BEST PRACTICE
Functional System Testing Technique Categories

Functional system testing ensures that the system requirements and specifications are achieved. The process normally involves creating test conditions for use in evaluating the correctness of the application. The types of techniques useful in performing functional testing techniques are briefly described in Table 8.

Table 8. Functional Testing Techniques

Technique	Description	Example
Requirements	System performs as specified.	Prove system requirements. Compliance to policies, regulations.
Regression	Verifies that anything unchanged still performs correctly.	Unchanged system segments function. Unchanged manual procedures correct.
Error Handling	Errors can be prevented or detected, and then corrected.	Error introduced into test. Errors re-entered.
Manual Support	The people-computer interaction works.	Manual procedures developed. People trained.
Intersystem	Data is correctly passed from system to system.	Intersystem parameters changed. Intersystem documentation updated.
Control	Controls reduce system risk to an acceptable level.	File reconciliation procedures work. Manual controls in place.
Parallel	Old system and new system are run and the results compared to detect unplanned differences.	Old and new systems can reconcile. Operational status of old system maintained.

Requirements Testing Techniques

Requirements testing must verify that the system can perform its function correctly and that the correctness can be sustained over a continuous period of time. Unless the system can function correctly over an extended period of time, management will not be able to rely upon the system. The system can be tested for correctness throughout the life cycle, but it is difficult to test the reliability until the program becomes operational.

Objectives

Successfully implementing user requirements is only one aspect of requirements testing. The responsible user is normally only one of many groups having an interest in the application system. The objectives that need to be addressed in requirements testing are:

- Implement user requirements.
- Maintain correctness over extended processing periods.
- Ensure that application processing complies with the organization's policies and procedures.

Secondary user needs have been included, such as:

- Security officer
- Database administrator
- Internal auditors
- Records retention
- Comptroller
- System processes accounting information in accordance with generally accepted accounting procedures.
- Application systems process information in accordance with governmental regulations.

How to Use Requirements Testing

Requirements' testing is primarily performed through the creation of test conditions and functional checklists. Test conditions are generalized during requirements, and become more specific as the SDLC progresses, leading to the creation of test data for use in evaluating the implemented application system.
Functional testing is more effective when the test conditions are created directly from user requirements. When test conditions are created from the system documentation, defects in that documentation will not be detected through testing. When the test conditions are created from other than the system documentation, defects introduced into the documentation will be detected.

Requirements Test Example

Typical requirement test examples include:

- Creating a test matrix to prove that the systems requirements as documented are the requirements desired by the user.
- Using a checklist prepared specifically for the application to verify the application's compliance to organizational policies and governmental regulations.

- Determining that the system meets the requirements established by the organization's department of internal auditors

When to Use Requirements Testing

Every application should be requirements tested. The process should begin in the requirements phase, and continue through every phase of the life cycle into operations and maintenance. It is not a question as to whether requirements must be tested but, rather, the extent and methods used in requirements testing.

Regression Testing Technique

One of the attributes that has plagued information technology professionals for years is the snowballing or cascading effect of making changes to an application system. One segment of the system is developed and thoroughly tested. Then a change is made to another part of the system, which has a disastrous effect on the thoroughly tested portion. Either the incorrectly implemented change causes a problem, or the change introduces new data or parameters that cause problems in a previously tested segment. Regression testing retests previously tested segments to ensure that they still function properly after a change has been made to another part of the application.

Objectives

Regression testing involves assurance that all aspects of an application system remain functional after testing. The introduction of change is the cause of problems in previously tested segments. The objectives of regression testing include:

- Determine whether systems documentation remains current.
- Determine that system test data and test conditions remain current.
- Determine that previously tested system functions perform properly after changes are introduced into the application system.

How to Use Regression Testing

Regression testing is retesting unchanged segments of the application system. It normally involves rerunning tests that have been previously executed to ensure that the same results can be achieved currently as were achieved when the segment was last tested. While the process is simple in that the test transactions have been prepared and the results known, unless the process is automated it can be a very time-consuming and tedious operation. It is also one in which the cost/benefit needs to be carefully evaluated or large amounts of effort can be expended with minimal payback.

Regression Test Example

Examples of regression testing include:

- Rerunning of previously conducted tests to ensure that the unchanged system segments function properly
- Reviewing previously prepared manual procedures to ensure that they remain correct after changes have been made to the application system
- Obtaining a printout from the data dictionary to ensure that the documentation for data elements that have been changed is correct

When to Use Regression Testing

Regression testing should be used when there is a high risk that new changes may affect unchanged areas of the application system. In the developmental process, regression testing should occur after a predetermined number of changes are incorporated into the application system. In maintenance, regression testing should be conducted if the potential loss that could occur due to affecting an unchanged portion is very high. The determination as to whether to conduct regression testing should be based upon the significance of the loss that could occur due to improperly tested applications.

Error-Handling Testing Technique

One characteristic that differentiates automated from manual systems is the predetermined error handling features. Manual systems can deal with problems as they occur, but automated systems must preprogram error-handling. In many instances the completeness of error-handling affects the usability of the application. Error-handling testing determines the ability of the application system to properly process incorrect transactions.

Objectives

Errors encompass all unexpected conditions. In some systems, approximately 50 percent of the programming effort will be devoted to handling error conditions. Specific objectives of error handling testing include:

- Determine that all reasonably expected error conditions are recognizable by the application system.
- Determine that the accountability for processing errors has been assigned and that the procedures provide a high probability that the error will be properly corrected.
- Determine that reasonable control is maintained over errors during the correction process.

How to Use Error-Handling Testing

Error-handling testing requires a group of knowledgeable people to anticipate what can go wrong with the application system. Most other forms of testing involve verifying that the application system conforms to requirements. Error-handling testing uses exactly the opposite concept.

A successful method for developing test error conditions is to assemble, for a half-day or a day, people knowledgeable in information technology, the user area, and auditing or error-tracking. These individuals are asked to brainstorm what might go wrong with the application. The totality of their thinking must then be organized by application function so that a logical set of test transactions can be created. Without this type of synergistic interaction on errors, it is difficult to develop a realistic body of problems prior to production.

Error-handling testing should test the introduction of the error, the processing of the error, the control condition, and the reentry of the condition properly corrected. This requires error-handling testing to be an iterative process in which errors are first introduced into the system, then corrected, then reentered into another iteration of the system to satisfy the complete error-handling cycle.

Error-Handling Test Examples

Error-handling requires you to think negatively, and conduct such tests as:

- Produce a representative set of transactions containing errors and enter them into the system to determine whether the application can identify the problems.
- Through iterative testing, enter errors that will result in corrections, and then reenter those transactions with errors that were not included in the original set of test transactions.
- Enter improper master data, such as prices or employee pay rates, to determine if errors that will occur repetitively are subjected to greater scrutiny than those causing single error results.

When to Use Error-Handling Testing

Error testing should occur throughout the system development life cycle. At all points in the developmental process the impact from errors should be identified and appropriate action taken to reduce those errors to an acceptable level. Error-handling testing assists in the error management process of systems development and maintenance. Some organizations use auditors, quality assurance, or professional testing personnel to evaluate error processing.

Manual Support Testing Techniques

Systems commence when transactions originate and conclude with the use of the results of processing. The manual part of the system requires the same attention to testing, as does the automated segment. Although the timing and testing methods may be different, the objectives of manual testing remain the same as testing the automated segment of the application system.

Objectives

Manual support involves all the functions performed by people in preparing data for, and using data from, automated applications. The objectives of testing the manual support systems are:

- Verify that the manual support procedures are documented and complete.
- Determine that manual support responsibility has been assigned.
- Determine that the manual support people are adequately trained.
- Determine that the manual support and the automated segment are properly interfaced.

How to Use Manual Support Testing

Manual testing involves first the evaluation of the adequacy of the process, and then, second, the execution of the process. The process itself can be evaluated in all segments of the systems development life cycle. The execution of the process can be done in conjunction with normal systems testing. Rather than prepare and enter test transactions, the system can be tested having the actual clerical and supervisory people prepare, enter, and use the results of processing from the application system.

Manual testing normally involves several iterations of the process. To test people processing requires testing the interface between people and the application system. This means entering transactions, and then getting the results back from that processing, making additional actions based on the information received, until all aspects of the manual computer interface have been adequately tested.

The manual support testing should occur without the assistance of the systems personnel. The manual support group should operate using the training and procedures provided them by the systems personnel. However, the systems personnel to determine if they have been adequately performed should evaluate the results.

Manual Support Test Example

The process of conducting manual support testing can include the following types of tests:

- Provide input personnel with the type of information they would normally receive from their customers and then have them transcribe that information and enter it into the computer.
- Output reports are prepared from the computer based on typical conditions, and the users are then asked to take the necessary action based on the information contained in computer reports.
- Users can be provided a series of test conditions and then asked to respond to those conditions. Conducted in this manner, manual support testing is like an examination in which the users are asked to obtain the answer from the procedures and manuals available to them.

When to Use Manual Support Testing

Verification that the manual systems function properly should be conducted throughout the systems development life cycle. This aspect of system testing should not be left to the latter stages of the life cycle to test. However, extensive manual support testing is best done during the installation phase so that the clerical people do not become involved with the new system until immediately prior to its entry into operation. This avoids the confusion of knowing two systems and not being certain which rules to follow. During the maintenance and operation phases, manual support testing may only involve providing people with instructions on the changes and then verifying with them through questioning that they properly understand the new procedures.

Intersystem Testing Technique

Application systems are frequently interconnected to other application systems. The interconnection may be data coming into the system from another application, leaving for another application, or both. Frequently multiple applications – sometimes called cycles or functions – are involved. For example, there is a revenue function or cycle that interconnects all of the income producing applications such as order entry, billing, receivables, shipping, and returned goods. Intersystem testing is designed to ensure that the interconnection between applications functions correctly.

Objectives

Many problems exist in intersystem testing. One is that it is difficult to find a single individual having jurisdiction over all of the systems below the level of

senior management. In addition, the process is time-consuming and costly. The objectives of intersystem testing include:

- Determine that proper parameters and data are correctly passed between applications.
- Ensure that proper coordination and timing of functions exists between the application systems.
- Determine that documentation for the involved systems is accurate and complete.

How to Use Intersystem Testing

Intersystem testing involves the operation of multiple systems in the test. Thus, the cost may be expensive, especially if the systems have to be run through several iterations. The process is not difficult; in that files or data used by multiple systems are passed from one another to verify that they are acceptable and can be processed properly. However, the problem can be magnified during maintenance when two or more of the systems are undergoing internal changes concurrently.

One of the best testing tools for intersystem testing is the integrated test facility. This permits testing to occur during a production environment and thus the coupling of systems can be tested at minimal cost.

Intersystem Test Example

Procedures used to conduct intersystem testing include:

- Developing a representative set of test transactions in one application for passage to another application for processing verification.
- Entering test transactions in a live production environment using the integrated test facility so that the test conditions can be passed from application to application to application, to verify that the processing is correct.
- Manually verifying that the documentation in the affected systems is updated based upon the new or changed parameters in the system being tested.

When to Use Intersystem Testing

Intersystem testing should be conducted whenever there is a change in parameters between application systems. The extent and type of testing will depend on the risk associated with those parameters being erroneous. If the integrated test facility concept is used, the intersystem parameters can be verified after the changed or new application is placed into production.

Control Testing Technique

Approximately one-half of the total system development effort is directly attributable to controls. Controls include data validation, file integrity, audit trail, backup and recovery, documentation, and the other aspects of systems related to integrity. Control testing techniques are designed to ensure that the mechanisms that oversee the proper functioning of an application system work.

Objectives

Control is a management tool to ensure that processing is performed in accordance with the intents of management. The objectives of control include:

- Data is accurate and complete.
- Transactions are authorized.
- An adequate audit trail of information is maintained.
- The process is efficient, effective, and economical.
- The process meets the needs of the user.

How to Use Control Testing

Control can be considered a system within a system. The term “system of internal controls” is frequently used in accounting literature to describe the totality of the mechanisms that ensure the integrity of processing. Controls are designed to reduce risks; therefore, in order to test controls the risks must be identified. The individual designing the test then creates the risk situations in order to determine whether the controls are effective in reducing them to a predetermined acceptable level of risk.

One method that can be used in testing controls is to develop a risk matrix. The matrix identifies the risks, the controls, and the segment within the application system in which the controls reside.

Control Testing Example

Control-oriented people frequently do control testing. Like error-handling, it requires a negative look at the application system to ensure that those “what-can-go-wrong” conditions are adequately protected. Error-handling is a subset of controls oriented toward the detection and correction of erroneous information. Control in the broader sense looks at the totality of the system.

Examples of testing that might be performed to verify controls include:

- Determine that there is adequate assurance that the detailed records in a file equal the control total. This is normally done by running a special program that accumulates the detail and reconciles it to the total.
- Determine that the manual controls used to ensure that computer processing is correct are in place and working.
- On a test basis, select transactions and verify that the processing for those transactions can be reconstructed.

When to Use Control Testing

Control testing should be an integral part of system testing. Controls must be viewed as a system within a system, and tested in parallel with other system tests. Knowing that approximately 50 percent of the total development effort goes into controls, a proportionate part of testing should be allocated to evaluating the adequacy of controls.

Parallel Testing Techniques

In the early days of computer systems, parallel testing was one of the more popular testing techniques. However, as systems become more integrated and complex, the difficulty in conducting parallel tests increases and thus the popularity of the technique diminishes. Parallel testing is used to determine that the results of the new application are consistent with the processing of the previous application or version of the application.

Objectives

The objectives of conducting parallel testing are:

- Conduct redundant processing to ensure that the new version or application performs correctly.
- Demonstrate consistency and inconsistency between two versions of the same application system

How to Use Parallel Testing

Parallel testing requires that the same input data be run through two versions of the same application. Parallel testing can be done with the entire application or with a segment of the application. Sometimes a particular segment, such as the day-to-day interest calculation on a savings account, is so complex and important that an effective method of testing is to run the new logic in parallel with the old logic.

If the new application changes data formats, then the input data will have to be modified before it can be run through the new application. This also makes it

difficult to automatically check the results of processing through a tape or disk file compare. The more difficulty encountered in verifying results or preparing common input, the less attractive the parallel testing technique becomes.

Parallel Test Example

Examples of the use of parallel testing include:

- Operate a new and old version of a payroll system to determine that the paychecks from both systems are reconcilable.
- Run the old version of the application system to ensure that the operational status of the old system has been maintained in the event that problems are encountered in the new application.

When to Use Parallel Testing

Parallel testing should be used when there is uncertainty regarding the correctness of processing of the new application, and the old and new versions of the application are similar. In applications like payroll, banking, and other heavily financial applications where the results of processing are similar, even though the methods may change significantly – for example, going from batch to online banking – parallel testing is one of the more effective methods of ensuring the integrity of the new application.

References

Guide – CSTE Common Body Of Knowledge, V6.1