IT Controls Part III: Systems Development, Program Changes, and Application Controls

LEARNING OBJECTIVES

After studying this chapter, you should:

- Be familiar with the controls and audit tests relevant to the systems development process.
- Understand the risks and controls associated with program change procedures and the role of the source program library.
- Understand the auditing techniques (CAATTs) used to verify the effective functioning of application controls.
- Understand the auditing techniques used to perform substantive tests in an IT environment.

This chapter concludes our treatment of IT controls as outlined in the COSO control framework. The focus of the chapter is on Sarbanes-Oxley (SOX) compliance regarding systems development, program changes, and application controls. This chapter examines the risks, controls, audit objectives, and tests of controls that may be performed to satisfy compliance or attest responsibilities. The chapter concludes with a discussion of embedded audit modules and generalized audit software used for substantive testing.

Systems Development Controls

Chapters 13 and 14 presented the systems development life cycle (SDLC) as a multiphase process by which organizations satisfy their formal information needs. An important point at this juncture is that specific SDLC steps will vary from firm to firm. In reviewing the effectiveness of a particular systems development methodology, the accountant should focus on the controllable activities common to all systems development approaches. These are outlined in the following section.

Controlling Systems Development Activities

This section and the one that follows examine several controllable activities that distinguish an effective systems development process. The six activities discussed deal with the authorization, development, and implementation of new systems. Controls over systems maintenance are presented in the next section.

Systems Authorization Activities

All systems should be properly authorized to ensure their economic justification and feasibility. This requires a formal environment in which users submit requests to systems professionals in written form.

User Specification Activities

Users need to be actively involved in the systems development process. The technical complexity of the system should not stifle user involvement. Regardless of the technology involved, the user should create a detailed written description of his or her needs. The creation of a user specification document often involves the joint efforts of the user and systems professionals. However, this document must remain a statement of user needs. It should describe the user's view of the problem, not that of the systems professionals.

Technical Design Activities

The technical design activities translate user specifications into a set of detailed technical specifications for a system that meets the user's needs. The scope of these activities includes systems analysis, feasibility analysis, and detailed systems design. The adequacy of these activities is measured by the quality of the documentation that emerges from each phase. Documentation is both a control and evidence of control and is critical to the system's long-term success. We discussed specific documentation requirements including designer, operator, user, and auditor documentation in Chapter 14.

Internal Audit Participation

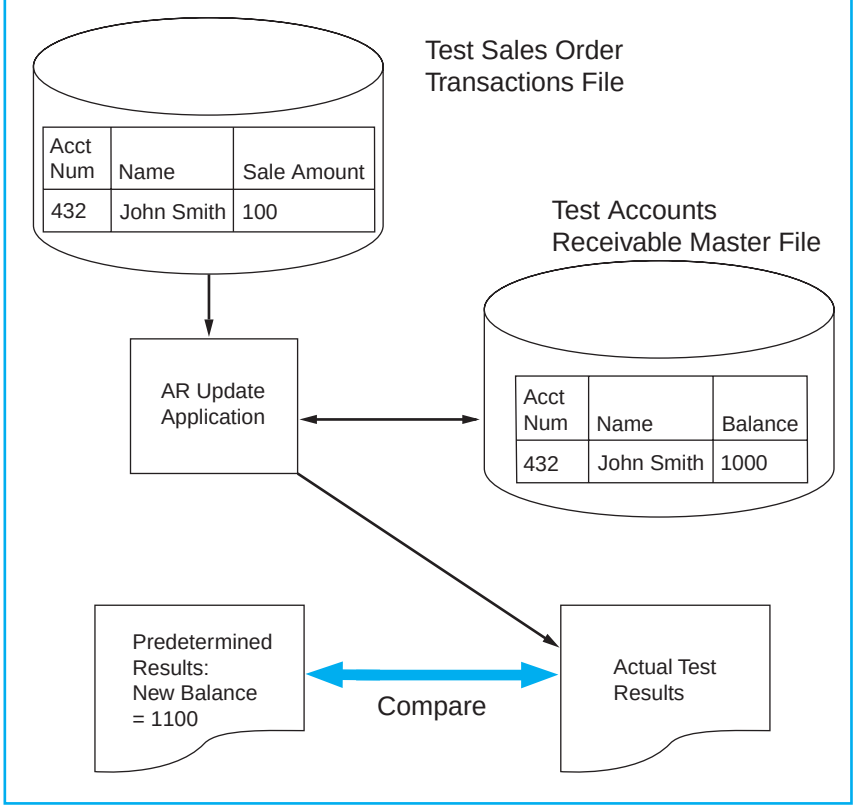
To meet the governance-related expectations of management under SOX, an organization's internal audit department needs to be independent, objective, and technically qualified. As such, the internal auditor can play an important role in the control of systems development activities. The internal auditor can serve as a liaison between users and the systems professionals to ensure an effective transfer of knowledge. An internal audit group, astute in computer technology and possessing a solid grasp of the business problems to be solved, is invaluable to the organization during all phases of the SDLC. Internal auditors should therefore become formally involved at the inception of the systems development process to oversee the definition of user needs requirements and appropriate controls. Furthermore, this involvement should continue throughout all phases of development and maintenance activities.

Program Testing

All program modules must be thoroughly tested before they are implemented. Figure 17-1 shows a program testing procedure involving the creation of hypothetical master files and transactions files that the tested modules process. The results of the tests are then compared against predetermined results to identify programming and logic errors. For example, a programmer testing the logic of the accounts receivable update module illustrated in Figure 17-1 might create an accounts receivable master file record for John Smith with a current balance of $1,000 and a sales order transaction record for $100. Before performing the update test, the programmer concludes that a new balance of $1,100 should result. To verify the module's internal logic, the programmer compares the actual results obtained from the test with the predetermined results. This is a very simple example of a program test. Actual testing would be extensive and involve many transactions that test all aspects of the module's logic.

The task of creating meaningful test data is time consuming. This should not, however, be considered a single-use activity. As we shall later see, some aspects of application control testing require test data. To efficiently meet future **audit objectives,** test data prepared during systems implementation should be preserved. This will give the auditor a frame of reference for designing and evaluating future audit tests. For example, if a program has undergone no maintenance changes since its implementation, the test results from the audit should be identical to the original test results. Having a basis for comparison, the auditor can thus quickly verify the integrity of the program code. On the other hand,

FIGURE 17-1 Program Testing Procedures



if changes have occurred, the original test data can provide a baseline for assessing the impact of changes. The auditor can thus concentrate tests of application controls on areas where computer logic was changed.

User Test and Acceptance Procedures

Prior to system implementation, the individual modules of the system need to be formally and rigorously tested as a whole. The test team should comprise of user personnel, systems professionals, and internal auditors. The details of the tests performed and their results need to be formally documented and analyzed. Once the test team is satisfied that the system meets its stated requirements, the system can be transferred to the user.

Many consider the formal testing and acceptance event to be the most important control over the systems development process. This is the last point at which the user can determine the system's acceptability prior to it going into service. Whereas discovering a major flaw at this juncture is costly, discovering it during day-to-day operations may be devastating.

Audit Objectives Relating to Systems Development

The auditor's objectives are to ensure that (1) systems development activities are applied consistently and in accordance with management's policies to all systems development projects; (2) the system as originally implemented was free from material errors and fraud; (3) the system was judged necessary and justified at various checkpoints throughout the SDLC; and (4) system documentation is sufficiently accurate and complete to facilitate audit and maintenance activities.

Tests of Systems Development Controls

The auditor should select a sample of completed projects (completed in both the current period and previous periods) and review the documentation for evidence of compliance with stated systems development policies. Specific points for review should include determining that:

- User and computer services management properly authorized the project.
- A preliminary feasibility study showed that the project had merit.
- A detailed analysis of user needs was conducted that resulted in alternative conceptual designs.
- A cost-benefit analysis was conducted using reasonably accurate figures.
- The detailed design was an appropriate and accurate solution to the user's problem.
- Test results show that the system was thoroughly tested at both the individual module and the total system level before implementation. (To confirm these test results, the auditor may decide to retest selected elements of the application.)
- There is a checklist of specific problems detected during the conversion period, along with evidence that they were corrected in the maintenance phase.
- Systems documentation complies with organizational requirements and standards.

Controlling Program Change Activities

Upon implementation, the information system enters the maintenance phase of the SDLC. This is the longest period in the SDLC, often spanning several years. Most systems do not remain static throughout this period. Rather, they undergo substantial changes that often constitute, in dollars, an amount many times their original implementation cost.

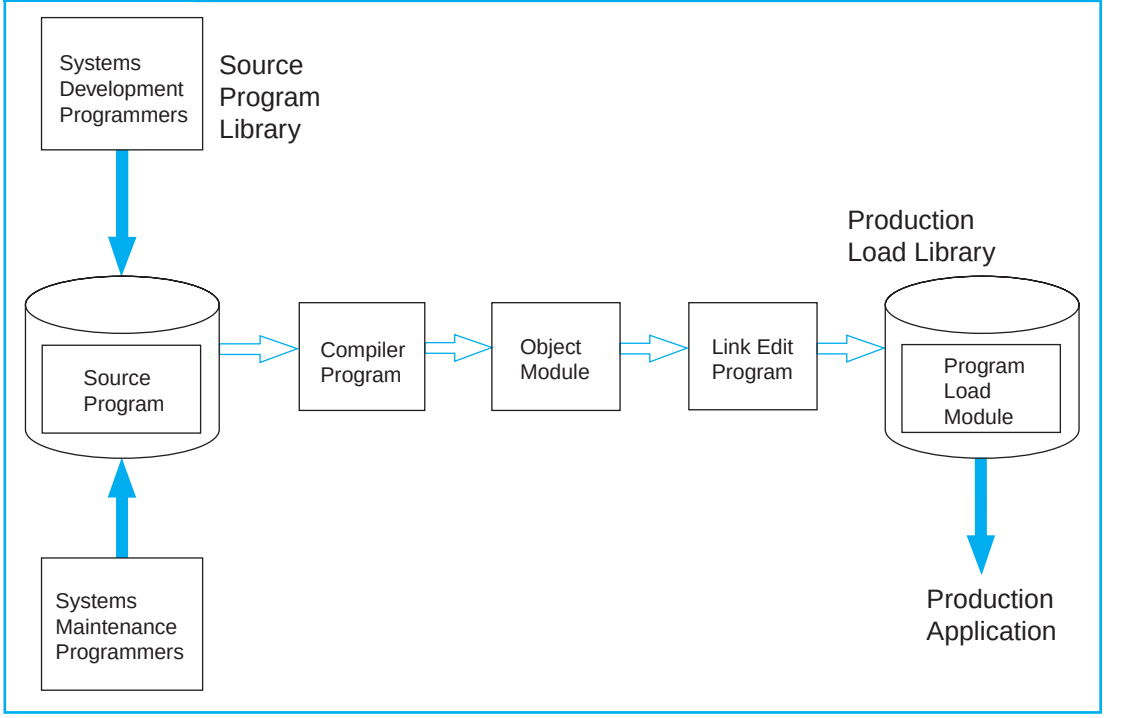
Little is served by designing and implementing controls over systems development activities if control is not continued into the maintenance phase. Maintenance access to systems increases the risk that logic will be corrupted either by accident or intent to defraud. To minimize the risk, all maintenance actions should require, as a minimum, four controls: formal authorizations, technical specifications, testing, and documentation updates. In other words, maintenance activities should be given essentially the same treatment as new development. The extent of the change and its potential impact on the system should govern the degree of control applied. When maintenance causes extensive changes to program logic, additional controls, such as involvement by the internal auditor and additional user test, and acceptance procedures may be necessary.

Source Program Library Controls

Even with formal maintenance procedures in place, individuals who gain unauthorized access to programs threaten application integrity. The remainder of this section deals with control techniques and procedures for reducing this risk.

In larger computer systems, application program modules are stored in source code form on magnetic disks called the source program library (SPL). Figure 17-2 illustrates the relationship between the SPL and other key components of the operating environment. This material presumes an understanding of the program compilation process. If you are uncertain about the meaning of the terms *source program*, *compiler*, and *load module*, review the section on language translators on the book's web page located at http://academic.cengage.com.

FIGURE 17-2 **Uncontrolled Access to the Source Program Library**



Executing a production application requires that the source code be compiled and linked to a load module that the computer can process. As a practical matter, programs in their compiled state are secure and free from the threat of unauthorized modification. At this point, the source code is not needed for the application to run. In fact, we could destroy it if no future changes were ever to be made to the application. To make such a change, however, requires changing the logic of the source code on the SPL. This is then recompiled and linked to create a new load module that incorporates the changed code. Clearly, protecting the source code on the SPL is central to protecting the production application.

The Worst-Case Situation: No Controls

Figure 17-2 shows the SPL without controls. In this situation, access to application programs is completely unrestricted. Legitimate maintenance programmers or others may access any programs stored in the library, which has no provision for detecting an unauthorized intrusion. Because these programs are open to unauthorized changes, no basis exists for relying on the effectiveness of controls designed into them. Even testing these controls proves only that they work now, but says nothing about how they worked last week or last month. In other words, with no control over access to the SPL, a program's integrity during the period in question cannot be established.

A Controlled SPL Environment

Controlling the SPL requires SPL management system (SPLMS) software. Figure 17-3 illustrates this approach. The black box surrounding the SPL signifies the SPLMS, which controls four critical functions: (1) storing programs on the SPL, (2) retrieving programs for maintenance purposes, (3) deleting obsolete programs from the library, and (4) documenting program changes to provide an audit trail of the changes.

You may have recognized the similarities between the SPLMS and a database management system (DBMS). This is a valid analogy, the difference being that SPL software manages program files and DBMSs manage data files. The computer manufacturer may supply SPLMS software as part of the **operating system,** or the software may be purchased through vendors.

The mere presence of an SPLMS does not guarantee program integrity. Again, we can draw an analogy with the DBMS. To achieve data integrity, the DBMS must be properly used; control does not come automatically—it must be planned. Likewise, an SPL requires specific planning and control techniques to ensure program integrity. The control techniques discussed in the following section address the most vulnerable areas and should be considered minimum SPL controls.

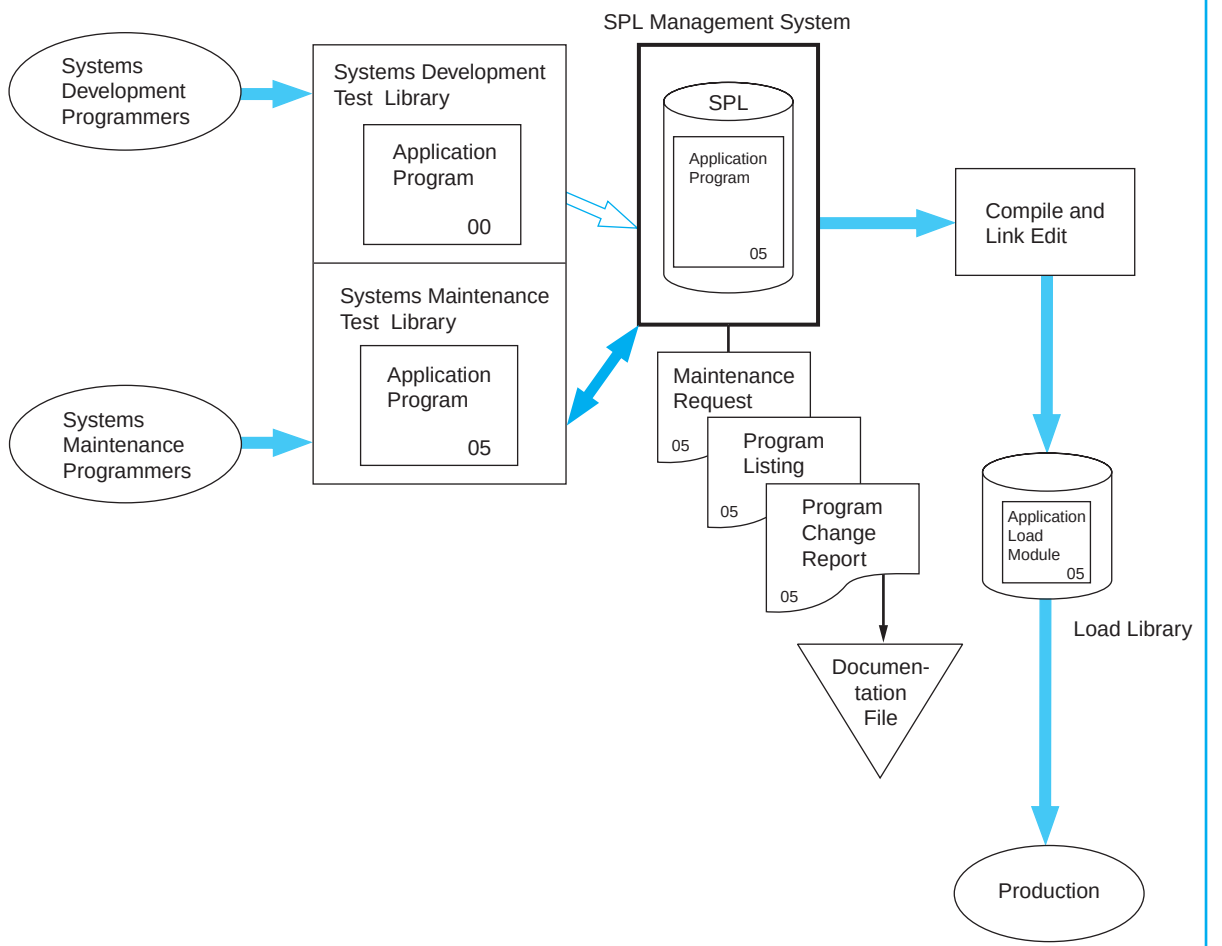
Password Control

Password control over the SPL is similar to password controls in a DBMS. Every financially significant program stored in the SPL can be assigned a separate password. As previously discussed, passwords have drawbacks. When more than one person is authorized to access a program, preserving the secrecy of a shared password is a problem. Because responsibility for the secrecy of a shared password lies with the group rather than with an individual, personal accountability is reduced.

Separation of Test Libraries

Figure 17-3 illustrates an improvement on the shared password approach through the creation of separate password-controlled libraries (or directories) for each programmer.

FIGURE 17-3 **Source Program Library under the Control of SPL Management Software**



Under this concept, a strict separation is maintained between the production programs that are subject to maintenance in the SPL and those being developed. Production programs are copied into the programmer's library for maintenance and testing purposes only. Direct access to the production SPL is limited to a specific librarian group that must approve all requests to modify, delete, and copy programs.

An enhancement to this control feature is the implementation of program naming conventions. The name assigned to a program clearly distinguishes it as being either a test or a production program. When a program is copied from the production SPL to the programmer's library, it is given a temporary test name. When the program is returned to the SPL, it is renamed with its original production name. This technique greatly reduces the risk of accidentally running an untested version of a program in place of the production program.

Audit Trail and Management Reports

An important feature of SPL management software is the creation of reports that enhance management control and support the audit function. The most useful of these are program modification reports, which describe in detail all program changes (additions and deletions) to

each module. These reports should be part of the documentation file of each application to form an audit trail of program changes over the life of the application. During an audit, the reports can be reconciled against program maintenance requests to verify that only approved changes were implemented. For example, if a programmer attempted to use a legitimate maintenance event as an opportunity to commit program fraud, the unauthorized code changes would be documented in the program modification report. These reports can be produced as hard copy or digital and can be governed by password control, thus limiting access to management and auditors.

Program Version Numbers

The SPLMS assigns a version number automatically to each program stored on the SPL. When programs are first placed in the libraries (at implementation), they are assigned version number zero. With each modification to the program, the version number is increased by one. For instance, after five authorized maintenance changes, the production program will be Version 05, as illustrated in Figure 17-3. This feature, when combined with audit trail reports, provides a basis for detecting unauthorized changes to the application program. An unauthorized change is signaled by a version number on the production load module that cannot be reconciled to the number of authorized changes. For example, if 10 changes were authorized but the production program is Version 12, then two possible control violations may have happened: (1) authorized changes occurred, which for some reason went undocumented, or (2) unauthorized changes were made, which incremented the version numbers. We will discuss this issue in more detail later.

Controlling Access to Maintenance Commands

Powerful maintenance commands are available for most library systems that can be used to alter or eliminate program passwords, alter the program version number, and temporarily modify a program without generating a record of the modification.

There are a number of legitimate technical reasons why systems designers must sometimes use these commands. If not controlled, however, maintenance commands open the possibility of unauthorized, and perhaps undocumented, program modifications. Hence, access to the maintenance commands themselves should be password controlled, and management or an IT security group should control the authority to use them.

Audit Objectives Relating to Systems Maintenance

The auditor's objectives are to determine that (1) maintenance procedures protect applications from unauthorized changes, (2) applications are free from material errors, and (3) program libraries are protected from unauthorized access.

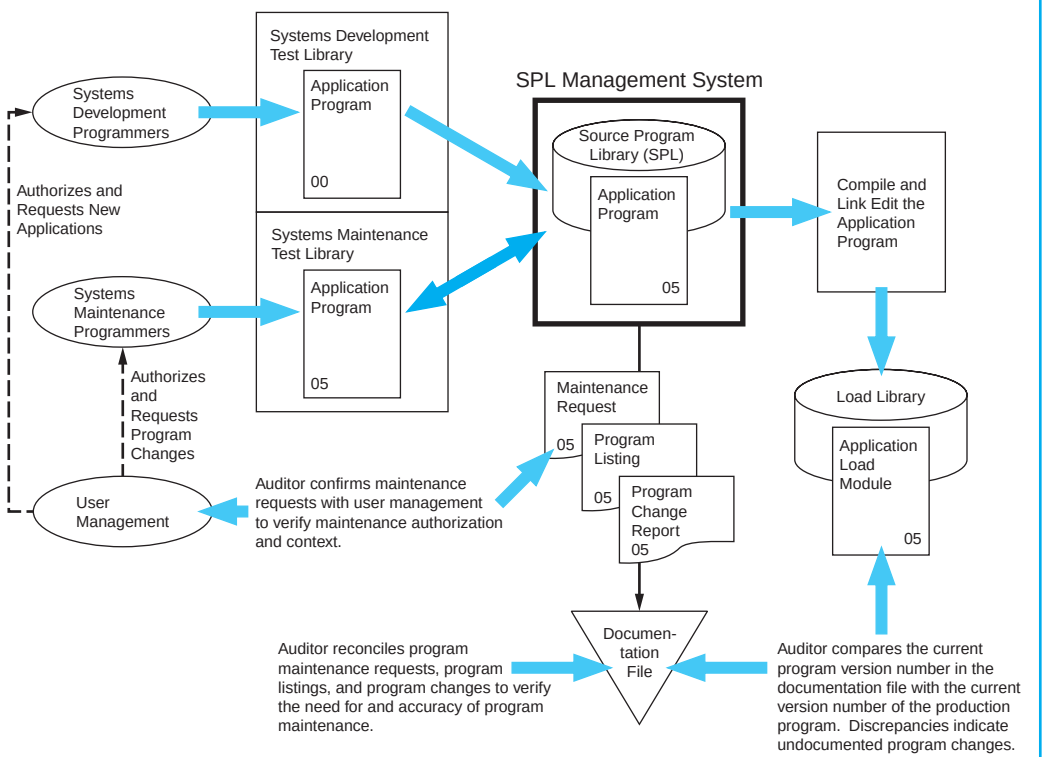
The **tests of controls** necessary to achieve each of these objectives are examined in the following section. The discussion assumes that the organization employs SPL software to control program maintenance. Without such software, achieving the audit objectives may be impossible. The procedures described below are illustrated in Figure 17-4.

Audit Procedures for Identifying Unauthorized Program Changes

To establish that program changes were authorized, the auditor should examine the audit trail of program changes for a sample of applications that have undergone maintenance. The auditor can confirm that authorization procedures were followed by performing the following tests of controls.

Reconcile Program Version Numbers. The permanent file of the application should contain program change authorization documents that correspond to the current version

FIGURE 17-4 Auditing SPL Software System



number of the production application. In other words, if the production application is in its tenth version, there should be 10 program change authorizations in the permanent file as supporting documentation.[1] Any discrepancies between version numbers and supporting documents may indicate that unauthorized changes were made.

Confirm Maintenance Authorization. The program maintenance authorization should indicate the nature of the change requested and the date of the change. The appropriate management from both computer services and the user departments should also sign and approve it. The auditor should confirm the facts contained in the maintenance authorization and verify the authorizing signatures with the managers involved.

Audit Procedures for Identifying Application Errors

The auditor can perform three types of tests of controls—reconcile the source code, review the test results, and retest the program—to determine that programs are free from material errors.

Reconcile the Source Code. Each application's permanent file should contain the current program listing and listings of all changes made to the application. These documents describe in detail the application's maintenance history. In addition, the nature of the program change should be clearly stated on the program change authorization

1 In most systems, a program in its original (unmodified) state has a version number of 00. Thus, ten changes will give a version number of 10.

document. The auditor should select a sample of applications and reconcile each program change with the appropriate authorization documents. The modular approach to systems design (creating applications that comprise many small discrete program modules) greatly facilitates this testing technique. The reduced complexity of these modules enhances the auditor's ability to identify irregularities that indicate errors, omissions, and potentially fraudulent programming codes.

Review the Test Results. Every program change should be thoroughly tested before being implemented. Program test procedures should be properly documented as to the test objectives, test data, and processing results. The auditor should review this record for each significant program change to establish that testing was sufficiently rigorous to identify any errors.

Retest the Program. The auditor can retest the application to confirm its integrity. We examine several techniques for application testing later in the chapter.

Audit Procedures for Testing Access to Libraries

The existence of a secure program library is central to preventing errors and program fraud. One control method is to assign library access privileges only to system librarians. Their function is to retrieve applications from the program libraries for maintenance and to restore the modified programs to the library. Thus maintenance programmers test applications in their private libraries but do not have access to the program library. The auditor may perform the following tests of controls to assess program library security.

Review Programmer Authority Tables. The auditor can select a sample of programmers and review their access authority. The programmer's authority table will specify the libraries a programmer may access. These authorizations should be matched against the programmer's maintenance authority to ensure that no irregularities exist.

Test Authority Table. The auditor may violate the authorization rules in an attempt to access unauthorized libraries to test the programmer's access privileges. The operating system should deny any such attempt.

Application Controls

In addition to IT general controls, SOX requires management and auditors to consider application controls relevant to financial reporting. Application controls are associated with specific applications, such as payroll, purchases, and cash disbursements systems. These fall into three broad categories: input controls, processing controls, and output controls.

Input Controls

Input controls are programmed procedures (routines) that perform tests on transaction data to ensure that they are free from errors. Input control routines should be designed into the system at different points, depending on whether transaction processing is real time or batch. Input controls in real-time systems are placed at the data collection stage to monitor data as they are entered from terminals. Batch systems often collect data in transaction files, where they are temporarily held for subsequent processing. In this case, input control tests are performed as a separate procedure (or run) prior to the master file update process. In any case, transaction data should never be used to update master files until the transactions have been tested for validity, accuracy, and **completeness.** If a

record fails an input control test, it is flagged as an error record. Later, we will see how to deal with these records. The following are examples of input controls.

Check Digit. Data codes are used extensively in transaction processing systems for representing such things as customer accounts, items of inventory, and general ledger accounts in the chart of accounts. If the data code of a particular transaction is entered incorrectly and goes undetected, then a transaction processing error will occur, such as posting to the wrong account. Two common classes of data input errors cause such processing problems: transcription errors and transposition errors.

Transcription errors are divided into three categories:

1. Addition errors occur when an extra digit or character is added to the code. For example, inventory item number 83276 is recorded as 832766.
2. Truncation errors occur when a digit or character is removed from the end of a code. In this type of error, the inventory item above would be recorded as 8327.
3. Substitution errors are the replacement of one digit in a code with another. For example, code number 83276 is recorded as 83266.

Transposition errors are of two types.

1. Single transposition errors occur when two adjacent digits are reversed. For instance, 83276 is recorded as 38276.
2. Multiple transposition errors occur when nonadjacent digits are transposed. For example, 83276 is recorded as 87236.

These problems may be controlled using a **check digit.** This is a control digit (or digits) added to the data code when it is originally assigned that allows the integrity of the code to be established during subsequent processing. The check digit can be located anywhere in the code, as a prefix, a suffix, or embedded someplace in the middle. The simplest form of check digit is to sum the digits in the code and use this sum as the check digit. For example, for the customer account code 5372, the calculated check digit would be

$$5 + 3 + 7 + 2 = 17$$

By dropping the tens column, the check digit 7 is added to the original code to produce the new code 53727. The entire string of digits (including the check digit) becomes the customer account number. During data entry, the system can recalculate the check digit to ensure that the code is correct. This technique will detect only transcription errors. For example, if a substitution error occurred and the above code were entered as 52727, the calculated check digit would be 6 ($5 + 2 + 7 + 2 = 16 = 6$), and the error would be detected. However, this technique would fail to identify transposition errors. For example, transposing the first two digits yields the code 35727, which still sums to 17 and produces the check digit 7. This error would go undetected.

A popular check digit technique for dealing with transposition errors is modulus 11. Using the code 5372, the steps in this technique are outlined next.

1. *Assign weights.* Each digit in the code is multiplied by a different weight. In this case, the weights used are 5, 4, 3, and 2, shown as follows:

Digit		Weight
5	–	5 = 25
3	–	4 = 12
7	–	3 = 21
2	–	2 = 4

2. *Sum the products:* (25 + 12 + 21 + 4 = 62).
3. *Divide by the modulus.* We are using modulus 11 in this case, giving 62/11 = 5 with a remainder of 7.
4. *Subtract the remainder from the modulus to obtain the check digit:* (11 − 7 = 4 [check digit]).
5. *Add the check digit to the original code to yield the new code:* 53724.

Using this technique to recalculate the check digit during processing, a transposition error in the code will produce a check digit other than 4. For example, if the code above was incorrectly entered as 35724, the recalculated check digit would be 6.

Missing Data Check. Some programming languages are restrictive as to the justification (right or left) of data within the field. If data are not properly justified or if a character is missing (has been replaced with a blank), the value in the field will be improperly processed. In some cases, the presence of blanks in a numeric data field may cause a system failure. When the control routine detects a blank where it expects to see a data value, the error is flagged.

Numeric–Alphabetic Check. This control identifies when data in a particular field are in the wrong form. For example, a customer's account balance should not contain alphabetic data, and the presence of it will cause a data processing error. Therefore, if alphabetic data are detected, the error record flag is set.

Limit Check. Limit checks are used to identify field values that exceed an authorized limit. For example, assume the firm's policy is that no employee works more than 44 hours per week. The payroll system input control program can test the hours-worked field in the weekly payroll records for values greater than 44.

Range Check. Many times, data have upper and lower limits to their acceptable values. For example, if the range of pay rates for hourly employees in a firm is between $8 and $20, this control can examine the pay rate field of all payroll records to ensure that they fall within this range. The purpose of this control is to detect keystroke errors that shift the decimal point one or more places. It would not detect an error where a correct pay rate of, say, $9 is incorrectly entered as $15.

Reasonableness Check. The error above may be detected by a test that determines if a value in one field, which has already passed a limit check and a range check, is reasonable when considered along with data in other fields of the record. For example, an employee's pay rate of $18 per hour falls within an acceptable range. This rate is excessive, however, when compared to the employee's job skill code of 693; employees in this skill class should not earn more than $12 per hour.

Validity Check. A validity check compares actual field values against known acceptable values. This control is used to verify such things as transaction codes, state abbreviations, or employee job skill codes. If the value in the field does not match one of the acceptable values, the record is flagged as an error.

This is a frequently used control in cash disbursement systems. One form of cash disbursement fraud involves manipulating the system into making a fraudulent payment to a nonexistent vendor. To prevent this, the firm may establish a list of valid vendors with whom it does business exclusively. Thus, before payment of any trade obligation,

the validation program matches the vendor number on the cash disbursement voucher against the valid vendor list. If the code does not match, payment is denied, and management reviews the transaction.

Processing Controls

After passing through the data input stage, transactions enter the processing stage of the system. Processing controls are programmed procedures and may be divided into three categories: batch controls, run-to-run controls, and audit trail controls.

Batch controls are used to manage the flow of high volumes of transactions through batch processing systems. The objective of batch control is to reconcile system output with the input originally entered into the system. This provides assurance that:

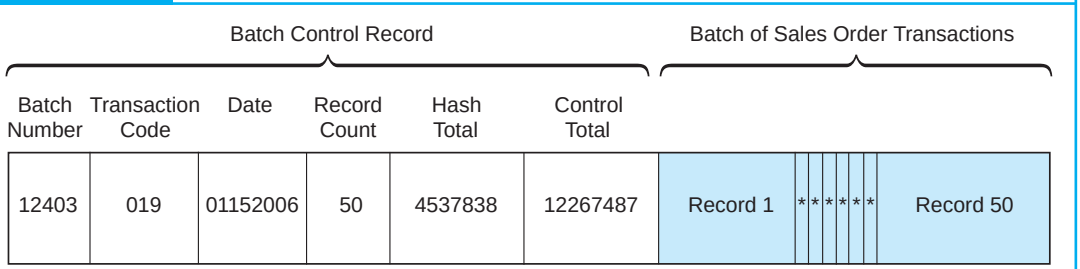
- All records in the batch are processed.
- No records are processed more than once.
- An audit trail of transactions is created from input through processing to the output stage of the system.

Batch control begins at the data input stage and continues through all data processing phases of the system. Batch control involves grouping together into batches similar types of transactions (such as sales orders) and controlling them as a unit of work throughout data processing. To achieve this, a batch control record is created when the batch of transactions is entered into the system. This may be a user department action or a separate data control step. The control record contains relevant information about the batch, such as:

- A unique batch number.
- A batch date.
- A transaction code (indicating the type of transactions, such as a sales order or cash receipt).
- The number of records in the batch (record count).
- The total dollar value of a financial field (batch control total).
- The total of a unique nonfinancial field (hash total).

Figure 17-5 depicts a batch control record in relation to the batch of transactions it describes. The data in the control record are used to assess the integrity of the batch during all subsequent processing. For example, the batch control record in the figure shows a batch of 50 sales order records with a total dollar value of $122,674.87 and a hash total of 4537838.

FIGURE 17-5 **Batch Control Record**

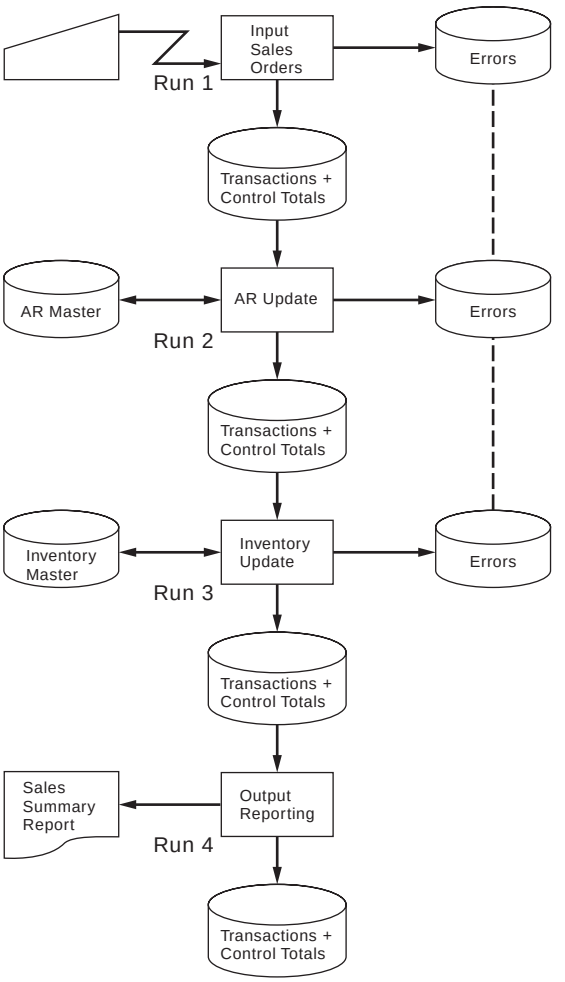


Run-to-run control is the use of batch figures to monitor the batch as it moves from one programmed procedure (run) to another. Thus at various points throughout processing and at the end of processing, the batch totals are recalculated and compared to the batch control record. This ensures that each run in the system processes the batch correctly and completely.

Figure 17-6 illustrates the use of run-to-run control in a sales order system. This application comprises four runs: (1) data input, (2) accounts receivable update, (3) inventory update, and (4) output. At the end of the accounts receivable run, batch control figures are recalculated and reconciled with the control totals passed from the data input run. These figures are then passed to the inventory update run, where they are again recalculated, reconciled, and passed to the output run. Errors detected in each run are flagged and placed in an error file. The run batch control figures are then adjusted to reflect the deletion of these records.

Notice from Figure 17-6 that error records may be placed on the error file at several different points in the process. In a separate procedure (not shown), an authorized user

FIGURE 17-6 Run-to-Run Controls



representative will make corrections to the error records and resubmit them as a special batch for reprocessing. Errors detected during processing require careful handling, because these records may already be partially processed. Simply resubmitting the corrected records to the system at the data input stage may result in processing portions of these transactions twice. Two methods are used to deal with this complexity. The first is to reverse the effects of the partially processed transactions and resubmit the corrected records to the data input stage. The second method is to reinsert corrected records into the processing stage at which the error was detected.

The term **hash total,** which was used in the preceding discussion, is the summation of a nonfinancial field to keep track of the records in a batch. Any numeric field, such as a customer's account number, a purchase order number, or an inventory item number, may be used to calculate a hash total. In the following example, the sales order number (SO#) field for an entire batch of sales order records is summed to produce a hash total.

SO#	
14327	
67345	
19983	
•	
•	
•	
•	
88943	
96543	
4537838	(hash total)

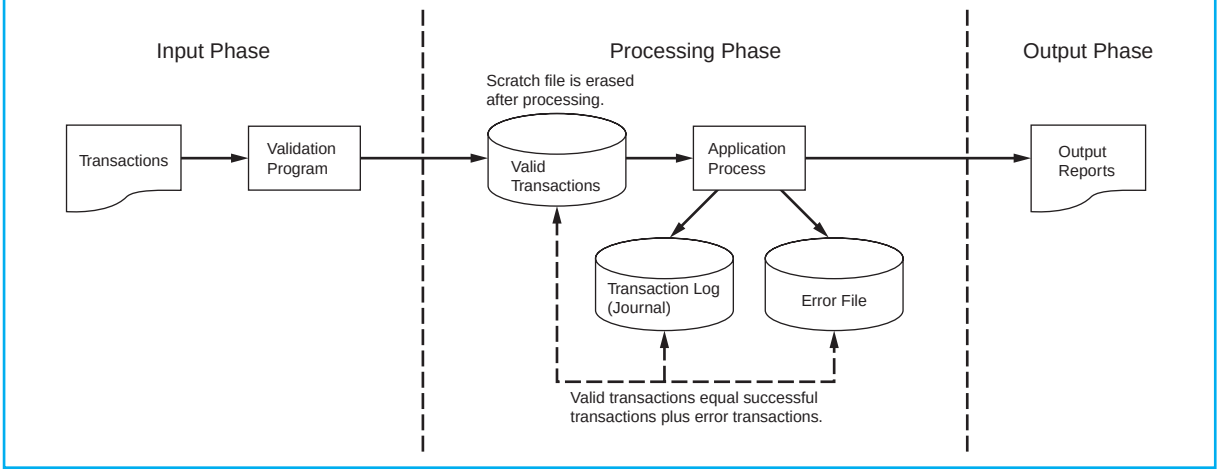
Let's see how we can use this seemingly meaningless number. Assume that after this batch of records is created, someone replaced one of the sales orders in the batch with a fictitious record of the same dollar amount. How would the batch control procedures detect this irregularity? Both the record count and the dollar amount control totals would still balance. The hash total that the batch control procedures calculated would, however, not balance. The irregularity would thus be detected.

Audit trail controls in an IT environment ensure that every transaction can be traced through each stage of processing from its economic source to its presentation in financial statements. The following are examples of audit trail control.

Transaction Logs. Every transaction the system successfully processes should be recorded on a transaction log, which serves as a journal. Figure 17-7 shows this process. Two reasons underscore the importance of this log. First, the transaction log is a permanent record of transactions, though the input transaction file is typically a temporary file. Once processed, the records on the input file are erased to make room for the next batch of transactions. Second, not all of the records in the input file may be successfully processed. Some of them will fail tests during subsequent processing and will be passed to an error file. A transaction log contains only successful transactions—those that have changed account balances. The transaction log and error files combined should account for all the transactions in the batch. The validated transaction file may then be scratched with no loss of data.

Log of Automatic Transactions. The system triggers some transactions internally. For example, when inventory drops below the reorder point, the system automatically

FIGURE 17-7 Transaction Log to Preserve the Audit Trail



generates a purchase order. To maintain an audit trail of these activities, all internally generated transactions must be placed in a transaction log.

Transaction Listings. The system should produce a (hard-copy) transaction listing of all successful transactions. These listings should go to the appropriate users to facilitate reconciliation with input. In addition, the responsible end user should receive a detailed listing of all internally generated transactions.

Output Controls

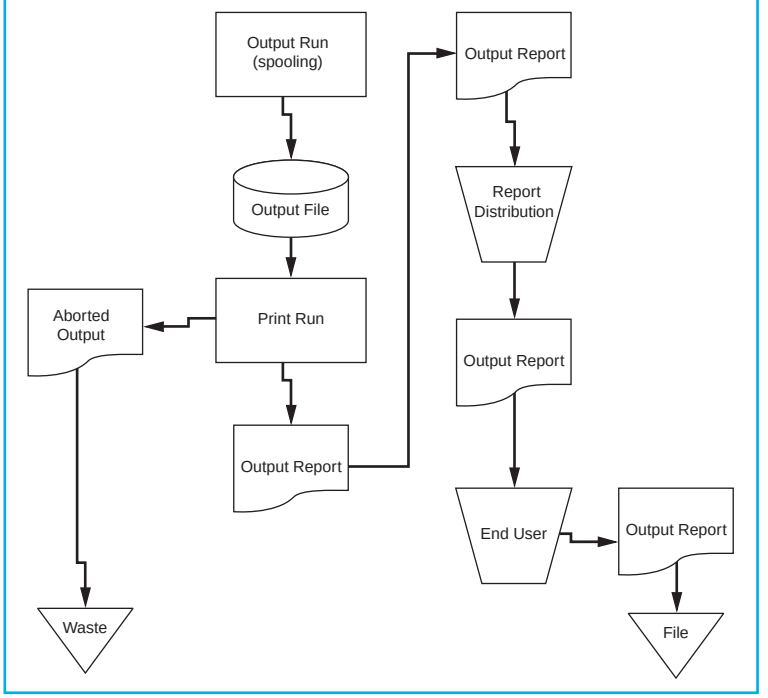
Output controls are a combination of programmed routines and other procedures to ensure that system output is not lost, misdirected, or corrupted and that privacy is not violated. Exposures of this sort can cause serious disruptions to operations and may result in financial losses to a firm. For example, if the checks a firm's cash disbursements system produces are lost, misdirected, or destroyed, trade accounts and other bills may go unpaid. This could damage the firm's credit rating and result in lost discounts, interest, or penalty charges. If the privacy of certain types of output is violated, a firm could have its business objectives compromised or could become exposed to litigation. Examples of privacy exposures include the disclosure of trade secrets, patents pending, marketing research results, and patient medical records. This section examines output exposures and controls for both hard copy and digital output.

Controlling Hard Copy Output

Batch systems usually produce hard copy, which typically requires the involvement of intermediaries in its production and distribution. Figure 17-8 shows the stages in this output process and serves as the basis for this section.

Output Spooling. In large-scale data processing operations, output devices such as line printers can become backlogged with many programs simultaneously demanding limited resources. This can cause a bottleneck and adversely affect system throughput. To ease this burden, applications are often designed to direct their output to a magnetic disk file

FIGURE 17-8 Stages in the Output Process



rather than print it directly. This is called **spooling.** Later, when printer resources become available, the output files are printed.

The creation of an output file as an intermediate step in the printing process presents an added exposure. A computer criminal may use this opportunity to:

1. Access the output file and change critical data values (such as dollar amounts on checks). The printer program will then print the fallacious output as if the system produced it.
2. Access the file and change the number of copies of output to be printed. The extra copies may then be removed without notice during the printing stage.
3. Make a copy of the output file to produce illegal output reports.
4. Destroy the output file before output printing takes place.

The management and auditors need to be aware of these potential exposures and ensure that proper access and backup procedures are in place to protect output files. We discussed file access and backup controls in Chapter 15.

Print Programs. When a printer becomes available, the print run program produces hard-copy output from the output file. Print programs are often complex systems that require operator intervention. Four common types of operator actions are:

1. Pausing the print program to load the correct type of output documents (check stocks, invoices, or other special forms).
2. Entering parameters that the print run needs, such as the number of copies to be printed.

3. Restarting the print run at a prescribed checkpoint after a printer malfunction.
4. Removing printed output from the printer for review and distribution.

Print program controls should be designed to deal with two types of exposures present in this environment: (1) the production of unauthorized copies of output and (2) employee browsing of sensitive data. Some print programs allow the operator to specify more copies of output than the output file calls for, which allows for the possibility of producing unauthorized copies of output. One way to control this is to employ output document controls. This is feasible only when dealing with prenumbered invoices for billing customers or prenumbered check stock. At the end of the run, the number of copies the output file specifies should be reconciled with the actual number of output documents used.

To prevent operators and others from viewing sensitive output, special multipart paper can be used, with a grayed-out top copy to prevent the print from being read. This type of product is often used for payroll check printing. An alternative privacy control is to direct the output to a special remote printer that can be closely supervised.

Waste. Computer output waste is a potential source of exposure. Aborted reports and the carbon copies from multipart paper need to be disposed of properly. Computer criminals disguised as janitorial staff have been known to sift through trash cans searching for carelessly discarded output that is presumed to be of no value. From such trash, computer criminals may obtain information about a firm's market research, credit ratings of its customers, or even trade secrets, which they can sell to a competitor. Computer waste is also a source of passwords that a perpetrator may use to access the firm's computer system. To control against this threat, all sensitive computer output should be passed through a paper shredder.

Report Distribution. The primary risks associated with the distribution of sensitive reports include their being lost, stolen, or misdirected in transit to the user. The following control techniques can be used:

1. The reports may be placed in a secure mailbox to which only the user has the key.
2. The user may be required to appear in person at the distribution center and sign for the report.
3. A security officer or special courier may deliver the report to the user.

End-User Controls. Once in the hands of the user, output reports should be examined for correctness. Errors the user detects should be reported to the appropriate computer services management. Such errors may be symptoms of an improper systems design, incorrect procedures, errors accidentally inserted during systems maintenance, or unauthorized access to data files or programs. Once a report has served its purpose, it should be stored in a secure location until its retention period has expired and then shredded.

Controlling Digital Output

Digital output can be directed to the user's computer screen or printer. The primary output threat is the interception, disruption, destruction, or corruption of the output message as it passes across the communications network. This threat comes from two types of exposures: (1) exposures from equipment failure and (2) exposures from subversive acts. We discussed techniques for controlling communications exposures in Chapter 16.

Testing Computer Application Controls

The appendix to Chapter 15 described how audit objectives are derived from management assertions such as **existence or occurrence,** completeness, accuracy, **rights and obligations, valuation or allocation,** and **presentation and disclosure.** Depending on the type of account being considered, a particular **management assertion** has different implications for the audit objective to be developed. Once developed, achieving the audit objectives requires designing **audit procedures** to gather evidence that either corroborates or refutes the underlying management assertions. Generally, this involves a combination of tests of application controls and substantive tests of transaction details and account balances.

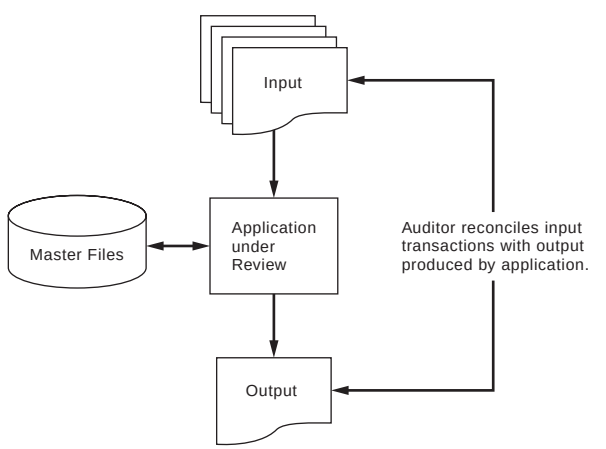
This section deals essentially with the tests of application controls, but at the end we will briefly review techniques for performing substantive tests. Tests of computer application controls follow two general approaches: (1) the black box (around the computer) approach and (2) the white box (through the computer) approach. First, the black box approach is examined. Then, several white box testing techniques are reviewed.

Black Box Approach

Auditors performing black box testing do not rely on a detailed knowledge of the application's internal logic. Instead, they analyze flowcharts and interview knowledgeable personnel in the client's organization to understand the functional characteristics of the application. With an understanding of what the application is supposed to do, the auditor tests the application by reconciling production input transactions processed by the application with output results. The output results are analyzed to verify the application's compliance with its functional requirements. Figure 17-9 illustrates the black box approach.

The advantage of the black box approach is that the application need not be removed from service and tested directly. This approach is feasible for testing applications that are relatively simple. However, complex applications—those that receive input from many sources, perform a variety of complex operations, or produce multiple outputs—often require a more focused testing approach to provide the auditor with evidence of application integrity.

FIGURE 17-9 **Auditing around the Computer—The Black Box Approach**



White Box Approach

The white box (through the computer) approach relies on an in-depth understanding of the internal logic of the application being tested. The white box approach includes several techniques for testing application logic directly. Typically these involve the creation of a small set of test transactions to verify specific aspects of an application's logic and controls. In this way, auditors are able to conduct precise tests, with known variables, and obtain results that they can compare against objectively calculated results. The most common types of tests of controls include the following:

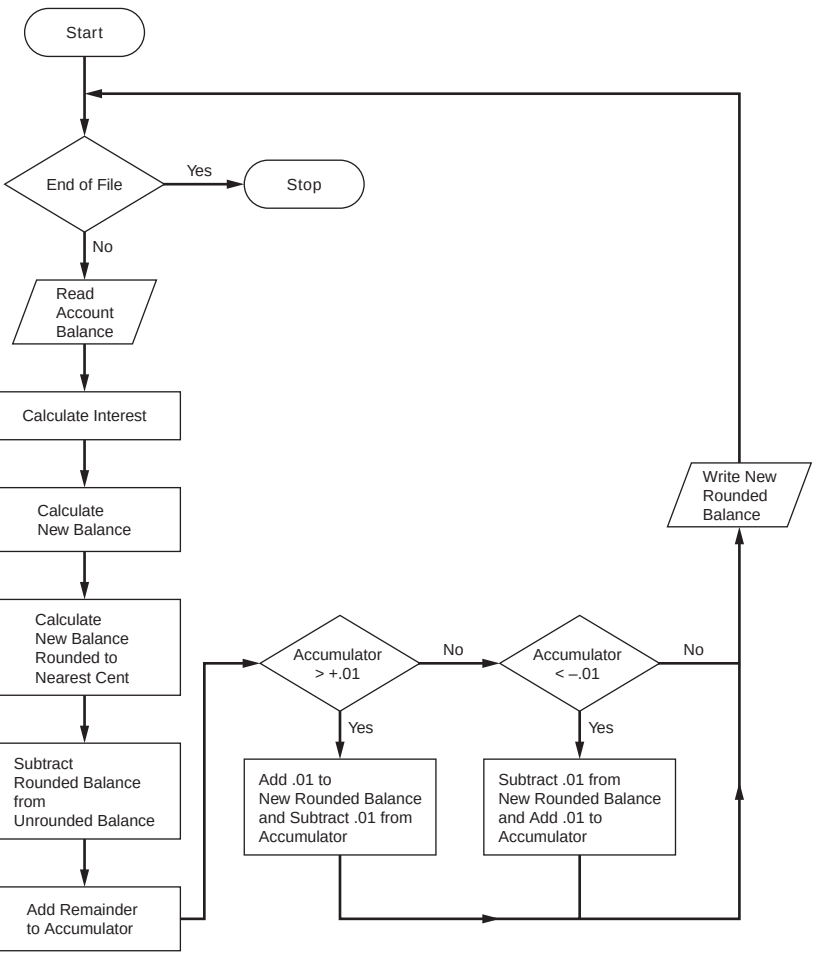
1. **Authenticity tests,** which verify that an individual, a programmed procedure, or a message (such as an electronic data interchange (EDI) transmission) attempting to access a system is authentic. Authenticity controls include user IDs, passwords, valid vendor codes, and authority tables.
2. **Accuracy tests,** which ensure that the system processes only data values that conform to specified tolerances. Examples include range tests, field tests, limit tests, and reasonableness tests.
3. **Completeness tests,** which identify missing data within a single record and entire records missing from a batch. The types of tests performed are field tests, record sequence tests, hash totals, and control totals.
4. **Redundancy tests,** which determine that an application processes each record only once. Redundancy controls include the reconciliation of batch totals, record counts, hash totals, and financial control totals.
5. **Access tests,** which ensure that the application prevents authorized users from unauthorized access to data. Access controls include passwords, authority tables, user-defined procedures, data encryption, and inference controls.
6. **Audit trail tests,** which ensure that the application creates an adequate audit trail. This includes evidence that the application records all transactions in a transaction log, posts data values to the appropriate accounts, produces complete transaction listings, and generates error files and reports for all exceptions.
7. **Rounding error tests,** which verify the correctness of rounding procedures. Rounding errors occur in accounting information when the level of precision used in the calculation is greater than that used in the reporting. For example, interest calculations on bank account balances may have a precision of five decimal places, whereas only two decimal places are needed to report balances. If the remaining three decimal places are simply dropped, the total interest calculated for the total number of accounts may not equal the sum of the individual calculations.

Figure 17-10 shows the logic for handling the rounding error problem. This technique uses an accumulator to keep track of the rounding differences between calculated and reported balances. Note how the sign and the absolute value of the amount in the accumulator determine how rounding affects the customer account. To illustrate, the rounding logic is applied to three hypothetical bank balances (see Table 17-1 on page 818). The interest calculations are based on an interest rate of 5.25 percent.

Failure to properly account for the rounding difference above can result in an imbalance between the total (control) figure and the sum of the detail figures for each account. Poor accounting for rounding differences can also present an opportunity for fraud.

Salami Fraud. Rounding programs are particularly susceptible to the so-called **salami fraud.** This fraud tends to affect large numbers of victims, but each in a minimal way.

FIGURE 17-10 Rounding Error Algorithm



The fraud scheme takes its name from the analogy of slicing a large salami (the fraud objective) into many thin pieces. Each victim gets one of these small pieces and is unaware of being defrauded. For example, a programmer, or someone with access to the rounding program in Figure 17-10, can modify the rounding logic, thus perpetrating a salami fraud, as follows: at the point in the process where the algorithm should increase the current customer's account (that is, the accumulator value is > +.01), the program instead adds one cent to another account—the perpetrator's account. Although the absolute amount of each fraud transaction is small, given the hundreds of thousands of accounts that could be processed, the total amount of the fraud can become significant over time.

Most large public accounting firms have developed special audit software that can detect excessive file activity. In the case of the salami fraud, there would be thousands of entries into the computer criminal's personal account that the audit software may detect. A clever programmer may funnel these entries through several intermediate accounts in order to disguise this activity. The accounts are then posted to a smaller number of

TABLE 17-1 Sample Data

Record 1	
Beginning accumulator balance	.00861
Beginning account balance	2,741.78
Calculated interest	143.94345
New account balance	2,885.72345
Rounded account balance	2,885.72
Adjusted accumulator balance	.01206 (.00345 + .00861)
Ending account balance	**2,885.73 (round up 1 cent)**
Ending accumulator balance	.00206 (.01206 – .01)
Record 2	
Beginning accumulator balance	.00206
Beginning account balance	1,893.44
Calculated interest	99.4056
New account balance	1,992.8456
Rounded account balance	1,992.85
Adjusted accumulator balance	–.00646 (.00206 – .0044)
Ending account balance	**1,992.85 (no change)**
Ending accumulator balance	–.00646
Record 3	
Beginning accumulator balance	–.00646
Beginning account balance	7,423.34
Calculated interest	389.72535
New account balance	7,813.06535
Rounded account balance	7,813.07
Adjusted accumulator balance	–.01111 (.00646 – .00465)
Ending account balance	**7,813.06 (round down 1 cent)**
Ending accumulator balance	.00111

intermediate accounts and finally to the programmer's personal account. By using many levels of accounts in this way, the activity to any single account is reduced, and the audit software may not detect it. There will be a trail, but it can be complicated. The auditor can also use audit software to detect the existence of unauthorized (dummy) files that contain the intermediate accounts used in such a fraud.

White Box Testing Techniques

To illustrate how application controls are tested, this section describes five **computer-assisted audit tools and techniques (CAATTs)** approaches: the test data method, base case system evaluation, tracing, integrated test facility, and parallel simulation.

Test Data Method

The **test data method** is used to establish application integrity by processing specially prepared sets of input data through production applications that are under review. The results of each test are compared to predetermined expectations to obtain an objective assessment of application logic and control effectiveness. The test data technique is illustrated in Figure 17-11. To perform the test data technique, the auditor must obtain a copy of the production version of the application. In addition, test transaction files and test master files must be created. As illustrated in the figure, test transactions may enter the system from magnetic tape, disk, or via an input terminal. Results from the test run will be in the form of routine output reports, transaction listings, and error reports. In addition, the auditor must review the updated master files to determine that account balances have been correctly updated. The test results are then compared with the auditor's expected results to determine if the application is functioning properly. This comparison may be performed manually or through special computer software.

Figure 17-12 lists selected hypothetical transactions and accounts receivable records that the auditor prepared to test a sales order processing application. The figure also shows an error report of rejected transactions and a listing of the updated accounts receivable master file. Any deviations between the actual results and those the auditor expects may indicate a logic or control problem.

Creating Test Data. Creating test data requires a complete set of valid and invalid transactions. Incomplete test data may fail to explore critical branches of application logic and error checking routines. Test transactions should be designed to test all possible input errors, logical processes, and irregularities.

Gaining knowledge of the application's internal logic sufficient to create meaningful test data may demand a large investment in time. The efficiency of this task can, however, be improved through careful planning during systems development. The auditor should save for future use the test data used to test program modules during the implementation phase of the SDLC. If the application has undergone no maintenance since its initial implementation, current audit test results should equal the original test results obtained

FIGURE 17-11 The Test Data Technique

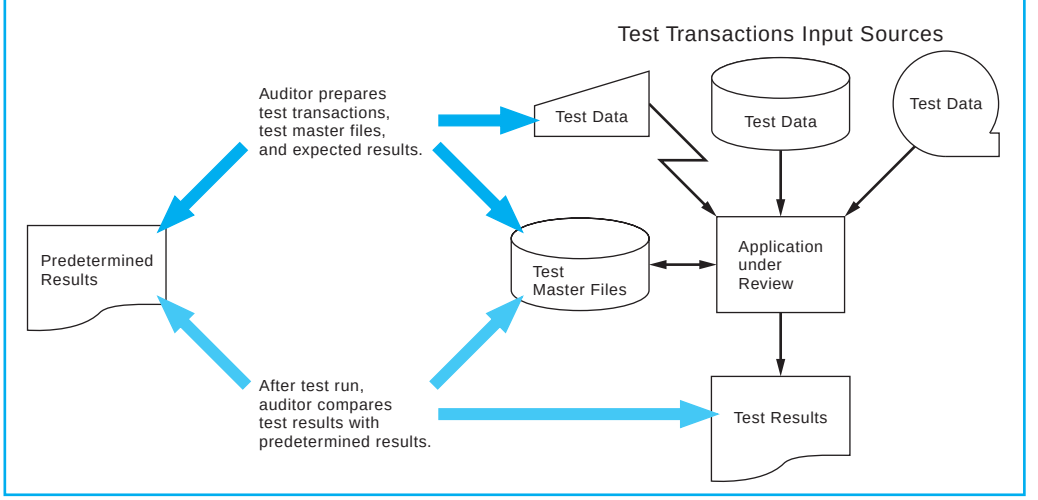


FIGURE 17-12 Example of Test Data and Test Results

Test Transaction File

REC NUM	CUST NUM	CUSTOMER NAME	PART NUM	DESCRIPTION	QNTY	UNIT PRICE	TOTAL PRICE
1	231893	Smith, Joe	AX-612	Water Pump	1	20.00	20.00
2	231893	Azar, Atul	J-912	Gear	3	15.00	45.00
3	245851	Jones, Mary	123-LM	Hose	20	20.00	400.00
4	256519	Lang, Tony	Y-771	Spacer	5	2.00	10.00
5	259552	Tuner, Agnes	U-734	Bushing	5	25.00	120.00
6	175995	Hanz, James	EA-74	Seal	1	3.00	3.00
7	267991	Swindle, Joe	EN-12	Rebuilt Engine	1	1,220.00	1,220.00

Original Test AR Master File

CUST NUM	CUSTOMER NAME	CUSTOMER ADDRESS	CREDIT LIMIT	CURRENT BALANCE
231893	Smith, Joe	1520 S. Maple, City	1,000.00	400.00
256519	Lang, Tony	18 Etwine St., City	5,000.00	850.00
267991	Swindle, Joe	1 Shady Side, City	3,000.00	2,900.00

Updated Test AR Master File

CUST NUM	CUSTOMER NAME	CUSTOMER ADDRESS	CREDIT LIMIT	CURRENT BALANCE
231893	Smith, Joe	1520 S. Maple, City	1,000.00	420.00
256519	Lang, Tony	18 Etwine St., City	5,000.00	860.00
267991	Swindle, Joe	1 Shady Side, City	3,000.00	2,900.00

Error Report

REC NUM	CUST NUM	CUSTOMER NAME	PART NUM	DESCRIPTION	QNTY	UNIT PRICE	TOTAL PRICE	EXPLANATION OF ERROR
2	231893	Azar, Atul X	J-912	Gear	3	15.00	45.00	CUSTOMER NAME does not correspond to CUST # 231893
3	245851 X	Jones, Mary	123-LM	Hose	20	20.00	400.00	Check digit error in CUST # field
5	259552	Tuner, Agnes	U-734	Bushing	5	25.00	120.00 X	Price extension error
6	175995 X	Hanz, James	EA-74	Seal	1	3.00	3.00	Record out of sequence
7	267991	Swindle, Joe	EN-12	Rebuilt Engine	1	1,220.00 X	1,220.00 X	Credit limit error

at implementation. If the application has been modified, the auditor can create additional test data that focus on the areas of the program changes.

Base Case System Evaluation

Base case system evaluation (BCSE) is a variant of the test data approach. BCSE tests are conducted with a set of test transactions containing all possible transaction types. These are processed through repeated iterations during systems development testing until

consistent and valid results are obtained. These results are the base case. When subsequent changes to the application occur during maintenance, their effects are evaluated by comparing current results with base case results.

Tracing

Another type of the test data technique called **tracing** performs an electronic walk-through of the application's internal logic. The tracing procedure involves three steps:

1. The application under review must undergo a special compilation to activate the trace option.
2. Specific transactions or types of transactions are created as test data.
3. The test data transactions are traced through all processing stages of the program, and a listing is produced of all programmed instructions that were executed during the test.

Figure 17-13 illustrates the tracing process using a portion of the logic for a payroll application. The example shows records from two payroll files—a transaction record showing hours worked and two records from a master file showing pay rates. The trace listing at the bottom of Figure 17-13 identifies the program statements that were executed and the order of execution. Analysis of trace options indicates that Commands 0001 through 0020 were executed. At that point, the application transferred to Command 0060. This occurred because the employee number (the key) of the transaction record did not match the key of the first record in the master file. Then Commands 0010 through 0050 were executed.

FIGURE 17-13 Tracing

Payroll Transaction File

Time Card #	Employee Number	Name	Year	Pay Period	Reg Hrs	OT Hrs
8945	33456	Jones, J.J.	2007	14	40.0	3.0

Payroll Master File

Employee Number	Hourly Rate	YTD Earnings	Dependents	YTD Withhold	YTD FICA
33276	15	12,050	3	3,200	873.62
33456	15	13,100	2	3,600	949.75

Computer Program Logic

```
0001  Read Record from Transaction File
0010  Read Record from Master File
0020  If Employee Number (T) = Employee Number (M)
0030        Wage = (Reg Hrs + [OT Hrs x 1.5] ) x Hourly Rate
0040        Add Wage to YTD Earnings
0050        Go to 0001
0060  Else Go to 0010
```

Trace Listing

0001, 0010, 0020, 0060, 0010, 0020, 0030, 0040, 0050

Advantages of Test Data Techniques

Test data techniques have three primary advantages. First, they employ through-the-computer testing, thus providing the auditor with explicit evidence concerning application functions. Second, if properly planned, test data runs can be employed with only minimal disruption to the organization's operations. Third, they require only minimal computer expertise on the part of auditors.

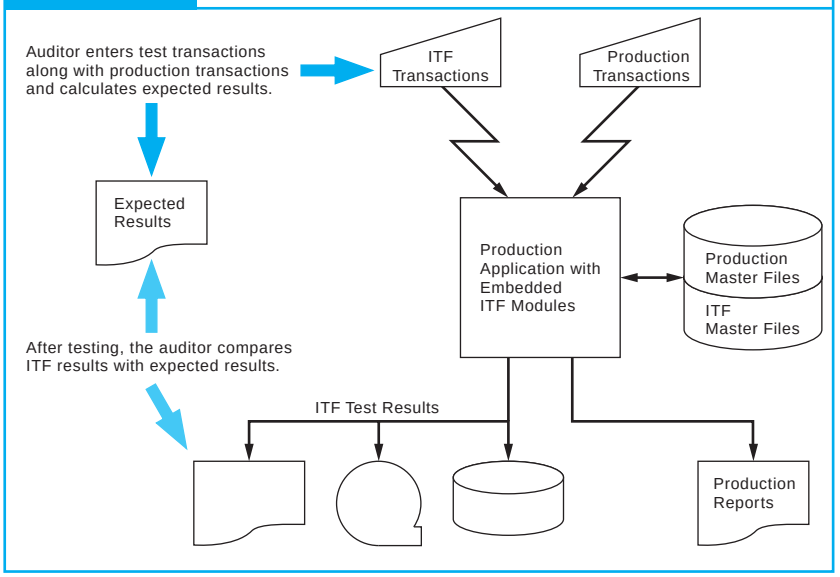
Disadvantages of Test Data Techniques

The primary disadvantage of test data techniques is that auditors rely on the client's IT personnel to obtain a copy of the production application under test. The **audit risk** here is that the IT personnel may intentionally or accidentally provide the auditor with the wrong version of the application. Audit evidence collected independently is more reliable than evidence the client supplies. A second disadvantage is that these techniques produce a static picture of application integrity at a single point in time. They do not provide a convenient means for gathering evidence of ongoing application functionality. High cost of implementation is a third disadvantage of test data techniques. The auditor must devote considerable time to understanding program logic and creating test data. The following section shows how automating testing techniques can resolve these problems.

The Integrated Test Facility

The **integrated test facility (ITF)** approach is an automated technique that enables the auditor to test an application's logic and controls during its normal operation. The ITF involves one or more audit modules designed into the application during the systems development process. In addition, ITF databases contain dummy or test master file records integrated among legitimate records. Some firms create a dummy company to which test transactions are posted. During normal operations, test transactions are merged into the input stream of regular (production) transactions and are processed against the files of the dummy company. Figure 17-14 illustrates the ITF concept.

FIGURE 17-14 The ITF Technique



ITF audit modules are designed to discriminate between ITF transactions and production data. This may be accomplished in a number of ways. One of the simplest and most commonly used is to assign a unique range of key values exclusively to ITF transactions. For example, in a sales order processing system, account numbers between 2000 and 2100 are reserved for ITF transactions and will not be assigned to actual customer accounts. By segregating ITF transactions from legitimate transactions in this way, ITF test data does not corrupt routine reports that the application produces. Test results are produced separately in digital or hard-copy form and distributed directly to the auditor. Just as with the test data techniques, the auditor analyzes ITF results against expected results.

Advantages of ITF

The ITF technique has two advantages over test data techniques. First, ITF supports ongoing monitoring of controls as COSO recommends. Second, ITF-enhanced applications can be economically tested without disrupting the user's operations and without the intervention of computer services personnel. Thus, ITF improves the efficiency of the audit and increases the reliability of the audit evidence gathered.

Disadvantages of ITF

The primary disadvantage of ITF is the potential for corrupting data files with test data that may end up in the financial reporting process. Steps must be taken to ensure that ITF test transactions do not materially affect financial statements by being improperly aggregated with legitimate transactions. This problem can be remedied in two ways: (1) adjusting entries may be processed to remove the effects of ITF from general ledger account balances or (2) data files can be scanned by special software that remove the ITF transactions.

Parallel Simulation

Parallel simulation involves creating a program that simulates key features or processes of the application under review. The simulated application is then used to reprocess transactions that the production application previously processed. This technique is illustrated in Figure 17-15. The results obtained from the simulation are reconciled with the results of the original production run to determine if application processes and controls are functioning correctly.

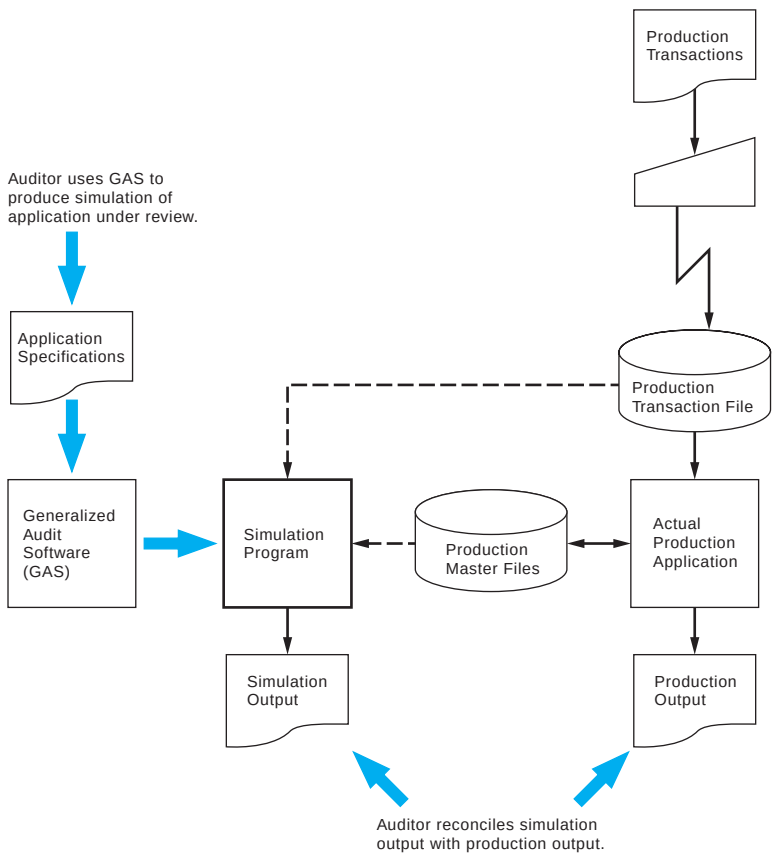
Creating a Simulation Program

Simulation packages are commercially available and are sometimes a feature of **generalized audit software (GAS).**[2] The steps involved in performing parallel simulation testing are outlined in the following section.

1. The auditor must first gain a thorough understanding of the application under review. Complete and current documentation of the application is required to construct an accurate simulation.
2. The auditor must then identify those processes and controls in the application that are critical to the audit. These are the processes to be simulated.
3. The auditor creates the simulation using a fourth-generation language or generalized audit software.

2 Although GAS can be used for testing internal controls, it is primarily a substantive testing technique. For this reason, this technology is discussed in the section that deals with substantive testing.

FIGURE 17-15 Parallel Simulation Technique



4. The auditor runs the simulation program using selected production transactions and master files to produce a set of results.
5. Finally, the auditor evaluates and reconciles the test results with the production results produced in a previous run.

Simulation programs are usually less complex than the production applications they represent. Because simulations contain only the application processes, calculations, and controls relevant to specific audit objectives, the auditor must carefully evaluate differences between test results and production results. Differences in output results occur for two reasons: (1) the inherent crudeness of the simulation program and (2) real deficiencies in the application's processes or controls, which the simulation program makes apparent.

Substantive Testing Techniques

Substantive tests are so named because they are used to substantiate dollar amounts in account balances. Substantive tests include but are not limited to the following:

1. Determining the correct value of inventory.
2. Determining the accuracy of prepayments and accruals.

3. Confirming accounts receivable with customers.
4. Searching for unrecorded liabilities.

Before substantive tests can be performed, these data must first be extracted from their host media and presented to the auditor in usable form. The two CAATTs examined in this section assist the auditor in selecting, accessing, and organizing data used for performing substantive tests.

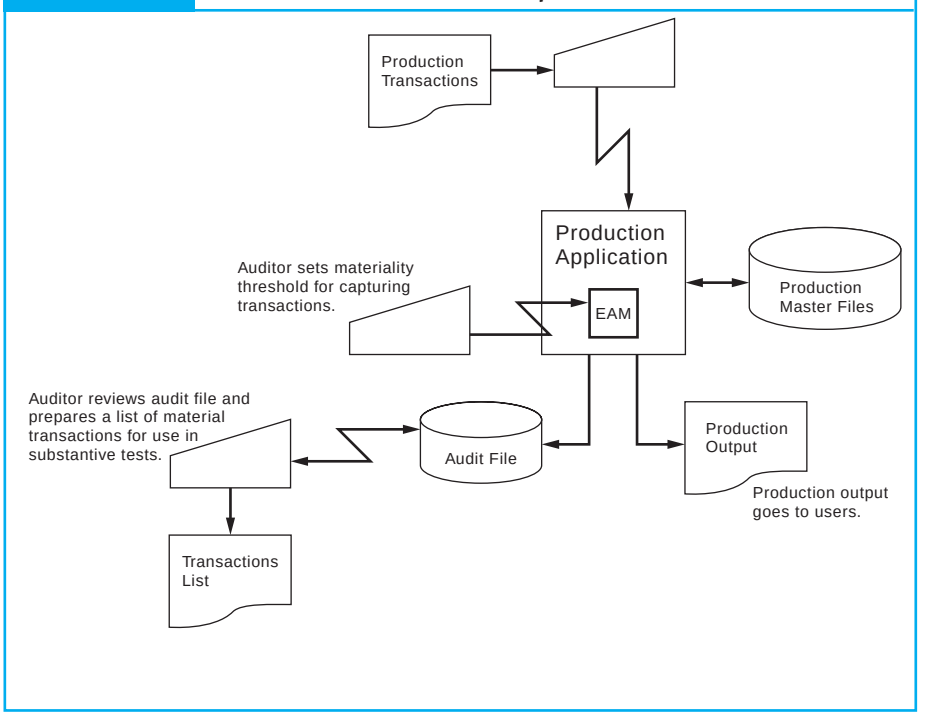
The Embedded Audit Module

Embedded audit module (EAM) techniques use one or more programmed modules embedded in a host application to select, for subsequent analysis, transactions that meet predetermined conditions. This approach is illustrated in Figure 17-16.

As the host application processes the selected transaction, a copy of it is stored on an audit file for subsequent review. The EAM approach allows material transactions to be captured throughout the audit period. The auditor retrieves captured transactions at period-end or at any time during the period, thus significantly reducing the amount of work the auditor must do to identify significant transactions for substantive testing.

To begin data capturing, the auditor specifies to the EAM the parameters and materiality threshold of the transactions set to be captured. For example, assume that the auditor establishes a $50,000 materiality threshold for transactions that a sales order processing system has processed. Transactions equal to or greater than $50,000 will be copied to the audit file. From this set of transactions, the auditor will select a subset to be used for substantive tests. The EAM will ignore transactions that fall below this threshold.

FIGURE 17-16 Embedded Audit Module Technique



Though primarily a substantive testing technique, EAMs may also be used to monitor application controls on an ongoing basis as recommended in the COSO framework. For example, transactions the EAM selects can be reviewed for proper authorization, completeness and accuracy of processing, and correct posting to accounts.

Disadvantages of EAMs

The EAM approach has two significant disadvantages. The first pertains to operational efficiency and the second to EAM integrity.

Operational Efficiency. From the user's point of view, EAMs decrease operational performance. The presence of an audit module within the host application may create significant overhead, particularly when the level of testing is high. One approach for relieving this burden from the system is to design modules that the auditor may turn on and off. Doing so will, of course, reduce the effectiveness of the EAM as an ongoing audit tool.

Verifying EAM Integrity. The EAM approach may not be a viable audit technique in environments with a high level of program maintenance. When host applications are undergoing frequent changes, the EAMs embedded within the hosts will also require frequent modifications. The integrity concerns raised earlier regarding application maintenance apply equally to EAMs. The integrity of EAM directly affects the quality of the audit process. Auditors must therefore evaluate the EAM integrity. This would be accomplished in the same way as testing the host application controls.

Generalized Audit Software (GAS)

GAS is the most widely used CAATT for IS auditing. GAS allows auditors to access electronically coded data files and perform various operations on their contents. ACL and IDEA are currently the leading products, but others exist with similar features. The following audit tasks can be performed using GAS:

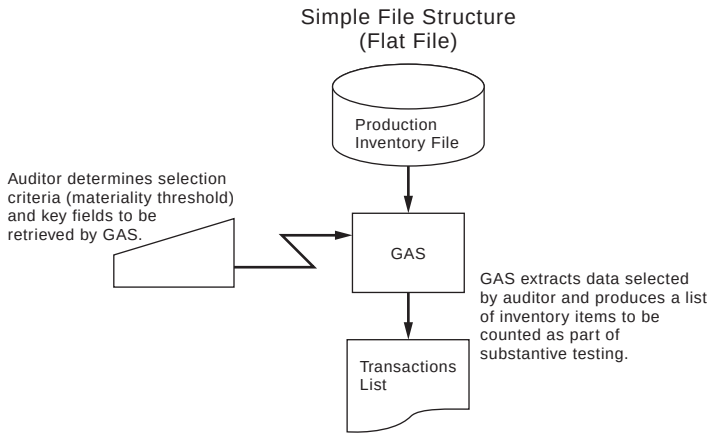
1. Footing and balancing entire files or selected data items.
2. Selecting and reporting detailed data contained on files.
3. Selecting stratified statistical samples from data files.
4. Formatting results of tests into reports.
5. Printing confirmations in either standardized or special wording.
6. Screening data and selectively including or excluding items.
7. Comparing two files and identifying any differences.
8. Recalculating data fields.

The widespread popularity of GAS is due to four factors: (1) GAS languages are easy to use and require little IT background on the part of the auditor, (2) GAS may be used on any type of computer because it is hardware independent, (3) auditors can perform their tests on data independent of client IT professional, and (4) GAS can be used to audit the data files of many different applications (in contrast with EAMs, which are application specific).

Using GAS to Access Simple Structures

Accessing flat-file structures (such as a text file) is a simple process, as illustrated in Figure 17-17. In this example, an inventory file is read directly into the GAS,

FIGURE 17-17 Using GAS to Access Simple File Structure



which extracts key information needed for the audit, including the quantity on hand, the dollar value, and the warehouse location of each inventory item. The auditor's task is to perform a physical count of a representative sample of the inventory on hand to verify the existence and value of the inventory. Thus, on the basis of a materiality threshold that the auditor provides, the GAS selects the sample records and prepares a report with the key information.

Using GAS to Access Complex Structures

Gaining access to complex structures, such as VSAM files and object-oriented database files, poses more of a problem for the auditor. Most DBMSs, however, have utility features that will reformat complex structures into flat files. In such cases, rather than accessing the complex structure directly, an intermediate flat file is produced, which the GAS then accesses. Figure 17-18 shows this technique.

To illustrate the file-flattening process, consider the complex database structure presented in Figure 17-19. The database structure uses pointers to integrate three related files—Customer, Sales Invoice, and Line Item—in a hierarchical model. It would be difficult, if not impossible, to extract audit evidence from a structure of this complexity using GAS. A simpler flat-file version of this structure is illustrated in Figure 17-20. The resulting single text represents the three record types as a sequential structure with variable length records that GAS can easily access.

Audit Issue Pertaining to the Creation of Flat Files

When auditors rely on client IT personnel to produce a flat file from their database, they run the risk that database integrity will be compromised. For example, if the auditor is confirming accounts receivable, certain fraudulent accounts in the original database may be intentionally omitted from the flat file provided to the auditor. Auditors skilled in relational and object database technology can avoid this problem. Not surprisingly, public accounting firms are aggressively seeking employees with strong computer skills to accompany their accounting training.

FIGURE 17-18 Using GAS to Access Complex File Structure

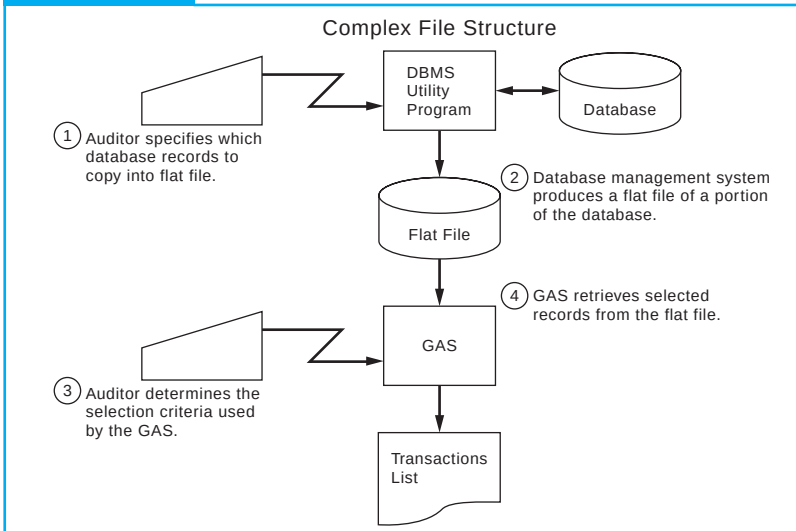


FIGURE 17-19 Complex Database Structure

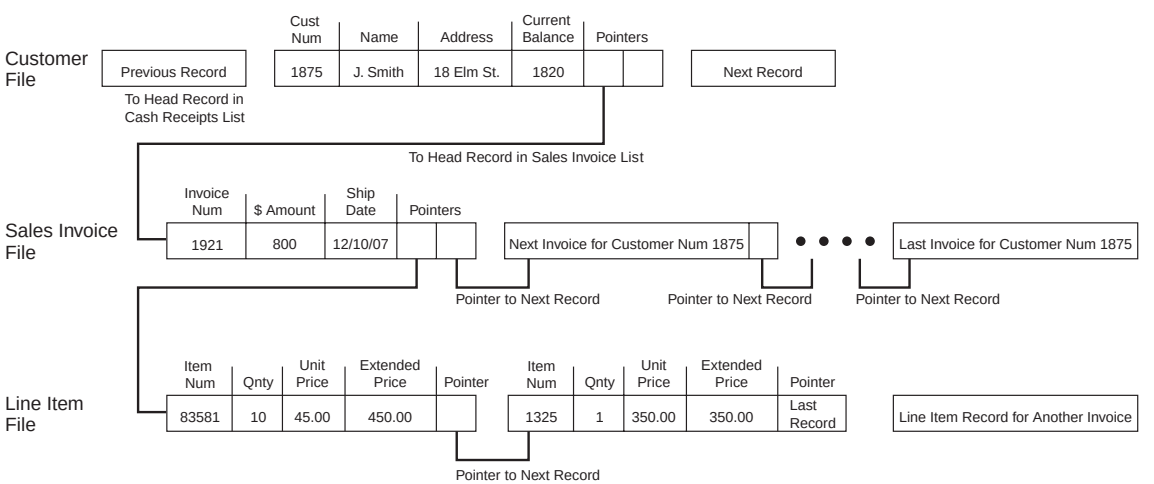
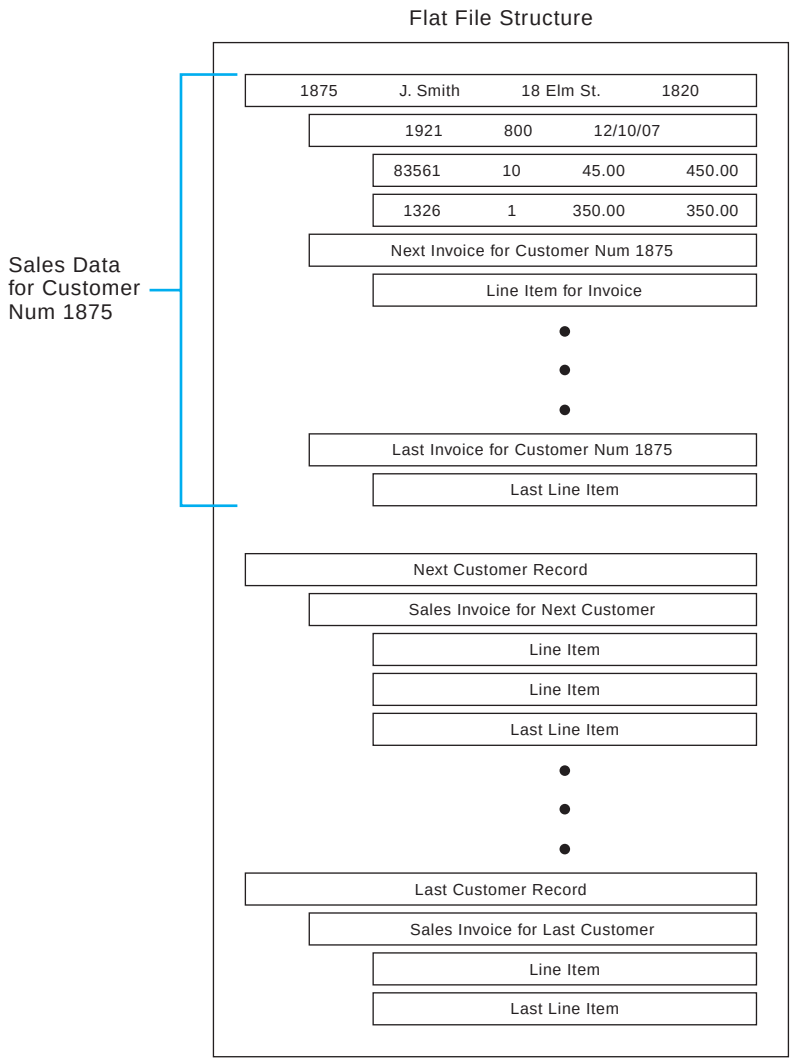


FIGURE 17-20 Flat Version of a Complex File Structure



Summary

SOX legislation requires management to design, implement, and certify controls over financial reporting. Similarly, external auditors are required to attest to management's assessment of controls. This chapter dealt with the business risks, IT controls, and test of controls pertaining to three areas of specific concern to SOX: systems development, program change procedures, and computer applications.

The integrity of financial data is directly dependent on the accuracy of the applications that process them. Likewise, the integrity of those applications depends on the quality of the systems development process that produced them and on the program change procedures through which they were modified. Lack of control over these areas, or inconsistency in their function, can result in unintentional application errors and program fraud.

The systems development and maintenance controls and the test of controls described in this chapter apply both to management's SOX-compliance objectives and the auditor's attest responsibility. To test specific application controls, auditors (internal and external) use several CAATT techniques, including the test data method, the integrated test facility, and parallel simulation. This chapter concluded with a discussion of two popular CAATTs (embedded audit module and generalized audit software) used for substantive testing.

Key Terms

access tests (816)
accuracy tests (816)
audit objectives (799)
audit procedures (815)
audit risk (822)
audit trail controls (811)
audit trail tests (816)
authenticity tests (816)
base case system evaluation (BCSE) (820)
batch controls (809)
check digit (807)
completeness (806)
completeness tests (816)
computer-assisted audit tools and techniques (CAATTs) (818)
embedded audit module (EAM) (825)
existence or occurrence (815)
generalized audit software (GAS) (823)
hash total (811)
integrated test facility (ITF) (822)
management assertion (815)
operating system (802)
parallel simulation (823)
presentation and disclosure (815)
redundancy tests (816)
rights and obligations (815)
rounding error tests (816)
run-to-run control (810)
salami fraud (816)
spooling (813)
substantive tests (824)
test data method (819)
tests of controls (804)
tracing (821)
transcription errors (807)
transposition errors (807)
valuation or allocation (815)

Review Questions

1. List the six systems development controls the chapter addresses. List the two systems maintenance controls.
2. Explain how program testing is conducted and the importance of test data.
3. List the control features that directly contribute to the security of the computer center environment.
4. What is the purpose of a valid vendor file?
5. What are the broad classes of input controls?
6. Give one example of an error that a check digit control detects.
7. What are the primary objectives of a batch control?
8. What are the categories of processing controls?
9. If all of the inputs have been validated before processing, then what purpose do run-to-run controls serve?
10. What is the objective of a transaction log?
11. How can spooling present an added exposure?
12. What tests may be conducted for identifying unauthorized program changes?
13. What tests may be conducted for identifying application errors?
14. What does auditing around the computer mean versus auditing through the computer? Why is this so important?
15. What are some white box tests?
16. What is an embedded audit module?
17. Explain what GAS is and why it is so popular with larger public accounting firms. Discuss the independence issue related to GAS.
18. What is the purpose of a limit check?
19. What is the purpose of a range check?
20. What is a reasonableness test?
21. What is a validity check?
22. What is a run-to-run control?
23. What information would a batch control record contain?

Discussion Questions

1. Discuss how a controlled SPL environment can help to deter unauthorized changes to programs. Can the use of maintenance commands mitigate these controls?
2. What types of output would be considered extremely sensitive in a university setting? Give three examples and explain why the information would be considered sensitive. Discuss who should and should not have access to each type of information.
3. What are the classes of transcription errors?
4. What is the purpose of a check digit?
5. Does a hash total need to be based on a financial data field? Explain.
6. Discuss the three common methods of handling errors in transaction files.
7. Why is computer waste disposal a potential internal control issue?
8. Why would a systems programmer create a back door if he or she has access to the program in his or her day-to-day tasks?
9. The systems development life cycle is a methodology. Why are auditors responsible for evaluating the controls in this process?
10. What factors do you think might cause an auditing team to spend more time than average on tests to identify application errors? For unauthorized program changes?
11. Explain how an embedded audit module works.

12. Compare and contrast the following techniques based on costs and benefits:
    - test data method
    - base case system evaluation
    - tracing
    - integrated test facility
    - parallel simulation
13. What is the control issue related to reentering corrected error records into a batch processing system? What are the two methods for doing this?

Multiple-Choice Questions

1. Computer applications use routines for checking the validity and accuracy of transaction data called
   a. operating systems.
   b. edit programs.
   c. compiler programs.
   d. integrated test facilities.
   e. compatibility tests.
2. How does a direct access file processing system edit individual transactions?
   a. takes place in a separate computer run
   b. takes place in online mode as transactions are entered
   c. takes place during a backup procedure
   d. is not performed due to time constraints
   e. is not necessary
3. Which of the following is an example of an input control?
   a. making sure that output is distributed to the proper people
   b. monitoring the work of programmers
   c. collecting accurate statistics of historical transactions while gathering data
   d. recalculating an amount to ensure its accuracy
   e. having another person review the design of a business form
4. A control designed to validate a transaction at the point of data entry is
   a. recalculation of a batch total.
   b. a record count.
   c. a check digit.
   d. checkpoints.
   e. recalculation of hash total.
5. In a computer system, how are accounting records posted?
   a. master file is updated to a transaction file
   b. master file is updated to an index file
   c. transaction file is updated to a master file
   d. master file is updated to a year-to-date file
   e. current balance file is updated to an index file
6. The controls in a computerized system are classified as
   a. input, processing, and output.
   b. input, processing, output, and storage.
   c. input, processing, output, and control.
   d. input, processing, output, storage, and control.
   e. collecting, sorting, summarizing, and reporting.
7. An employee in the receiving department keyed in a shipment from a remote terminal and inadvertently omitted the purchase order number. The best systems control to detect this error would be a
   a. batch total.
   b. completeness test.
   c. sequence check.
   d. reasonableness test.
   e. compatibility test.
8. In an automated payroll processing environment, a department manager substituted the time card for a terminated employee with a time card for a fictitious employee. The fictitious employee had the same pay rate and hours worked as the terminated employee. The best control technique to detect this action using employee identification numbers would be a

a. batch total.
b. record count.
c. hash total.
d. subsequent check.
e. financial total.

9. SOX legislation calls for sound internal control practices over financial reporting and requires SEC-registered corporations to maintain systems of internal control that meet SOX standards. An integral part of internal control is the appropriate use of preventive controls. Which of the following is not an essential element of preventive control?
   a. separation of responsibilities for the recording, custodial, and authorization functions
   b. sound personnel practices
   c. documentation of policies and procedures
   d. implementation of state-of-the-art software and hardware
   e. physical protection of assets
10. Which of the following is NOT a test for identifying application errors?
    a. reconciling the source code
    b. reviewing test results
    c. retesting the program
    d. testing the authority table
11. Which of the following is NOT a common type of white box test of controls?
    a. completeness tests
    b. redundancy tests
    c. inference tests
    d. authenticity tests
12. An electronic walk-through of the application's internal logic is called
    a. a salami logic test.
    b. an integrated test.
    c. tracing.
    d. a logic bomb test.

Problems

1. Input Validation

Describe the types of application control used for the following data in a payroll system.

a. Employee name
b. Employee number
c. Social Security number
d. Rate per hour or salary
e. Marital status
f. Number of dependents
g. Cost center
h. Regular hours worked
i. Overtime hours worked
j. Total employees this payroll period

2. Computer Fraud and Controls

Although the threat to security via external penetration is often seen as the greatest threat, many threats are internal. Computer frauds include (1) input manipulation, (2) program alteration, (3) file alteration, (4) data theft, and (5) sabotage.

Required:

Explain how each of these five types of fraud is committed. Also, identify a method of protection against each without using the same protection method for more than one type of fraud. Use the following format.

Type of Fraud	Explanation	Description of Protection Methods
a.		
b.		
c.		
d.		
e.		

3. Processing Controls

A well-designed system can prevent both intentional and unintentional alteration and destruction of data. These data controls can

be classified as (1) input controls, (2) processing controls, and (3) output controls

Required:
For each of the three control categories listed, provide two specific controls and explain how each control contributes to ensuring the reliability of data. Use the following format.

Control Category	Specific Controls	Contribution to Data Reliability

4. Input Controls and Data Processing

A catalog company has hired you to computerize its sales order entry forms. Approximately 60 percent of all orders are received over the telephone, with the remainder received by either mail or fax. The company wants the phone orders to be input as they are received. The mail and fax orders can be batched together in groups of 50 and submitted for keypunching as they become ready. The following information is collected for each order:

- Customer number (if customer does not have one, one needs to be assigned)
- Customer name
- Address
- Payment method (credit card or money order)
- Credit card number and expiration date (if necessary)
- Items ordered and quantity
- Unit price

Required:
Determine control techniques to make sure that all orders are entered accurately into the system. Also, discuss any differences in control measures between the batch and the real-time processing.

5. Audit Plan

Rainbow Paint Company, a medium-sized manufacturing firm, has no internal auditing department. It recently hired a new accounting firm to perform the external audit.

Required:
Outline an audit plan to examine operating system control, program maintenance controls, and organizational system controls. Include in your plan the audit objectives, exposures, necessary controls, and test of controls. Also include any documentation the auditors should request.

6. Audit Plan

The auditors for Golden Gate Company have a gut feeling that liabilities may be unrecorded. Their initial suspicions stem from a radical decline in accrued liabilities from last year. Golden Gate's records are all computerized.

Required:
Devise a plan to search the data files to perform a substantive test for identifying unrecorded liabilities.

7. Risk Identification and Plan of Action

Two years ago, an external auditing firm supervised the programming of embedded audit modules for Previts Office Equipment Company. During the audit process this year, the external auditors requested that a transaction log of all transactions be copied to the audit file. The external auditors noticed large gaps in dates and times for transactions being copied to the audit file. When they inquired about this, they were informed that increased processing of transactions had been burdening the mainframe system and that operators frequently had to turn off the EAM to allow the processing of important transactions in a timely fashion. In addition, much maintenance had been performed during the past year on the application programs.

Required:
Outline any potential exposures and determine the courses of action the external auditors should use to proceed.

8. Risk Identification and Plan of Action

The internal auditors of Brown Electrical Company report to the controller. Due to changes made in the past year to several of the transaction processing programs, the internal auditors created a new test data set. The external auditors requested that the old data set also be run. The internal auditors embarrassingly explained that they overwrote the original test data set.

Required:

Outline any potential exposures and determine the courses of action the external auditor should take.

9. Risk Identification and Plan of Action

As the manager of the external audit team, you realize that the embedded audit module only writes material invoices to the audit file for the accounts receivable confirmation process. You are immediately concerned that the accounts receivable account may be substantially overstated this year and for the prior years in which this EAM was used.

Required:

Explain why you are concerned because all "material" invoices are candidates for confirmation by the customer. Outline a plan for determining if the accounts receivable are overstated.

10. Audit Objectives and Procedures

You are conducting substantive tests on the accounts receivable file to verify its accuracy. The file is large, and you decide to test only a sample of the records. Because of the complexity of the database structure, you cannot access the database directly. The client's systems programmer uses a utility program to write a query that produces a flat file, which he provides for testing purposes.

Required:

Discuss any concerns you would have as an auditor and any actions you would take.

11. Systems Development and Program Changes

Avatar Financials, Inc., located on Madison Avenue in New York, is a company that provides financial advice to individuals and small to mid-sized businesses. Its primary operations are in wealth management and financial advice. Each client has an account where basic personal information is stored at a server within the main office in New York City. The company also keeps the information about the amount of investment of each client on a separate server at their data center in Bethlehem, Pennsylvania. This information includes the total value of the portfolio, type of investments made, the income structure of each client, and associated tax liabilities.

Avatar decided to purchase software for asset management from specialized vendors. This software allows them to run analytics on the portfolios and run detailed simulations of market trends and is called Siman (SIMulation ANalaytics). V-Dot Solutions, another contractual company that is customizing and installing Siman, has sent a team of six systems analysts to carry out this task. They anticipate additional hardware installations to run the simulation analytics on Siman.

V-Dot's setup requires them to train two people from Avatar who will be responsible for minor issues and basic maintenance of the system. Special consultants from V-Dot will deal with major problems and issues. It takes four weeks to completely have the system operational and integrated into Avatar's existing computer system. The testing phase of the project has been readjusted to allow the two employees of Avatar to run these tests and ensure compatibility.

A year after the installation of the simulation software Siman, Avatar finds it very useful. To upgrade the systems to the next level, they decide to go to another data source company for a raw market data feed that will be used to run the simulations. However, this requires changes to the source code of Siman. Fortunately, within its analytics department that uses Siman, Avatar has two programmers

who are well versed in the programming language that Siman was written in. These programmers are able to implement the changes that will allow Siman-II to use the new data feed.

To remain competitive, Avatar has placed the programmers under a tight time constraint. To expedite the process, the documentation process is shortened with the intention that it will be looked into once the systems are running. The programmers also will be deployed back to the maintenance operations once the project is complete. The contract with Siman's original vendor, V-Dot, has expired and the company does not want to extend their maintenance services for another year. Instead it feels that these two programmers will be able to perform the same tasks for less money.

Required:

a. Discuss the major internal control issues in Avatar's systems development approach.
b. Comment on the duties the two programmers of Avatar perform. Are systems maintenance and program development extensions of the same responsibility?
c. Identify potential issues that might arise due to weak internal controls.

12. Computer-Assisted Audit Tools and Techniques (CAATTs)

Required:

a. Explain the advantages of using GAS to assist with audits and give several examples of how it may be used.
b. Describe the audit purpose facilitated and the procedural steps to be followed when using the following CAATTs.
   1. ITF
   2. EAM
   3. parallel simulation

13. Audit of Systems Development

The Balcar Company's auditors are developing an audit plan to review the company's systems development procedures. Their audit objectives are to ensure that

1. the system was judged necessary and justified at various checkpoints throughout the SDLC.
2. systems development activities are applied consistently and in accordance with management's policies to all systems development projects.
3. the system as originally implemented was free from material errors and fraud.
4. system documentation is sufficiently accurate and complete to facilitate audit and maintenance activities.

The following six controllable activities have been identified as sources of audit evidence for meeting these objectives: systems authorization; user specification; technical design; internal audit participation; program testing; and user testing and acceptance.

Required:

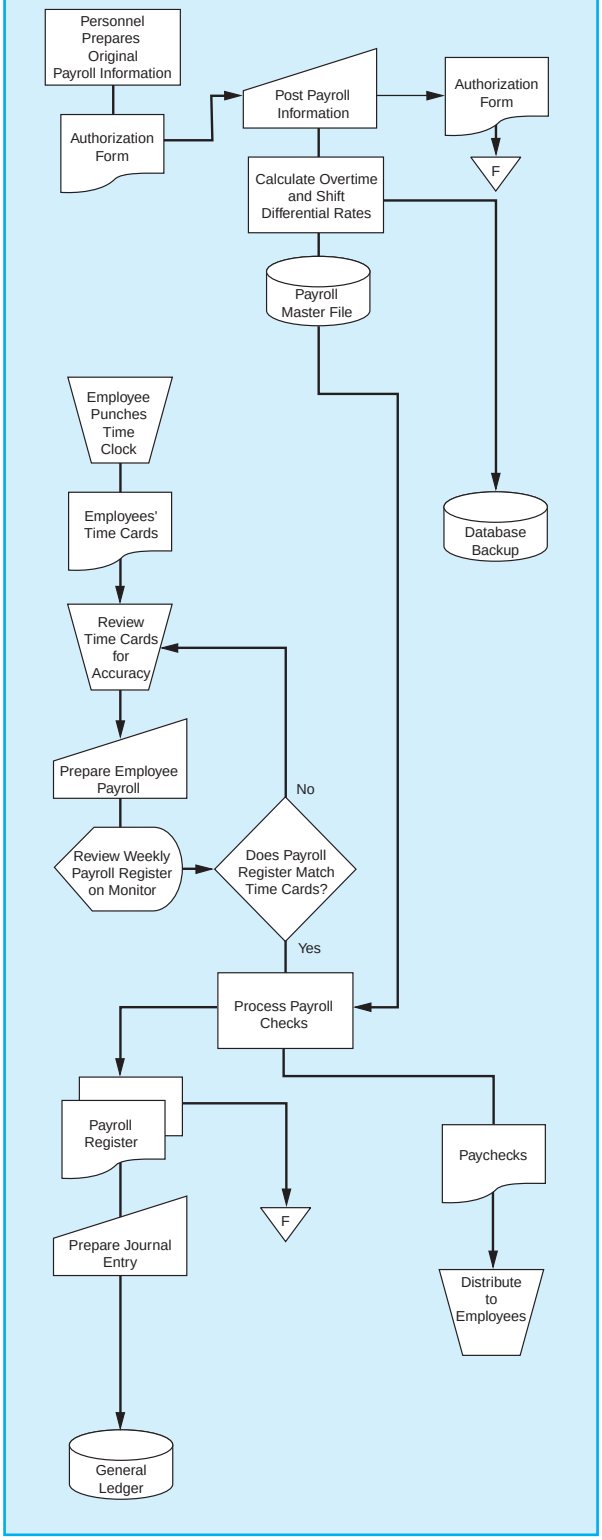
a. Explain the importance of each of the six activities in promoting effective control.
b. Outline the tests of controls that the auditor would perform in meeting audit objectives.

14. Payroll Application Control

Using this supplemental information, analyze the flowchart on the following page.

- The personnel department determines the wage rate of all employees. To start the process, personnel sends the payroll coordinator, George Jones, an authorization form in order to add an employee to the payroll. After Jones enters this information into the system, the computer automatically determines the overtime and shift differential rates for the individual, updating the payroll master files.
- Employees use a time clock to record the hours worked. Every Monday morning, George Jones collects the previous week's time cards and begins the computerized processing of payroll information to

Problem 14:
Payroll Application Control



produce paychecks the following Friday. Jones then reviews the time cards to ensure that the hours worked are correctly totaled; the system determines overtime and/or any shift differential.

- Jones performs all other processes displayed on the flowchart. The system automatically assigns a sequential number to each payroll check produced. The check stocks are stored in a box next to the computer printer to provide immediate access. After the checks are printed, an automatic check-signing machine signs them with an authorized signature plate that Jones keeps locked in a safe.
- After the check processing is completed, Jones distributes the checks to the employees, leaving the checks for the second- and third-shift employees with the appropriate shift supervisor. Jones then notifies the data processing department that he is finished with his weekly processing, and data processing makes a backup of the payroll master for storage in the computer room.

Required:

Identify and describe:

a. Areas in the payroll processing system where the internal controls are inadequate.

b. Two areas in the payroll system where the system controls are satisfactory.