TEST CASE DESIGN SUPPORT SYSTEM

Keizo Tatsumi
Quality Assurance Department, Computer Software Development Group
FUJITSU LIMITED
JAPAN

Abstract

This paper describes a system which supports the design of software tests from external specifications which is called black-box testing. In the black-box testing, test case designs are based on the external specifications. However, not all of the factors influencing program operations are clearly described in such documents. Furthermore, designing test cases for large-scale software requires knowledge, not only of the functions which are tested, but also of the associated software and hardware. Sometimes, it causes a problem that the quality of testing heavily depends on the scope of knowledge and insight of testing personnel. The test case design support system described in this paper offers the following functions to standardize the quality of testing at higher level without depending on one's ability: 1) accumulating and utilizing the knowledge needed to design test cases 2) extracting test factors automatically by analyzing the external specifications 3) generating test cases automatically based on the concepts of the Design of Experiments.

1. INTRODUCTION

Software reliability is often expressed by comparing the number of errors detected by testing with statistical data. In this method, it is assumed that all test cases are of the standard quality. Therefore, it is a significant importance in quality assessment to design test cases of the standard quality. In tests by independent section from development section (e.g., quality assurance section), 'black-box test' method is often used and test cases are designed from external specifications. To date, several black-box oriented test case design methodologies (e.g., 'equivalence partitioning', 'boundary value analysis', 'cause-effect graphing') have been devised. These methodologies, however, do not solve all the problems encountered during black box testing. The followings are the examples of the problems which need to be solved:

- The input conditions must be extracted before equivalence partitioning and boundary value analysis can be used: these methodologies do not provide a means of extracting the input conditions from the external specifications alone.
- Cause-effect graphing requires a method to find out the relationship between causes and effects, but since there is no effective method, it can hardly be extended to large-scale software in its present form.

The growing size of software and increasing complexity of relationship between software have also caused the need to consider the followings:

- Testing large-scale software requires a wide knowledge, including component programs and hardware.
- In tests involving many inspectors, a means is essential to standardize the quality of testing conducted by each inspector, but there is a limit as to the extent to which one can share the knowledge and skill through training.

To solve these problems, in addition to the methodologies of test case design, the system which allows each individuals to share these knowledge and test cases to keep the standard quality must be constructed.

2. TEST CASE DESIGN PROCEDURES

Test case design follows these steps:

Step-1) Division of functions.
The functions are divided into smaller logical segments to create an outline for the test case design operations to follow. This procedure is called test factor classification.

Step-2) Identification of test factors.
The input conditions and their values are analyzed based on the external specifications of each segment of the test functions. This procedure is called test factor analysis. In this paper, input conditions are called factors and their possible values are called states. The factors and their states are collectively called test factors. Test factor analysis is conducted by entering the test factors identified from the external specifications into a test factor table.

Step-3) Generation of test cases.
Test cases are generated by combining the states of the factors in the test factor table. This procedure is called test case generation. It is done by filling in a test case table.

Step-4) Definition of test results.
Expected results of each generated test cases are entered in a test results description table.

Figure 1 shows the example of test case design from external specifications of a command 'ASCFILE'. Test Case Design Support System helps the test factor analysis process and the test case generation process which are main processes of test case design procedures.
3. PROBLEMS IN TEST CASE DESIGN

Figure 2 gives the reasons for failure to detect errors in products prior to their shipment. The omission of test cases during test factor analysis is often associated with problems relating to the scope of knowledge and insight of the testing personnel, such as the following:

- Inadequate consideration of the internal operation factors not visible from the external specifications.
- Inadequate consideration of the operation factors of other products related to the products under test.

The omission of test cases during test case generation is often caused by omissions or careless errors during the process in which many test factors are implemented. Two of these factors are as follows:

- Omission of test factor combinations.
- Failure to include analyzed test factors as test cases.

Since main task of test case generation is to combine test factors, one can perform this task mechanically to a certain extent. On the other hand, test factor analysis needs experience. Suppose that the quality of test factor analysis is expressed by the number of factors and states extracted, that is, the more test factors are extracted, the better the quality of test factor analysis will be. Figure 3 represents the relation between employed years and the number of test factors which can be extracted from the same external specifications of IT testing personnel. This figure shows that the number of test factors increases in proportion to employed years up to 6 years and stays constant after that. Therefore, to improve the quality of test factor analysis as a whole, it is a significant importance to transfer knowledge to less experienced testing personnel.

4. TEST FACTOR ANALYSIS

4.1 Test Factor Analysis Process

The test factor analysis process passes through the following phases:

1) Factor Analysis
A factor analysis is made by extracting input factors and environmental factors from the external specifications.

a) Input factors
Most of the functions of a program run according to input explicitly supplied from outside in an instruction format, such as a command, control statement or statement. With commands, the input information is supplied in the command itself or by its operands.

b) Environmental factors
Program operations vary depending on the operating status of associated programs or hardware. These factors relating to associated programs and hardware are called environmental factors.
2) Analogy of Associated Factors

In factor analysis, if, for example, the factor 'file name' is extracted, possible associated factors, such as 'file organization' and 'block size', are analogously extracted from the word 'file' and selected or discarded as appropriate. The factor analysis often begins by extracting input factors from external specifications and proceeds to identify environmental factors not explicitly covered in the external specifications; that is, the process works by gaining a clue from one keyword (i.e., factor) to the next keyword.

3) State Analysis

States can be analyzed in the following categories according to their type of values:

a) Factors indicating numeric values

These factors can assume the states of the values resulting from equivalence partitioning or boundary value analysis, as in the format 'maximum value, minimum value, intermediate value, zero...'.

b) Factors specifying a selection format

With 'access-name = device-name model-name DA', for example, these factors can assume all specifications, in this case 'device-name, model-name, DA, default, invalid specification', as their states.

c) Factors in generic name format

For example, the factor 'direct access storage device (DASD)' can assume all types of DASD as its state. The generic name format can be considered a selection format, but since a choice is already made, selectable options are often not expressed in the external specifications.

Thus, state analyses can be broadly divided into logical analyses as in a) and analyses that enumerate established facts as in b) and c).

4) Test Factor Analysis Methods

If the test factor analysis processes described above are organized into a map of test factor analysis methods, the result is shown below. [2]

4.2 Test Factor Analysis Support

A scheme permitting automated application of test factor analysis methods and accumulation of the factors and states as their entities is essential to support test factor analysis.

1) Accumulation of Test Knowledge

The knowledge needed to conduct test factor analyses (called test knowledge) includes the test factor analysis methods and the factors and states that serve as their entities. Test knowledge can be divided into simple knowledge and associative knowledge as shown in Figure 4.

2) Test Knowledge Accumulates in databases and utilized in the course of test factor analysis.

a) Accumulation of simple knowledge

The states obtained by performing logical and factual state analysis are accumulated in a database keyed by factor name. This database is called the test factor database. Terms mentioned in the external specifications must be used as factor names to ensure that either different factors with identical names or identical factors with different names are stored.

b) Accumulation of associative knowledge

Associative knowledge is a set of simple knowledge which is associated. The set is defined by assigning a supervisory name (called factor group name) to a group of associated factors. This information, too, is accumulated in a test factor database. Figure 5 illustrates stored associative knowledge.

---

Figure 4. Test knowledge

---

Figure 5. Associative Knowledge Storage
facility, test knowledge items are searched for during the creation of test factor tables in the following ways:

a) Simple knowledge search
The test factor database is searched using characters entered in the factor column as keys and the states accumulated in the database are displayed.

b) Associative Knowledge Search
Associative knowledge is searched for in the same sequence as simple knowledge, and associated test factors are displayed. Figure 6 shows this process.

Figure 6. Associative Knowledge Search

3) Automatic Analysis of External Specifications
In the external specifications, syntax of commands or control statements is described by using the following notations:

a) Terminal symbols
Terminal symbols indicate that user must enter literally as described in the external specifications. With ASCFILE command shown in Figure 1, terminal symbols are ACNAME, DSNAME, DUMMY, * , .., (, and ).

b) Non-terminal symbols
Non-terminal symbols indicate that user must replace with actual value. With ASCFILE command, non-terminal symbols are access-name and file-name.

c) Meta symbols
Meta symbols represent the conditions when terminal symbols or non-terminal symbols are specified. The followings are meta symbols:
- Brackets [ ] indicate the enclosed item can be furnished or omitted. When more than one item is enclosed, only one of them can be selected or all of them can be omitted.
- Braces { } indicate that one of two or more items enclosed should be selected.
- Repetition symbol ... indicates that the immediately preceding item can be specified any number of times in this context.
- Underline _ indicates the default value.
- Selection symbol | separates selection items.

Extraction of input factors starts by analyzing the external specifications according to above notations. To support these analysis, this system generates test factor table from external specifications automatically according to conversion rules which are established between syntax notation and test factors to be extracted. During conversion process, operands of command or control statement are separated into terminal symbols and non-terminal symbols. Non-terminal symbols are used as key (i.e. test knowledge name) for associated test factors. Figure 7 shows the example of test factor table generation from external specifications.

Figure 7. Generation of Test Factor Table

5. TEST CASE GENERATION

Test cases can be generated by combining the states of appropriate factors. One known method of accomplishing such combination is cause-effect graphing, but in actual application, this requires so much knowledge and labor to find out the relationship between causes and effects that it is not suitable as a tool for testing large-scale software. To overcome this drawback, this system has chosen to combine the factors with regard only to the input conditions (that is, causes) while taking advantage of cause-effect graphing concepts, such as Boolean graphs and constraints, with the method of their combination being derived from the table of orthogonal array used in the Design of Experiments.

5.1 Standard of Test Case Generation

One technique used in the Design of Experiments is generating experimental cases with a table of orthogonal array. The table of orthogonal arrays have a characteristic that the same number of combinations of all states exists between any two factors. Thus, the following standard of test case generation with the table of orthogonal arrays can be established:

- Standard-1: All combinations of states must always exist the same number of times between any two factors.

In the experimental field, it is essential that all combinations of states always exist the same number of times between any two factors to minimize the effects of measurement errors and other adverse conditions. Since
in software tests, all combinations of states need not exist the same number of times. standard-I may be relaxed to read as follows:

- Standard-2: Each combination of states must always exist at least once between any two factors.[3]

Table 1 presents a table of orthogonal array modified to meet standard-2. This table is called a combination table.[3]

Table 1. Combination Table

| Factor | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | ...
|--------|---|---|---|---|---|---|---|---|---|---
| T1     | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1
| T2     | 2 | 1 | 2 | 2 | 1 | 2 | 1 | 2 | 1 | 2
| T3     | 2 | 1 | 2 | 2 | 1 | 2 | 1 | 2 | 1 | 2
| T4     | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1
| T5     | 2 | 1 | 2 | 2 | 1 | 2 | 1 | 2 | 1 | 2
| T6     | 2 | 1 | 2 | 2 | 1 | 2 | 1 | 2 | 1 | 2

Note: The digits (1 or 2) in the table correspond to the number of states.

5.2 Application of the Combination Table to Test Factor Tables

The combination table thus completed is applicable only to factors with two states. This implies that the test factor tables must have been standardized to have two states for each their factors. Generally, meeting this requirement is impractical, for even factors indicating numeric values can have more than two states, including a maximum, minimum, intermediate, greater than maximum, and less than minimum, and also factors seldom have a uniform number of states. It is hence necessary to process the test factor tables to meet this requirement. To this end, a Boolean graph is created as shown in Figure 8, in which states and results are connected as input and output, respectively, using a logical instruction (each node always having two inputs), so that the combination table can be applied at each node [3]. Figure 9 shows the process of creating a test case table from a test factor table. Furthermore, not all of the combinations of the factors and states entered in the test factor table actually exist. For example, a state may have an exclusive relation: that is, when it is selected, another factor or state cannot be selected. Conditions indicating such relations between factors and states are called constraints, as in cause-effect graphing. Those conditions deserve special consideration in generating test cases.

Constraints are treated in the following ways:

- Exclusive:
  Test cases including a specified invalid combination of states are regarded as an error.

- Inclusive:
  Test cases not including a specified required combination of states are regarded as an error.

- One and only one:
  Test cases including two or more of the specified states are regarded as an error. (This means that test cases are normal when they include none or one of the specified states.)

Figure 9. Application of the Combination Table

5.3 Effects of Application of the Design of Experiments

Figure 10 compares the design of experiments and the conventional technique in terms of the number of test cases involved and that of errors detected. There are no significant difference in program A and B, but the design of experiments has raised problems, such as significantly fewer test cases (program C) and fewer errors detected for the number of test cases (program D and E).

Figure 10. Effects of Application of the Design of Experiments
6. TEST CASE DESIGN SUPPORT SYSTEM

The preceding discussions have stated that the test case design support system assists the test factor analysis process by accumulating and utilizing the test knowledge and the test case generation process by application of the combination table. Furthermore, this system has the following facilities for enhancing its practicability:

- Test specifications (test factor table and test case table) editing functions
  Editing commands of this system enables to copy, move, insert and delete the factors, states and test cases on a display screen.
- Test specifications management functions
  In tests of large-scale software, many test specifications are made. This system manages test specifications by accumulating them according to their test classification codes and by recording their update history.

Figure 11 shows an overall picture of the test case design support system.

7. CONCLUSIONS

As the tested software grows in size, it creates problems that cannot be resolved by a mere application of methodology. The test knowledge concepts of the test case design support system presented in this paper are the important key to these problems. But the test knowledge process is still primitive and the following problems are needed to be solved to enhance the effectiveness of this system.

- Automatic accumulation of test knowledge
  The validity of this system is affected by the amount of accumulated test knowledge. Automatic accumulation of test knowledge is necessary.
- Selective usage of test knowledge
  In the current system, test knowledge is selected by test factor names. But this is too broad and sometimes unnecessary entries were selected. The method for accumulating and selecting valid data is necessary.

REFERENCES