BEST PRACTICE
Building Test Cases

Testers and users have been using test cases since the inception of computer programming. The concept of test data is a simple one – creating representative processing conditions using test cases. The complex part of creating test data is determining which processing events to make test cases. Experience shows that it is uneconomical to test all conditions in an application system. Experience further shows that most testing exercises less than one-half of the computer instructions. Therefore, optimizing testing through selecting the most important processing events is the key aspect of building test cases.

Several of the test tools are structured methods for designing test cases. For example, correctness proof, data flow analysis, and control flow analysis are all designed to develop extensive sets of test cases, as is exhaustive testing. Unfortunately, all of these tools, while extremely effective, require large amounts of time and effort to implement. Few organizations allocate sufficient budgets for this type of testing. In addition, many IT personnel are not trained in the use of these test tools.

Test cases can be built as follows:

- Built manually
- Created by a test case generator
- Extracted from a production file

Process for Building Test Cases

The recommended process for the creation and use of test cases is a nine-step process as follows:

1. **Identify test resources**. Testing using test cases can be as extensive or limited a process as desired. Unfortunately, many programmers approach the creation of test data from a “we’ll do the best job possible” perspective and then begin developing test transactions. When time expires, testing is complete. The recommended approach suggests that the amount of resources allocated for the test data test tool is determined and then a process developed that optimizes that time.

2. **Identify conditions to be tested**. A testing matrix is recommended as the basis for identifying conditions to test. As these matrices cascade through the developmental process, they identify all possible test conditions. If the

matrix concept is not used, then the possible test conditions should be identified during the use of this test tool. These should be general test conditions, such as in a payroll application to test the FICA deductions.

3. **Rank test conditions**. If resources are limited, the maximum use of those resources will be obtained by testing the most important test conditions. The objective of ranking is to identify high-priority test conditions that should be tested first. Ranking does not mean that low-ranked test conditions will not be tested. Ranking can be used for two purposes: first, to determine which conditions should be tested first; and second, and equally as important, to determine the amount of resources allocated to each of the test conditions. For example, if testing the FICA deduction was a relatively low-ranked condition, only one test transaction might be created to test that condition, while for the higher ranked test conditions several test transactions may be created.

4. **Select conditions for testing**. Based on the ranking, the conditions to be tested should be selected. At this point, the conditions should be very specific. For example, “testing FICA” is a reasonable condition to identify and rank, but for creating specific test conditions it is too general. Three test situations might be identified – such as employees whose year-to-date earnings exceed the maximum FICA deduction; an employee whose current-period earnings will exceed the difference between the year to-date earnings and the maximum deduction; and an employee whose year-to-date earnings are more than one pay period amount below the maximum FICA deductions. Each test situation should be documented in a testing matrix. This is a detailed version of the testing matrix that was started during the requirements phase.

5. **Determine correct results of processing**. The correct processing results for each test situation should be determined. A unique number should identify each test situation, and then a log made of the correct results for each test condition. If a system is available to automatically check each test situation, special forms may be needed as this information may need to be converted to machine-readable media. The correct time to determine the correct processing results is before the test transactions have been created. This step helps determine the reasonableness and usefulness of test transactions. The process can also show if there are ways to extend the effectiveness of test transactions, and whether the same condition has been tested by another transaction.

6. **Create test cases**. Each test situation needs to be converted into a format suitable for testing. In some instances, this requires the creation of a test

case and master information to be stored by the program for the purpose of processing the test case. The method of creating the machine-readable transaction will vary based on the application and the test rules available in the information systems department.

The most common methods of creating test cases include:

- Key entry
- Test data generator
- Preparation of an input form which will be given to user personnel to enter

7. **Document test conditions**. Both the test situations and the results of testing should be documented.

8. **Conduct test**. The executable system should be run, using the test conditions. Depending on the extent of the test, it can be run under a test condition or in a simulated production environment.

9. **Verify and correct**. The results of testing should be verified and any necessary corrections to the programs performed. Problems detected as a result of testing can be attributable not only to system defects, but to test data defects. The individual conducting the test should be aware of both situations.

References

Guide – CSTE Common Body Of Knowledge, V6.1