From Cause to Effect: An Empirical Study of Cause - Effect Graphing Testing Techniques and Its Test-Measurement: A Review

[1]Jawahar Lal, [2]Satinder Singh
[1]Dept. of Computer Engineering, F/O Engg. & Technology, Jamia Millia Islamia, New Delhi, India
[2]Dept. of Computer Science & I.T, Lyallpur Khalsa College, Jalandhar, Punjab, India

Abstract
This paper discusses a review on Empirical Investigation of Cause-Effect Graphing Testing Techniques and Its Test- Measurements. A Cause-Effect Graph in Software Testing has been defined as: "It is a directed Graph that maps a set of causes to a set of effects. The Causes may be thought of as input and the effect is thought of as the output". Usually the graph shows the nodes representing the causes on the left hand side and the nodes representing the effects on the right hand side. There may be intermediate nodes in between that combine inputs using logical operators such as AND, OR, Not etc.

Keywords
Cause-Effect-Graph, Decision Table, Test, Test case, Test Suite

I. Introduction

Software Testing [1] is a process of demonstrating that errors are not present. Its purpose is to show that a program performs its intended function correctly and it also builds the confidence that the program is performing its well defined function.
In general we can say that:
Software Testing= Software Verification + Software Validation.
Verification: Checking the Software with respect to Specifications.
Validation: Checking the Software with respect to Customer's Expectation.

II. Test, Test Case and Test Suite

Test and Test Case terms are used interchangeably [1]. Test case describes an input description and an expected output description. Inputs can be classified into 2 types:
Pre conditions (Circumstances that hold prior to test Case Execution) and the actual inputs that are identified by some testing methods.
Expected output is also of two types: Post conditions and actual outputs.
A good test case has a high probability of finding an error.
The test case designer's main objective is to identify good test cases. The template for a typical test case is given as below.

Table 1: A Test Case Template

Test case id:	
Section-I (Before Execution)	Section-II (After Execution)
Purpose:	Execution History:
Pre condition:	Result:
inputs:	If fails, any possible reason (optional)
Expected Outputs:	Any other Observation:
Post conditions:	Any Suggestion:
Written by:	Run by:
Date:	Date:

Test cases are valuable and useful- they need to be developed, reviewed, used, managed and saved.

III. Cause- Effect Graph

It is a Software testing technique that helps us to select, in a systematic way, a high-yield of test-cases. It considers only the desired external behavior of a system [7]. "This is a testing technique that aids in selecting test cases that logically relate causes (inputs) to effects (Outputs) to produce-test-cases.
The Graph Direction is as follows.

Causes ⟶ Intermediate Nodes ⟶ Effects.
The Graph can arranged so there is only one node between any input and any output."

IV. Step Used in Cause-Effect Graph

The Following Steps [4] are used to draw a Cause-Effect Graph.
Step1: For a module, identify the input conditions (Causes), and actions (output).
Step2: Develop a Cause-Effect Graph.
Step3: Transform Cause-Effect graph into a Decision Table.
Step4: Convert Decision table rules to Test cases. Each Column of the decision table represents a test case.
Basic Symbol used in Cause- effect graphs are as under.

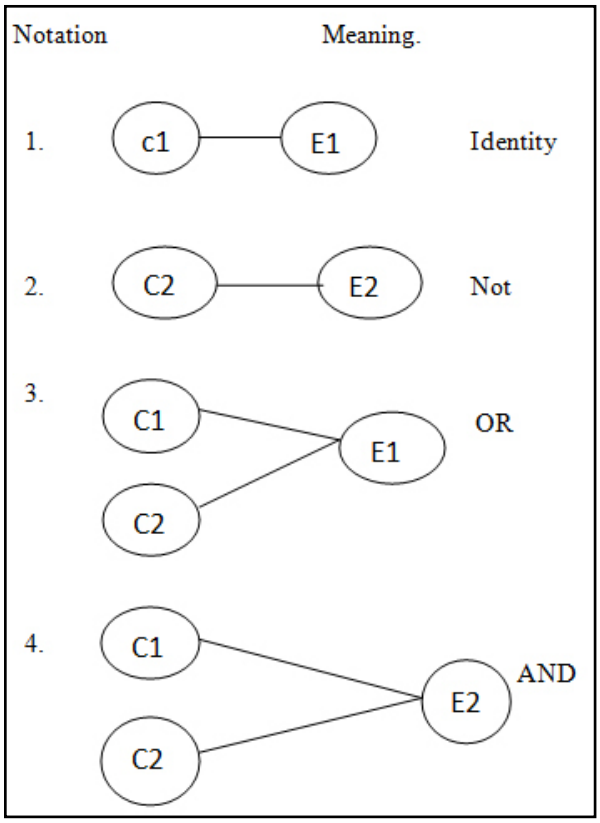

Fig. 1:

Cause- effect graphs captures [2] the relationship between specific combinations of inputs i.e. causes and effects.
The graphs also include a number of intermediate nodes linking Causes and effects. Some examples of the primitive cause-effects graphs are shown as follows:

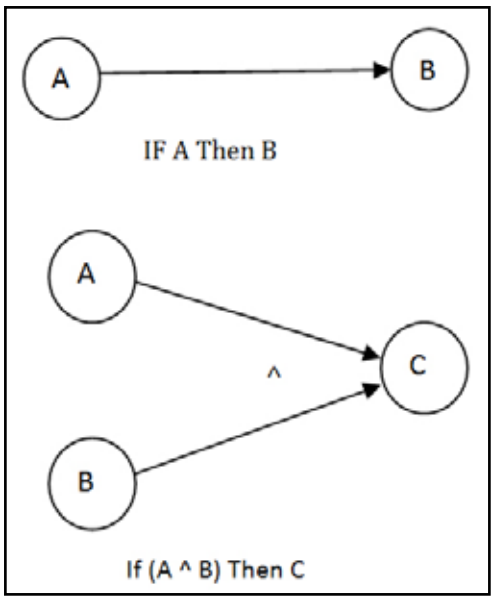

Fig. 2: Primitive Cause Effect Graphs

V. Derivation of Test Cases

The following process is used to derive test cases:
1. The specification is divided into "Workable pieces". For instance, when testing a timesharing system, a "Workable piece" might be specification for an individual command.
2. The causes and effects in the specification are identified.
- Causes: Distinct conditions
- Effects: An Output Condition or a system Transformation.

3. Assign a unique number to each cause and effect.
4. The semantic content of the specification is analyzed and transformed into a Boolean Graph linking the Causes and Effects. This is the Cause effect graph.
5. The graph is annotated with constraints describing combinations of causes and/or effects that are impossible because of syntactic or environmental constraints.
6. By methodically tracing state conditions in the graph, the graph is converted into a limited entry
Decision –table. Each Column in the table represents a test case.
7. The columns in the decision table are converted into test cases.
In the following example, [3] withdrawing money at an Automated Teller Machine (ATM) shall illustrate how to prepare a Cause-Effect Graph. The following conditions must be fulfilled in order to get money from the machine,:
- The bankcard is valid
- The PIN must be correctly entered
- The maximum number of PIN Inputs is three
- There is money in the machine, and in the account

The following actions are possible at the machine:
- Reject Card
- Ask for another PIN input
- "Eat" the card.
- Ask for an alternate dollar amount
- Pay the requested amount of Money

The given fig. 3 shows the Cause- effect graph of the example.

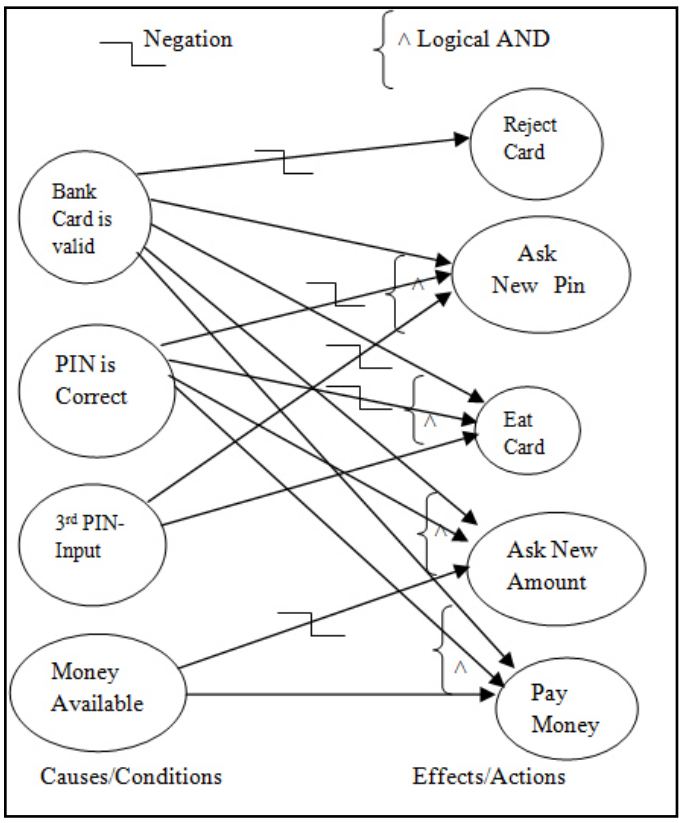

Fig. 3: Example of Cause Effect Graph

The Graph must be transformed into a → decision table from which the test cases can be taken. The steps to transform a graph into a table are as follows:
1. Choose an effect.
2. Look in the graph, find combination of causes that have this effect and combinations that do not have this effect.
3. Add one column into the table for every one of these cause- combinations and the caused states of the remaining effects.
4. Check if decision table entries occur several times and, if yes, delete them.

Table 2:

Condition/ Cause	1	2	3	4	5
Bankcard is valid	N	Y	Y	Y	Y
PIN is correct	-	N	N	Y	Y
3 Incorrect PIN	-	N(EXIT)	Y	-	-
Money Available	-	-	-	N	Y
Effect/Action					
Reject card	Y	N	N	N	N
Ask New Pin	N	Y	N	N	N
Eat Card.	N	N	Y	N	N
Ask New Amount	N	N	N	Y	N
Pay Money	N	N	N	N	Y

The above table is Optimized Decision table for the ATM.

A. Test with the Decision Tables

The test [3] based on decision tables has the objective to design tests for executing "interesting" combinations of inputs. Interesting in the sense that possible failures can be detected. Besides the causes and effects, intermediate results may be included in the

decision table.
A decision table has two parts. In the upper half, inputs (causes) are listed; the lower half contains the effects. Every column is a test case, i.e., the combination of condition and the expected effects or outputs for this combination.
In the least optimized case, every combination of causes is considered as a test case. However, condition may influence or exclude each other in such a way that not all combinations make sense. The fulfillment of every cause and effect is noted with a "yes" or "no". Each cause and effect should occur at least once with "yes" and "no" in the table.
Every column of the above table should be interpreted as a test case. From the table, the necessary input conditions and expected actions can be directly found. Test case 5 shows the following conditions: the money is delivered only if the card is valid, the PIN is correct after maximum three tries, and there is money available both in machine and in the account.
Let us take another Example,
A simple ATM banking Transaction system [2] can have the following causes and effects.

B. Causes (Inputs)

- C1: Command is withdrawal
- C2: Command is Deposit.
- C3: Account Number is valid
- C4: Transaction amount is valid

C. Effects (Outputs)

- E1: Print "Invalid Command"
- E2: Print "invalid Account Number"
- E3: Print " Withdrawal amount not valid"
- E4: Print "Successful Withdrawal made"
- E5: Print " successful deposit made"

Then the causes- effect graph can be drawn as shown in fig. 4 below:

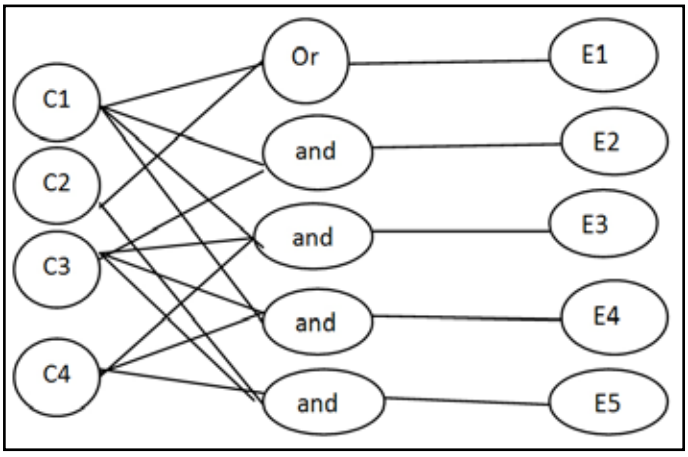

Fig. 4: Cause Effect Graph

If C1 or C2 is true, then one can have a valid command and thus, the negation of C1 or C2 will become an invalid command, i.e. E1. Similarly, if C1 is true, i.e. command is withdrawal (can choose C2 instead of C1), but C3 is false, i.e. Account Number is invalid, then the effect is "Print invalid account number i.e. E2" IF withdrawal command is correct, i.e. C1 is true, account number is valid, i.e. C3 is true, but the transaction amount is invalid, i.e. C4 is false, and then the effect is "Print withdrawal amount not valid" i.e. E3. Finally, for the effect E4, the causes can be C1, C3 and C4 and for the effect E5, The causes can be C2, C3 and C4. This is shown in cause-effect Graph. However in some ATM applications where deposit is not permitted, the effect E5 and the cause C4 can be omitted.
The Decision table for the ATM example against the cause-Effect graph is shown in table.

Table 3: Decision Table for the ATM

Decision Table					
	R1	R2	R3	R4	R5
C1	0	1	1	1	x
C2	0	x	x	x	1
C3	x	0	1	1	1
C4	x	x	0	1	1
E1	1	0	0	0	0
E2	0	1	0	0	0
E3	0	0	1	0	0
E4	0	0	0	1	0
E5	0	0	0	0	1

Each Rule can be reflected as one column in the Decision Table. 1 indicates True, 0 indicates false and x indicates don't care, i.e. May be true, May be false. All the rules that are used to draw the Cause-Effect graph can be put in this decision table. For Example, E2 effect requires C1 as true and C3 as false. Thus in the decision table, the second column against r2 contain 1 in C1, 0 in C3 and 1 against E2. Also, x is indicated in both C2 and C4. Similarly, all other columns can be created.

VI. Conclusion

This Paper is an Empirical Investigation and review of the Cause- Effect Graphing Testing Techniques along with its Test-Measurement. It summarizes the major aspects related to Cause-Effect Graphing testing techniques. First it gives an introduction to Software Testing, Later Verification and Validation and Test, Test Case, Test Suite and symbols used in drawing a Cause-Effect Graph and then steps required to draw a Cause-Effect Graph along with its Test data and then steps required to derive test cases.

References

[1] K.K. Aggarwal, Yogesh Singh,"Software Engineering: Program, Documentation", Operating Procedure: Cause-Effect Graph in Software Testing New Age Publishers 2010.
[2] Jibitesh Mishra, Ashok Mohanty,"Software Engineering: "Cause-Effect Graphs in Black Box Testing", pp. 221-223. 2011.
[3] Andreas, Spillner, Tilo Linz, Hans Schaefer,"Software Testing Foundations", 2nd Edition, Cause- Effect Graphing and Decision Table Technique: pp.126-129 September 2010.
[4] Pankaj Jalote,"An Integrated Approach to Software Engineering, (Springer Verlag 1991, 3rd Edition 2005.
[5] Aditya P.Mathur,"Foundation of Software Testing", fourth Impression, 2011 Dorling Kindersley (India) Pvt. Ltd.
[6] Bharat Bhushan Agarwal, Sumit Prakash Tayal,"Software Engineering", FIREWALL MEDIA (an Imprint of Laxmi Publication Pvt. Ltd.)
[7] [Online] Available: http://www.softwaretestinggenius.com/articalDetails.php?qry=498

[8] [Online] Available: http://en.wikipedia.org/wiki/Cause%E2%80%93effect_graph
[9] [Online] Available: http://www.searchsoftwarequality.techtarget.com/answer/Designing-test-cases-using-Cause-Effect-Graphing-Technique

Mr. Jawahar Lal is an Assistant Professor in the Department of Computer Engineering at Jamia Millia Islamia (A Central University), New Delhi. He obtained B.Tech Degree in Computer Engineering from Kurukshetra University in 2003. Later he obtained his Master Degree in Computer Technology & Application from Department of Computer Engineering, Delhi College of Engineering, Delhi. (University of Delhi.) Securing First Division in both. Prior to joining JMI, He worked with Department of Computer Science at Disha Institute, Jind-Haryana [2003-2004] as a member of faculty, thereafter as a Lecturer in Department of Computer Science Engineering at B.M. Institute of Engineering & Technology, Sonepat [2005] and Guru Tegh Bahadur Institute of Technology, Delhi. (G.T.B.I.T, Delhi.) –An Institute Affiliated to G.G.S. IP. University, Delhi. [2006]. He also worked with Binary Semantics Limited Gurgaon [2002],which is a Client- Centric global Software Development Company where he had performed Software Testing on Two Live Project named "Quality Control Management Information System (QCMIS)" and Web page testing of a website named "Knowledge Planet Virtual University (KPVU)" during 2002.
His Teaching interest includes- Software Engineering: focusing on SDLC models, Requirement Engineering, Testing, Predicate Logic, Internet Fundamentals, Operating System, Computer Architecture & Organization etc.
He has published many Technical report and papers in the field of Computer Engineering.

Satinder Singh received his master degree in Computer Science, M.Tech degree in Information Technology from Guru Nanak Dev University, Amritsar. He is presently working as Assistant Professor in the Department of Computer Science & I.T at Lyallpur Khalsa College, Jalandhar (Punjab) India. He has more than 6 years of teaching experience of post graduate classes. He has guided number of major and minor projects of graduate and post graduate classes. He has also supervised thesis of M.Tech students. His research interests include theory of automata, cloud computing, software engineering. He has to his credit more than 14 research papers in National and International conferences and 6 textbooks in Computer subjects.