IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Volume-2, Issue-4 E-ISSN: 2347-2693

Test Case Design for Critical Systems using Test Matrix and Truth Table

Himanshu Joshi[1*], Ravi Leelu Chowdary[2] and Hearsh Varma[3]

[1*,2] *Departmet of CSE, JNTUH-CEH, India,*
[2] *Quality analyst team, S&P Capital IQ, India*

www.ijcseonline.org

Received: 7 /04/2014 Revised: 12/04/2014 Accepted: 20/04/2014 Published: 30/04/2014

Abstract— Testing is done to find out any errors in the applications and to ensure that they are fit for use. Ordinarily, teams put in their best efforts to find and fix as many bugs as possible. Sometimes due to factors such as lack of exhaustive test cases & build deadlines extensive testing is not done. Also, missing test cases in terms of complex systems due to human errors is very much possible. The post production errors are not catastrophic when the applications are meant for non-critical purposes. But, in life critical applications such as aerospace & medicine, fully comprehensive testing needs to be performed. The success of stopping a bug leakage in release phase depends considerably on the test cases used to perform the testing. Effective set of test cases should be designed to enable detection of maximum number of errors. This paper proposes Test Matrix technique & Truth Table techniques as profound testing mechanisms for complex test flows and inputs.

Index Term— Testing, Test Matrix, Truth Table

I. INTRODUCTION

In general, test activities contain the below phases which are depicted in Figure 1 [1].

- Test planning
- Test case design
- Test execution
- Test reporting

The test case design plays the most significant role for improving the quality of product and testing, test case design techniques when not designed properly, lead to poor testing [2]. This is so because it defines the input values, execution procedure, expected output, pre-conditions and post-conditions. Test cases bring in some standardization, minimize the ad-hoc approach in testing and validate the testing coverage of the application [3].

Effective test cases are written using experience and in-depth analysis of the application. Test cases are drawn from the available specifications and later on modified as per actual functional & structural flow of the system. To derive the test cases there are so many techniques like boundary value analysis, equivalence partitioning, etc [4]. These techniques are helpful only to derive critical test inputs, but these techniques will not solve the purpose of achieving the effective test design.

The above mentioned approach is reactive which makes it non suitable for life critical/ safety critical applications. Many formal methods have been devised for testing critical applications. IBM CICS & Inmos/Oxford T800 Transputer Floating-Point Unit Projects are notable ones [5].
Funding done by NASA & DARPA has resulted in many tools for formal testing of complex systems.

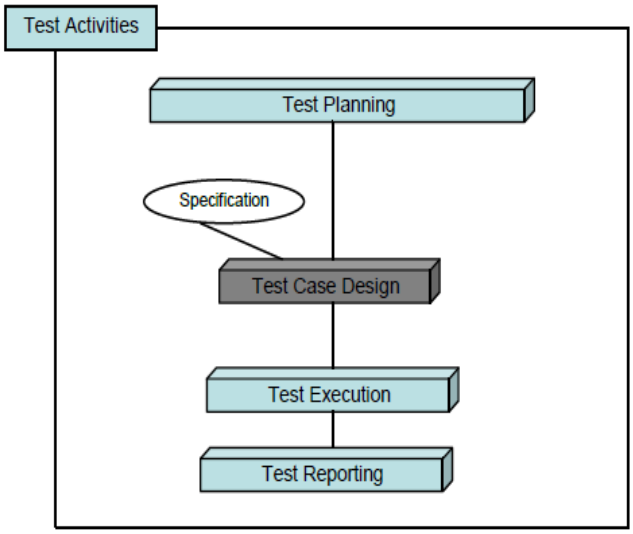


Fig. 1: Test Activities

The problem with formal methods is that they are too expensive and time consuming. Hence, this paper presents simpler and proactive approach for designing test cases for life critical applications. The following proposed techniques can be applied to formulate the effective test cases which in turn help to maximize the test coverage.

- Test Matrix Technique (identifies combinations of test flows)
- Truth Table Technique (identifies combinations of test inputs)

The rest of the paper is organized as follows. Section 2 describes the Test Matrix technique. Truth Table mechanism is discussed in section 3. Section 4 briefs the result & discussion. Section 5 presents the conclusion.

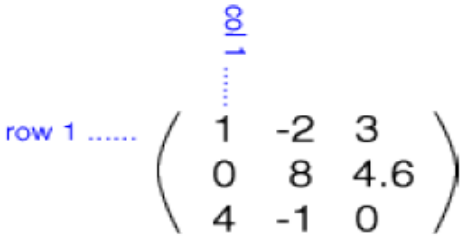
Corresponding Author: *Himanshu Joshi, hmnshjsh@gmail.com*

II. TEST MATRIX TECHNIQUE

Test Matrix Technique is a technique in which combination of test flows are plotted in m-by-n matrix structure. Hence, it is called 'Test Matrix'.

The evaluation of this test matrix comes from the basic mathematical matrix concept. In mathematics, a square (n × n) or rectangular (m × n) array of elements (numbers or algebraic variables) used to facilitate the study of problems in which the relation between the elements is important. A matrix is a collection of numbers arranged into a fixed number of rows and columns. In most of the cases, the numbers are real numbers. In general, matrices can contain complex numbers those are not present here.

Below is an example of a matrix with three rows and three columns [6].



The top row is row 1. The leftmost column is column 1. This matrix is a 3x3 matrix because it has three rows and three columns. In describing matrices, the format is:

rows X columns

Each number that makes up a matrix is called an element of the matrix. The elements in a matrix have specific locations.

The advantage of matrices is that they can be studied algebraically by assigning a single symbol to a matrix rather than considering each element separately.

For example, for the given input sets, the combinations of test flows are plotted in a matrix structure as shown in Figure 2.

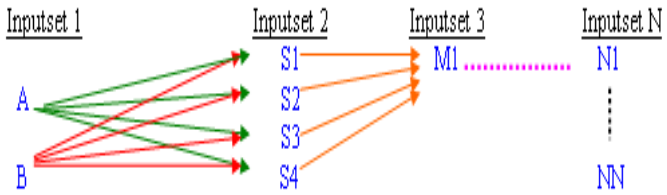


Fig. 2: Test Matrix

For given N input sets, many flows like A-S1-M1…N1, A-S2-M1…N1 and so on can be derived. Each flow may behave in a different manner. So it's important to capture all the flows. If number of input sets is very less say 2 sets, it may look simple in deriving combinations. But, when we think of huge number of input sets, this technique will stand out to be easier compared to other methods.

The relation between the input sets is clearly established, so that impact of these combinations in the output is clearly visible. In this technique, the matrix formation starts with first two input sets (set1 & 2) by relating the corresponding inputs as follows;

Considering input set 1 & 2;

$$\text{Matrix 1} = \begin{bmatrix} A - S1 & B - S1 \\ A - S2 & B - S2 \\ A - S3 & B - S3 \\ A - S4 & B - S4 \end{bmatrix}$$

Considering input set 1,2, & 3;

$$\text{Matrix 2} = \begin{bmatrix} A - S1 - M1 & B - S1 - M1 \\ A - S2 - M1 & B - S2 - M1 \\ A - S3 - M1 & B - S3 - M1 \\ A - S4 - M1 & B - S4 - M1 \end{bmatrix}$$

Considering input set 1,2,3,…& N;

$$\text{Matrix N} = \begin{bmatrix} A - S1 - M1...N1 & B - S1 - M1...N1 \\ A - S2 - M1...N1 & B - S2 - M1...N1 \\ A - S3 - M1...N1 & B - S3 - M1...N1 \\ A - S4 - M1...N1 & B - S4 - M1...N1 \end{bmatrix}$$

Once the first matrix is formed, the next matrix is formed by adding third input set. This procedure is followed till the last input set. At the end, we come out with all combinations of test flows.

III. TRUTH TABLE TECHNIQUE

In Truth Table Technique combination of test inputs are derived from boolean values as shown in figure 3 present below [7].

Input	
P	Q
0	0
0	1
1	0
1	1

Input	
P	Q
	√
√	
√	√

Fig. 3: Truth Table

A truth table is a breakdown of a logic function by listing all possible values the function can attain. It is rarely possible to guess the numerical solution to a problem, and because there are an infinite number of numbers it is obvious that one cannot try all possible solutions in order to find one that solves the problem [8]. But in logic, we only have two "numbers": True and False. Therefore, any logical statement (input set) which contains a finite number of logical variables (inputs) can be analyzed using a table which lists all possible values of the variables: a "truth table". Since each variable (input) can take only two values, a statement (input set) with "n" variables requires a table with 2^n rows.

In this technique, once the inputs are available, the combination of inputs is formed as per the below procedure.

Step 1: Collecting and organizing the inputs
The inputs are collected from the available test data and organized in such a way that the truth table can be formed (horizontal alignment of inputs-P, Q).

Step 2: Assigning boolean values
From the organized inputs, the boolean values (0 & 1) are assigned to the table. This ends with the table filled in by the combination of 0's & 1's.

Step 3: Deriving combination of inputs
Once the truth table is formed with boolean values, the combination of inputs are derived by converting 1's into valid (√) and 0's into invalid (Empty).

For the given inputs P and Q, the table yields 4 combinations of inputs ($2^2 = 4$) as shown in figure 4.

If an application has two inputs P & Q, then the test execution can be done without considering two inputs, by selecting any one of the inputs at a time, by selecting both.

Input	
P	Q
	√
√	
√	√

Fig. 4: Derived input combinations

The extension to more than two inputs should now be obvious:

1. For the two inputs, the rows are split into four sections: the first and third quarters are valid while the second and fourth quarters are invalid
2. For the three inputs, the rows are split into eighths, with alternating eighths having valid's and invalid's
3. In general, for 'n' inputs, the rows are split into 2^n parts, with alternating valid's and invalid's in each part.

IV. Result & Discussion

The quality of an application relies on the complete coverage of each flow cascaded with different equations and critical inputs. In an application involved with enormous amount of formulae/equations, complex flows with critical inputs, each flow will cascade different equations. Hence, it is vital to test each & every flow with all the critical inputs. In order to cover all the flows, the test matrix technique is applied by organizing the input sets in a matrix structure. From the final matrix, each & every flow of the application is arrived.

The truth table technique is used by organizing the inputs to arrive at the combination of inputs. These techniques will also help to filter out the critical flows & inputs. The combinations of flows & inputs are identified. Since this application involves enormous amount of formulae/equations, the equations with different parameters are formulated in MS Excel using macros to verify the result of the clustered equations of each flow.

With all this work, the effective test case design is in place. During the test execution, the result of each flow in the application is compared with excel result. In this way, it is feasible to get closer to the exhaustive testing. Based on the application needs, these techniques can be applied together or separately.

V. Conclusion

The usage of test matrix and truth table techniques will be more beneficial compared to other formal methods because of the following:

- Covers all combinations of test inputs and flows
- Enables designing an effective set of test cases which in turn enables detection of the maximum number of errors
- Makes all stakeholders of a project understanding the application flow clearly and quickly.
- Reduces the test case design cycle time to a greater extent.
- Helps a lot during regression testing to identify & execute the critical test flows/inputs.
- Acts as a knowledge base about the application.

The techniques can be refined in future to create a more generalized and standardized version.

ACKNOWLEDGMENT

Thanks to all the near ones for supporting us in completing this paper.

REFERENCES

[1] Software Testing, http://en.wikipedia.org/wiki/Software_testing, 2014

[2] Eldh S, Hansson H, Punnekkat S, "Analysis of Mistakes as a Method to Improve Test Case Design", IEEE Fourth International conference on Software Testing, Verification and Validation (ICST), E-ISBN: 978-0-7695-4342-0, Page No (70-79), March 21-25, 2011

[3] What is the procedure to write an effective test case?, http://www.bayt.com/en/specialties/q/8821/what-is-the-procedure-to-write-an-effective-test-case/, 2014

[4] Boundary Value Analysis & Equivalence Class Partitioning with Simple Example, http://www.softwaretestingclass.com/boundary-value-analysis-and-equivalence-class-partitioning-with-simple-example/, 2014

[5] Formal Methods for Life Critical Software, http://shemesh.larc.nasa.gov/fm/papers/Butler-1993-Formal-Methods-For-Life-Critical-Software.pdf, 2014

[6] Definition of Matrix, http://chortle.ccsu.edu/vectorlessons/vmch13/vmch13_2.html , 2014

[7] Truth Table, http://en.wikipedia.org/wiki/Truth_table, 2014

[8] Logical Operations & Truth Tables, http://kias.dyndns.org/comath/21.html, 2014

AUTHORS PROFILE

Himanshu Joshi is currently pursuing his Bachelor's degree at JNTU Hyderabad with majors in Computer Science. He has two research internships to his credit of which one is with Honeywell Technology Solutions Lab Pvt. Ltd., Bangalore and another with APGENCO. His domain of interests inter alia includes autonomic computing, cross platform application development and computational bio-chemistry.

Ravi Leelu Chowdary is a Quality Analyst at S&P Capital IQ Hyderabad. She graduated from GNITS with bachelors in Computer Science and Engineering. During final year of graduation she has done research on Load Performance, risk assessment. When free she works on Micro-Enterprise ideas and estimates risk for the same. She also enjoys doing mixed media art works and oil paintings which has been her hobby from past 13 years.

Hearsh Varma is currently pursuing his Bachelor's degree at JNTU Hyderabad with majors in Computer Science. He has done a research internship at Honeywell Technology Solutions Lab Pvt. Ltd., Bangalore. His domain of interests inter alia includes cloud computing, risk assessment methodologies and database management.