International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169
Volume: 3 Issue: 3 1298 - 1301

Decision-Table Based Testing

Anupama Y.K[1], B I Khodanpur[2]
[1] PG Student, [2]Prof.
CSE Dept., RNSIT, Bangalore, India.

Abstract— Software testing is a method of assessing the correctness of functions implemented in software program. There are mainly two approaches to identify test cases in software testing. They are structural testing(white-box testing) and functional testing(black box testing). In this paper Decision Table-Based Testing is carried out as a functional test design technique to determine the test cases for complex logic in software. It is ideal testing because it generates few number of effective test cases.

Keywords — *Functional Testing, Complex Logic, Decision Table*

*****

I. INTRODUCTION

Software testing[6] is one of the important stage in software development life cycle. Early testing in the software development life cycle reduces the budget of the overall project. Main purposes of considering testing are

- To prove the quality of software with respect to the operation of the product.
- To detect faults as early as possible. So, that the budget of fixing failure can be reduced.
- To make sure that the build(developed project) is behaving to the expectation of clients.

There are mainly two approaches to identify test cases in software testing. They are structural testing(white-box testing) and functional testing(black box testing). Structural testing is considered with respect to the internal structure of the function implemented. Functional testing considers only inputs and related outputs and it does not considers the internal structure of function implemented. Under the functional testing strategy there are three categories. First one is decision table based testing, second one is boundary value analysis testing[5] and third one is equivalence class of testing. Since the early 1960's, Decision tables[1][7] are used in analyzing relationships that are complex in nature[2]. It is a technique used to consider a complete set of test cases by considering the logical dependencies of inputs and related outputs. In boundary value analysis test cases generation is 5times more than the decision table based testing because it considers only the area of testing. Where as in case of equivalence class the number of test cases generation 1.5times more than that of decision table based testing because it considers data dependencies of similar inputs and outputs and then grouping them in terms of classes. Equivalence class of testing is more sophisticated method of test case development concerned with values inside the area of application(boundary value analysis testing). But the effort in generating the test cases is more in case of decision table based testing comparing to other two functional testing techniques.

Testing is a destructive process. Any test case which finds an error is a good test case. If testing is not properly performed on particular software program before the release, then the result may be disastrous. Real scenarios of software failures because of not completely testing software such as

- Therac-25 six accidents between 1985 and 1987
- The patriot missile failure in 1991
- Explosion of ariane 5 in 1996
- 2013 Launch of Obamacare**.**

Considering generation of test cases using automation testing tool[2] is likely to be speed up the testing process. So, that it helps in completing the project on time.

II. SYSTEM ARCHITECTURE

A. Structure chart

A structure chart(SC) works similar to that of divide and conquer strategy as shown in Fig 1. In that each instance is divided into sub instances or modules. The top box represents the entire instance, the bottom of the chart shows a number of boxes representing the less complicated manageable sub instances.

1.1 Get Conditions: Depending on the application tester has to enter the inputs. The inputs are number of conditions and condition cases. After completing this the next function to be processed is get actions.

1.2 Get Actions: For this module the tester considers the inputs as the number of actions and action cases. After this process the next activity to be considered is test case generation.

1.3 Generation of test case and report generation: Here the actions are verified with respect to conditions. Then, the test

 ISSN: 2321-8169
1298 - 1301

cases are generated for randomly generated values or as the values entered by the tester.

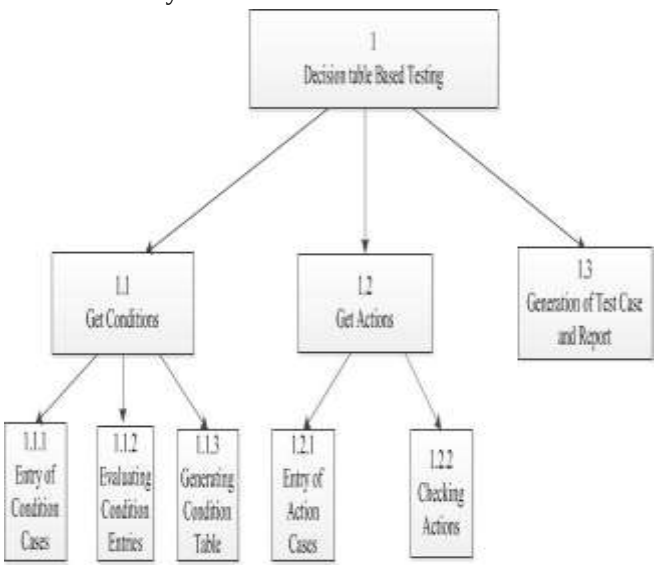

Fig 1. Structure Chart

B. Flow Chart

Flow chart helps in visualizing the internal structure of each module to be implemented. The following Fig. 2 shows the flow chart to generate test cases using decision table based testing.

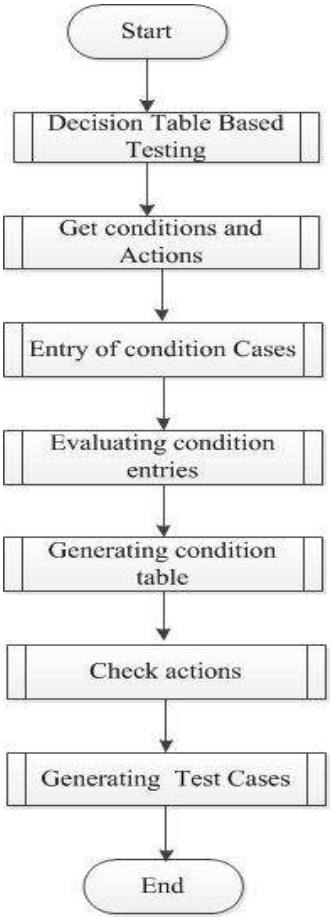

Fig 2. Flow Chart

As per the structure chart modules considered in flow chart are as follows

1. Entry of conditions
2. Evaluation of condition entries
3. Generation of condition table
4. Entry of actions
5. Generation of Test case

1. Entry of conditions

- Input: Number of variables and number of conditions.
- Output: Condition Cases.
- Description: Depending on the application, tester considers the inputs as the number of conditions and condition cases. After completing this the next function to be processed is evaluation of condition entries.

2. Evaluation of condition entries

- Input: Condition Cases.
- Output: Storing value of condition cases in a data structure.
- Description: Here evaluation of each condition for randomly generated values or as the values entered by the tester. The values obtained after the evaluation are stored in a data structure for the respective conditions. Next process to be considered is generation of condition table.

3. Generation of condition table

- Input: Number of condition cases .
- Output: Condition table.
- Description: Here generating a condition table consisting of 0's or 1's depending on the number of condition cases. After this the next function can be considered is entry of actions.

4. Entry of actions

- Input: Set of Actions.
- Output: Store them in a data structure.
- Description: Here tester has to enter all the number of actions and action cases involved in testing a program. These entries are stored in a data structure. Then the next process to be considered is generation of test case.

5. Generation of Test case

- Input: Condition table and verified actions.
- Output: Test cases.

- Description: Here the actions are verified with respect to conditions. The test cases are generated for randomly generated values or as the values entered by the tester.

III. IMPLEMENTATION

To design test cases the input and output values of a program can be taken in the form of a decision table. A decision table structure is categorized into four main units is as shown in Fig 3. First one is condition stub portion, second one is the condition entry portion, third one is the action stub portion and fourth one is the action entry portion.

.

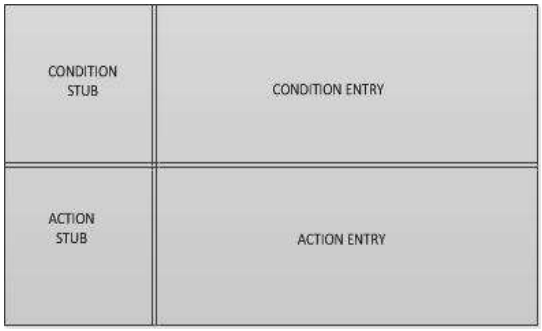

Fig 3. Decision Table Units

A package has been developed to generate the various effective test cases for the benefit of the tester. In this package depending on the application the tester is required to enter the inputs as conditions and actions. It has been tested for the some commonly occurring problems and results are documented.

Example: A decision table for finding biggest value among three variable values can be considered as shown in Table.1.

Table .1. Decision-Table for finding biggest value

Conditions	R1	R2	R3	R4	R5	R6	R7	R8
C1:a<b	1	1	1	1	0	0	0	0
C2:b<c	1	1	0	0	1	1	0	0
C3:a>c	1	0	1	0	1	0	1	0
Actions								
BIGA					X		X	
BIGB			X	X				
BIGC		X				X		
IMPOSSIBLE	X							X

Here, number of conditions as three, condition cases as a<b, b<c and a>c and action cases as BIGA, BIGB, BIGC and IMPOSSIBLE are entered as inputs by the tester. Then depending on the values generated for condition cases (a<b, b>c and a>c) condition table values will be filled. After this action cases that are possible with respect to condition table are considered as shown in the Table.1.

Let a=4, b=5 and c=9 are the values entered by tester. Then the condition cases a<b, b<c and a>c will be evaluated and respectively values 1, 1 and 0 will be generated. After this action cases will be verified with respect to the values generated in the condition table. According, action cases will be displayed as expected output. In this case it will display BIGC action case as the expected output.

IV. EXPERIMENTAL RESULTS

A. Problem Definition: Test cases for finding Biggest Value

The problem is to find the biggest of three numbers having three condition cases and three action cases. Sample Test Cases generated for finding biggest value among three variable values is as shown in Table.2.

Table.2. Test cases for finding Biggest Value

Test Case ID	Inputs*			Expected Output	Actual Output*	Remarks*
	a	b	C			
1	9	6	4	A is big(BIGA)	?	--
2	9	4	6	A is big(BIGA)	?	--
3	2	4	3	B is big(BIGB)	?	--
4	2	4	1	B is big(BIGB)	?	--
5	3	4	8	C is big(BIGC)	?	--
6	4	5	9	C is big(BIGC)	?	--

Inputs- To be set by the tester.
Actual Output- To be filled by the tester.
Remarks- If the actual output does not match with expected output.

B. Problem Definition: Test cases for Railway Reservation System

The problem is to verify whether reservation is possible or not, having four condition cases and three action cases. Sample Test Cases for Railway Reservation System generated is as shown in Table.3.

Table.3. Test cases for Railway Reservation System

Test Case ID	Inputs				Expected Output	Actual Output	Remarks
	a*	n*	s*	b*			
1	1	1	1	1	Discount Reservation	?	--
2	1	1	0	1	Reservation	?	--
3	0	1	1	1	Error-E1*	?	--
4	1	0	1	1	Error-E2*	?	--
5	1	1	1	0	Error-E3*	?	--
6	1	1	0	0	Error-E3*	?	-

a- Account exist
n- Net banking
s- Senior citizen
b- Before 15 days
E1- Reservation is not possible. (Account does not exist)
E2- Reservation is not possible. (Credit card number entered is not proper)
E3- Reservation is not possible. (Not before 15days)

C. Problem definition: Test cases for ATM Withdrawal

The problem is to verify whether withdrawal from the ATM is possible or not. Sample Test Cases generated for ATM Withdrawal are as shown in Table.4.

Table.4. Test Cases for ATM Withdrawal

Test Case ID	Inputs					Expected Output	Actual Output	Remarks
	a*	p*	w*	b*	t*			
1	1	1	500	50000	10000	Withdraw done	?	--
2	1	1	5000	2000	10000	Withdraw not possible	?	--
3	1	1	12000	2000	10000	Withdraw not possible	?	--
4	1	1	12000	2000	10000	Withdraw not possible	?	--
5	1	1	15000	20000	10000	Withdraw not possible	?	--
6	0	1	2000	5000	10000	Withdraw not possible	?	--

a- account number
p- pin number
w- withdrawal amount
b- balance amount
t- transaction limit

V. CONCLUSION

In this paper a program has been developed to generate the test cases which will help tester to check the application program for the expected results. The program has been implemented in c language using CYGWIN tool and it has been thoroughly tested for different types of application. Effective test cases are generated from this package and is likely to cut down the testing time. The complexity of the package is around 2000 lines of c code and it can be extended for more number of conditions.

REFERENCES

[1] Benjamin Moreno-Montiel and Rene McKinney-Romero, "ParalTabs: A Parallel Scheme of Decision Tables Construction", in ENC, 2013, page 47-54, IEEE Computer Society.

[2] Tomohiko Takagi, Zengo Furukawa, and Yoshinobu Machida,"Test Strategies Using Operational Profiles Based on Decision Tables", in COMPSAC, 2013, page 722-723,IEEE Computer Society.

[3] Mamatha Sharma and Subhash chandra B, "Automatic Generation of Test Suites from Decision table-theory and implementation", in ICSEA,2010,page 459-464, IEEE Computer Society.

[4] Fevzi Belli, Axel Hollmann and W.Eric Wong, "Towards Scalable Robustness Testing", in SSIRI, 2010, page 208-216, IEEE Computer Society.

[5] T.Tuglular, C.A. Muftuoglu, O. Kaya and F. Belli, and M. Linschulte, "GUI- Based Testing of Boundary overflow vulnerability", in COMPSAC, 2009, page 539-544, IEEE Computer Society.

[6] Siripol Noikajana and Taratip Suwannasart, " Web Service Test Case Generation Based on Decision Table", in QSIC,2008, page 321-326, IEEE Computer Society.

[7] Jianchao Han "Mining association Decision Rules in decision Tables through Attribute Value Reduction", in Granular Computing, 2012, page 148-153, IEEE Computer Society.

[8] Software Testing(a Craftsman's Approach, Third Edition) - Paul C. Jorgensen.

[9] Software Testing Techniques(Second Edition)- Boris Beizer.