively, the auditor may use a general purpose audit program[2] to retrieve information from the files, and to summarize and analyse these data as necessary. Many such programs are now available to the profession; most were created by major public accounting firms with the auditor's needs in mind. These programs do not, by themselves, test the validity of the procedures used in developing the files. However, they do simplify the retrieval of data for manual comparison with original documents.

---

[1] For a useful discussion of such controls, see Moore, M.R., "EDP Audits, A Systems Approach", Arthur Young Journal, Winter 1968 and also John, R.C. and Nissen, T.J., "Evaluating Internal Control in EDP Audits," Journal of Accountancy February 1970.

[2] See Porter, W.T., "Generalized Computer Audit Programs," Journal of Accountancy, January, 1969, for a description of one such program.

A third method the auditor may use is the test deck approach. By creating a set of sample transactions, and an appropriate master file, the auditor may verify the processing steps taken by the client's computer programs by observing whether the program handles his sample data properly.

The fourth method, and the principal subject of this article, is the technique of program auditing. Under this approach, the auditor seeks to familiarize himself with the steps the computer has been programmed to take by the straightforward process of examining the text of the computer program itself.

The Objectives of the System Review

The auditor is interested in answering four questions in inspecting a computerized system, namely

1. Can the system handle valid data correctly?
2. Can the system deal properly with wrong data?
3. Are all the steps the system takes legitimate?
4. Is the system inspected, the one in normal use?

We shall not consider question four further, for reasons of space.

The first question can be answered quite satisfactorily by using any of four methods listed earlier. The choice of method would depend on the relative costs of carrying out the various tests, and on the thoroughness with which the auditor wished to study the system.

All four methods can also deal with question two, but with varying degrees of success. The study of the procedures for handling wrong items must, obviously, include an evaluation of the various types of error which can happen. The auditor would then inspect the system in operation to see if each of the various errors has been handled appropriately. The printout method and the general purpose audit program method are limited in this respect. They operate only on the actual data.

They cannot supply tests or examples of the procedures in use for situations which did not arise during the period. If a particular type of error did not come up, the capacity of the system to handle the situation cannot be discovered by inspection. The test deck and program audit methods are not limited in this way, as the auditor may include all types of error in his test deck, and the program auditor will study all the data control instructions as part of his routine. Of course, if a sufficient period is studied, the user of the first two methods may argue that the absence of certain types of error from his inspection period means they were not very important types in any case. The validity of this argument clearly rests on the possible damage that could be done by a single violation of one of the unexplored error checking procedures, and cannot be discussed in general terms.

This brings us to the third of our questions, concerning the legitimacy of the steps in the system. The auditor is expected to understand the procedures prescribed, but also to assure himself that they are "operating as planned" as statement #33 puts it. He cannot hope to establish that the procedures are <u>never</u> violated; that would get the auditor into the business of fraud detection, a responsibility he has never accepted. He may reasonably be expected, however, to note and comment on <u>systematic</u> violations of planned procedures. If the actual routine procedures are found to include steps which were not envisaged in the plans the auditor ought to observe this discrepancy.

How can the auditor find out about such systematic violations of planned procedure in a computerized system? We shall leave aside, again for space reasons, the parts of the system lying outside the computer. Within the machine, the only ways systematic deviations from planned procedures could arise would be if they were built into the program, or if a substitute program were used, or if a revised

operating system were employed. The last possibility would be uncovered rapidly by a study of the printout on the operators console, and is quite unlikely in any case. Standard procedures are also available to deal with the problem of the substitute program[1]

The remaining problem is that of a deviant procedure built into the program. For instance, the documentation may say that certain tests are imposed on the input data. The program, however, may not impose the test, or may omit the test in certain cases. If the test is missed out entirely, a test deck approach would bring out this omission, but if the program was arranged to omit the test only in certain predetermined circumstances, the test deck would not find the deviation except by good fortune. Certainly, the printout method and the general audit retrieval program methods are not designed to deal with this phenomenon, and would be very unlikely to uncover it.

An example of this type may be instructive. The simplest possible case of an actual system differing from the planned is shown in exhibit 1. The program segment[2] purports to print a list of customers whose balances exceed their predetermined credit limits. In most cases, the list will be prepared as planned. However, in the case of a particular customer number (the number stored in the variable KRUNCH), the program is designed to suppress the printing of the customer's particulars. Of course, the same basic technique could be used in a receivables program to avoid the posting of a sale to a particular customer, or to effect any of a host of other maneuvers.

---

[1]See Porter, W.T., Auditing Electronic Systems, Belmont, California, Wadsworth Publishing Co., 1966, Page 60 et. seg.

[2]The fact that the example is in Fortran is not significant. The same result could be obtained in Cobol, Assembler, PL/1, or any other programming language.

```
      DO 1 J=LO,HI                          Cycle through all customers with numbers from LO to HI

      IF (BALANS(J).LE.CREDIT(J))GO TO 1    If the current balance is less than the customer's
                                            credit limit, skip to the next customer.

      IF (J.EQ.KRUNCH) GO TO 1              The fraudulent instruction.  If the customer number
                                            is equal to the value of the variable KRUNCH, the
                                            test will be bypassed.

      WRITE(6,2)J,BALANS(J),CREDIT(J)       Print out the number, balance, and credit limit of
                                            those customers whose limit has been violated.

1     CONTINUE                              End of the cycle.
```

Exhibit 1 A fortran program segment illustrating a fraudulent limit check bypass

The auditor may object, of course, that the extra commands in the program are fraudulent in nature, so that he is not obliged to locate them. It seems crucially important, to this writer, to distinguish between two activities the auditor might undertake. He is not expected to pin down a fraud. He is expected to establish with "a reasonable degree of assurance" that the procedures are operating as planned. Can the auditor claim such assurance as to procedures if he makes no study of that significant part of the procedure which is contained in the program? At the time of writing, no legal pronouncements have settled this question. The 1963 statement of the Institute states that adequate assurance can be obtained by "testing the accounting records and related data and by relying on such evaluation for the selection and timing of his other auditing procedures".[1] This statement does not, unfortunately, deal directly with the type of built-in, programmed abnormality just illustrated, though it clearly absolves the auditor of the need to look for a unique, "once-only" fraud.

The auditor must always make trade-offs in deciding where to commit his recources. Given the undecided status of the question posed in the previous paragraph, and given the already extensive demands on the auditor's time, it seems at the present time reasonable for auditors to claim adequacy of assurance without a program audit in the absence of special circumstances. The special circumstances might include suspicion of irregularity from any cause, observation of severe defects in the internal control system, or inability to trace the origin of observed transactions. In the remainder of this article we shall assume that some such occurrence has alerted the auditor's attention.

---

[1]AICPA, ibid, page 11.

Auditing a Program--Some Suggestions on Procedure

If an auditor plans to make use of the program audit method as a part of his procedure for a particular client, he must decide first how to go about it. In particular, he must decide which programs to audit, and what he is going to look for in each of them.

As we have previously indicated, the program audit method is most valuable in testing whether all the steps in a program are legitimate. It is likely to be most useful, therefore, in reviewing the programs which are most likely to contain instructions which are not legitimate. The auditor may, by reviewing the organization of the concern and the skills of the various employees, determine where the need is greatest. Payroll and receivables programs are prime candidates for such attention, and the auditor may add others depending on the characteristics of the client.

The nature of the program will dictate the objectives of the program audit search in each instance. Particular care must be exercised to verify the presence and thoroughness of data controls, such as batch totals, limit checks, and file validity labels.[1] There has been a tendency, in some organizations, to omit such controls, or to include only the barest minimum in the programs. The professional accountant is uniquely equipped to assess the value of the controls in use, and to suggest additional controls if needed. In the course of a program audit, the auditor's programming assistant may be instructed to list the data controls actually observed in the program. The professional accountant may then review the list and decide if it is sufficiently thorough. A more detailed discussion of programmed data controls is presented in the appendix to this article, illustrating two of the commoner types of control.

---

[1]For a review of the nature of such controls, see Peterson N.D., "Error Control on EDP Systems," Management Accounting, November 1970 and Davis, G.B., Auditing and EDP, AICPA, 1968, Chapter 6.

In reviewing the program text, the auditor should also advise his programming assistants to be on the look out for superfluous instructions. Exhibit 2 illustrates how a block of illegitimate instructions may be found in a program.

```
      IF (ACCNT.NE.V1.OR.DATE.NE.V2) GO TO 99
                Unless both ACCNT = V1 and DATE = V2, the program will
                proceed directly to statement 99.

      Insert any set of commands here.
                The commands could be of any type, and need not bear upon
                the transaction being processed.

   99 CØNTINUE
                End of the fraud segment.  The majority of transactions
                would bypass the segment completely.  So will the transactions
                in the test deck, except by chance.
```

Exhibit 2 A Fortran program segment illustrating a double-key transfer.

In the exhibit, a "double-key transfer" in a receivables program is shown. If the variables ACCNT and DATE both take on predetermined values, the illegitimate program segment will be called into action. If either of the two key variables has another value, the segment will be ignored. Clearly, it would be difficult to spot this procedure by means of a test deck. No details of the illegitimate program segment have been shown in the exhibit. The reason for this deliberate omission is that the segment could take on almost any form and could effect any account, not just the one used as key. The program adjuster could, for example, arrange it so that any time customer number 12345 had a transaction on the 14th of the month, a credit would be entered into the account of customer 93762 in the amount of the previous transaction processed or the account balance, whichever is lower. There are many possible ways of dealing with the debit. Notice that the segment does not affect customer 12345 in any way.

Some auditors may elect to perform a program audit by flowchart inspection. A flowchart may be easier to follow than the actual program text. The source code of the client program can be obtained, and processed with a flowchart generating program.[1] The output from this run would be a flowchart of the client program, which could then be studied by the programming assistants to the auditor. It is not desirable to audit the flowcharts developed by the company staff before the program was written; these flowcharts are likely to be out of date.

The Economic Feasibility of Program Auditing

There are two problems facing the auditor who wishes to implement program auditing: first, the availability of suitably trained personnel, and second, the cost of the audit. Each of these difficulties merits careful attention.

To carry out a program audit of a complicated accounting computer program, certain skills are required of the audit personnel. First, they must be able to think logically and must know the accounting procedure which the program purports to perform. In addition, however, the auditor must have a sound working knowledge of the computer programming language in which the program has been written. This skill is less easy to find. While more and more schools of business are including computer programming courses in their cirricula, and more and more students with a major in accounting are taking these courses, there is still a serious shortage of auditors with the requisite facility in a computer language. Those of us on the academic side of the profession must certainly see that our students appreciate the need for this skill and have the opportunity to acquire it at an early stage in their academic career. Those who are already in practice may find it advisable to learn one or more programming languages themselves, by attending one of the many one-or two-week seminars offered by various organizations around the country. Even

---

[1] As an example of this type of program, see Flowgen/F, Anaheim, California, Calcomp Computer Products, Inc., 1968.

this limited level of understanding should permit the trained auditor to direct the attention of programmers on his staff toward the most significant program segments to be audited. It is likely that the best audit results would be achieved by using skilled programmers to perform the detailed instruction-by-instruction review of the program, because such people could perform the task much more efficiently than a novice. However, if the auditor himself is sufficiently skilled at program analysis to be able to tell them what to look for, the programmer-auditors need not be professionally trained in accountancy. There is no question that the staff problem is serious, but with a concerted effort by practitioners as well as academic members of the profession this shortage should be reduced within the decade, at least to the point where it is comparable to shortages in other segments of the profession.

There are three separate aspects to the problem of cost. The first and most obvious has to do with the actual time it takes to perform a reliable program audit. Very little experimental work has been done on this question. This writer tried performing a complete program audit on a student's accounts receivable program written in Fortran. Without hurrying unduly, I found it possible to audit the program at a rate of one instruction per minute. This means that, allowing a maximum of five hours work per day, a 2,000-instruction payroll program could be examined fully in just over six days. The relatively short working day is proposed on my assumption that the quality of work would deteriorate if more hours were attempted.

Naturally, the auditor would not expect to make a complete audit on a given computer program every year. Many standard computer programs are used with only minor modifications for several years. The auditor could therefore concentrate on satisfying himself in the second and later years that the program changes were

sound and were the only changes made. The full investigation would be necessary only once every three or four years.

When a program audit has been performed, the auditor can be quite sure of the validity of the procedures. He may, as a result of this assurance, elect to reduce the amount of detail checking which he causes his staff to undertake. If he does so elect, the cost of performing the program audit will be partially offset.

The total cost of performing an audit of a program is likely to be highest the first time it is tried. The auditor and his programming staff will be learning and experimenting, and the client staff will not be used to the idea. Also, the first program audit must cover the whole program, so the second year economies mentioned above will not be available.

Summary

The technique of program auditing may be a valuable addition to the auditors tool kit in certain circumstances. If the auditor believes the program is likely to merit careful scrutiny, because of the way it is used in the client company or because of its importance to specific employees, program auditing should be considered as a possible technique. Program auditing may be used instead of a test deck approach; the latter method is likely to be less expensive in the first year, but does not normally provide as high a level of assurance as program auditing will.[1] In choosing which method to use in a particular instance, the auditor must consider the longer term cost (second and later audits) as well as the level of assurance he seeks and the initial expense.

It is to be hoped that an auditing firm will undertake an experimental test of the program audit method. Probably one of the larger firms would be best

[1]The problems which can arise with test decks are fully discussed by Horwitz, G.B., "EDP Auditing--The Coming of Age," Journal of Accounting, August, 1970.

equipped to conduct such an experiment, from both staffing and cost viewpoints. It is only by testing out such innovations in the field for a reasonable period of time that progress in audit methodology can be achieved.

Appendix--Programmed Data Controls

The purpose of this appendix is to show what programmed data controls may look like. The literature referred to in the main text of the article has sufficiently explored the rationale for the various controls. It is hoped that this appendix will assist auditors who wish to communicate the nature of such controls to their programming assistants.

Two of the many possible types of checks which should be present in a data processing program are "existence" and "limit" checks. Special computer programming languages have been devised for the express purpose of performing tests of this sort.[1] Exhibits 1 and 2 present simple examples of these data control procedures. Although the program segments in these exhibits have been written in the Fortran language, the same basic principles apply in any other language. We shall briefly examine Exhibits 1 and 2 before turning to the audit problem associated with each.

In exhibit 1 a test is being imposed on the employee number on a time card. Instead of proceeding directly to the calculation of pay, the program segment compares the number of the employee against a prepared file of valid employee numbers. If the number read in from the time card is not found in the file of employee numbers, an error message is printed advising the supervisor that the employee number was not on the list. Further, the processing of the time card is suspended. On the other hand, if the employee number is found to occur more than once on the verification list, another error message is printed, and processing is suspended once again. It is probable that the second type of error exists in the list itself rather than on the data card, but processing is suspended in any case to permit the supervisor to take whatever action seems required. This

[1]For example, the Edit Program Generator language developed for Computer Sciences Corporation.

situation might arise if two employees had been accidentally assigned the same employee number. If the employee number on the time card is found to occur only once on the list of valid numbers, processing of that card is permitted to continue.

Having employed an existence check to verify the reasonableness of the employee number, we now apply a limit check to the hours worked. Exhibit 2 presents a Fortran program segment implementing one possible set of limit checks which might be used in this situation. Exhibit 3 summarizes the limits in tabular form. Naturally, the particular set of limits used would vary from one corporation to another, but the principles employed in these examples are generally valid. The processing of a pay check is permitted only if the number of hours worked according to the time card lies in the range of one to eighty hours in the week. If the hours worked exceed forty-eight, overtime becomes payable. If the number of hours worked exceeds eighty, payment is suspended and appropriate officials are notified by the printing of a suitable list. Another list is prepared of those employee numbers for whom zero hours were recorded.

```
8   READ(5,1)HOURS,NUMBER          Read in time card for next employee

    EXIST=0                        Set verification counter to zero.

    DO 2 J=LO,HI                   Search through all the verification numbers.

2   IF(NUMBER.EQ .CHEK(J))EXIST=EXIST+1
                                   If the read in number is equal to the verification
                                   number currently being considered, increase the counter
                                   by one.

    IF(EXIST-1)3,4,5               Go to instruction 3 if 'exist' is zero, instruction 4 if
                                   "exist" is 1, and instruction 5 if it is greater than 1.

3   WRITE(6,6)NUMBER               Print error message shown in statement 6

    GO TO 8                        Read in another time card

6   FORMAT('NO SUCH PERSON AS', I6)
                                   Error message. The employee number read in on the time
                                   card has been compared with the list of all valid
                                   numbers, and has not been found on that list.

5   WRITE(6,9)EXIST,NUMBER         Print error message shown in statement 9.

    GO TO 8                        Read in another time card.

9   FORMAT('THERE ARE ',I5,4X,I6,'IN THE FILE OF VALID EMPLOYEE NUMBERS')
                                   Error message. The employee number has been checked
                                   against all the numbers in the valid numbers file, and
                                   has occurred more than once. This indicates a possible
                                   error in the composition of the file, rather than in
                                   the time card; however processing of the time card has
                                   been suspended until the manager has a chance to
                                   find out what has happened.

4   CONTINUE                       End of the cycle.
```

Exhibit 1 A Fortran program segment illustrating a simple "existence" check.

	IF(HOURS.LT.169) GO TO 11	Test whether hours worked in the week are possible at all. If so go to instruction 11.
	WRITE(6,12)NUMBER,HOURS	Write error message as in instruction 12.
2	FORMAT('HOURS OVER 168',I6,4X.I6)	Notify supervisor of payroll that an impossibl number of hours were recorded for this man.
	GO TO 8	Read a new time card. Suspends payment of the employee with the excessive hours.
1	IF(HOURS.LT.80) GO TO 13	Test whether hours are within the normal maximum limit. If not, suspension of payment is required.
	WRITE(8,14)NUMBER,HOURS	Write error message as in instruction 14.
4	FORMAT('EXCESS HOURS',I6,4x,I6)	By printing the excess hours in channel 8, the list of such employees is separated from the rest, for the information of a separate officer.
	GO TO 8	Read a new time card. Suspends payment.
3	IF(HOURS.LE.48) GO TO 15	If hours do not include any overtime, proceed to the next calculation section.
	PAY=HOURS*RATE+(HOURS-48)*OTRATE	Compute gross pay as total hours at normal rate plus overtime hours at the overtime premium rate.
	WRITE(7,16)NUMBER,HOURS,PAY	Print details of employees receiving overtime on channel 7, for the advice of the relevant manager
6	FORMAT('OVERTIME', I6,4X,I6,4X,I6)	Overtime listing format.
	GO TO 17	Proceed to the check-writing routine.
5	IF(HOURS.EQ.0)GO TO 18	If the recorded hours are zero, an error may have occurred. Steps to correct this must be begun.
	PAY=HOURS*RATE	Compute normal pay for those employees working up to 48 hours (but more than zero hours).
	GO TO 17	Proceed to check-writing routine.
8	WRITE(3,19)NUMBER	Write out the numbers of those employees for whom zero hours were recorded on channel 3, fo the information of the relevant official.
9	FORMAT('ZERO HOURS',I6)	Format for the listing of employees with no time recorded.
	GO TO 8	Read a new time card.
7	CONTINUE	Start of the main computation.

xhibit 2 A Fortran program segment illustrating a simple set of limit checks.

HOURS WORKED		ACTION TO BE TAKEN
Minimum	Maximum	
0		Suspend payment; print ZERO HOURS error message
1	48	Process payment normally
49	80	Process payment normally; include overtime computation. Print overtime notification list.
81	168	Suspend payment; normal maximum overtime exceeded. Print notification to relevant official
169	up	Suspend payment. Impossible number of hours for one week. Probably a data input error. Notify the payroll supervisor.

Exhibit 3 Table of limits used in the program segment illustrated in Exhibit 2.

OTHER WORKING PAPERS

Working Paper Number	
1	"Reflections on Evolving Competitive Aspects in Major Industries," by Sidney C. Sufrin
2	"A Scoring System to Aid in Evaluating a Large Number of Proposed Public Works Projects by a Federal Agency," by M. Lynn Spruill
3	"The Delphi Method: A Systems Approach to the Utilization of Experts in Technological and Environmental Forecasting," by John D. Ludlow
4	Out of print (to be published in a forthcoming issue of the Journal of Marketing Research)
5	"Performance Issues of the Automobile Industry," by Sidney C. Sufrin, H. Paul Root, Roger L. Wright, and Fred R. Kaen
6	"Management Experience with Applications of Multidimensional Scaling Methods," by James R. Taylor
7	"Profitability and Industry Concentration," by Daryl Winn
8	"Why Differences in Buying Time? A Multivariate Approach," by Joseph W. Newman and Richard Staelin
9	"The Contribution of the Professional Buyer to the Success or Failure of a Store," by Claude R. Martin
10	"An Empirical Comparison of Similarity and Preference Judgments in a Unidimensional Context," by James R. Taylor
11	"A Marketing Rationale for the Distribution of Automobiles," by H. O. Helmers
12	"Global Capital Markets," by Merwin H. Waterman

OTHER WORKING PAPERS (Continued)

Working Paper Number	
13	"The Theory of Double Jeopardy and Its Contribution to Understanding Consumer Behavior," by Claude R. Martin, Jr.
14	"A Study of the Sources and Uses of Information in the Development of Minority Enterprise--A Proposal for Research on Entrepreneurship," by Patricia L. Braden and Dr. H. Paul Root