Requirements-Based Testing
- Cause-Effect Graphing

Gary E. Mogyorodi, B.Math., M.B.A.
Certified Tester, Foundation Level (CTFL)
Certified Tester, Advanced Level – Functional Tester (CTAL-FT)
Certified Tester, Advanced Level – Test Manager (CTAL-TM)
President - Canadian Software Testing Board

President
Software Testing Services
Toronto, Ontario, Canada M8V 0A5
Phone: (647) 692-5040
Fax: (509) 356-6647
Email: garym@softestserv.ca
Web: www.softestserv.ca

Requirements-Based Testing – Cause-Effect Graphing

Introduction

This article provides an overview of the Requirements-Based Testing (RBT) process. RBT is a rigorous technique for improving the quality of requirements, while deriving the minimum number of test cases to cover 100% of those requirements. RBT is comprised of two techniques: Ambiguity Reviews and Cause-Effect Graphing.

An Ambiguity Review is used in the requirements phase of software development to identify ambiguities in functional[1] requirements. The intent of an Ambiguity Review is to identify anything that is unclear, not concise or ambiguous in the requirements. The elimination of these ambiguities improves the quality of those requirements.

Cause-Effect Graphing is a test case design technique that is performed once requirements have been reviewed for ambiguity, followed by a review for content. Requirements are reviewed for content to insure that they are correct and complete. The Cause-Effect Graphing technique derives the minimum number of test cases to cover 100% of the functional requirements to improve the quality of test coverage.

This article addresses the second RBT technique - Cause-Effect Graphing. The first article described Ambiguity Reviews and included a sample ambiguity review of the requirements I will design test cases for in this article.

Test Case Design

The test case design challenge is to derive the necessary and sufficient set of test cases to cover 100% of the functionality for the system under test. Without 100% functional coverage, there is a certainty that functionality will go into production untested.

There are many test case design techniques, but few insure that the test cases will provide 100% functional coverage. The Cause-Effect Graphing technique begins with the set of requirements, and determines the minimum number of test cases to completely cover the requirements.

Test Case Review

Once the test cases have been derived by the test case designer, the RBT process includes the review of these test cases by the following stakeholders:

- The author of the requirements verifies that he/she agrees with the test case designer's translation of requirements to test cases.
- The domain experts review the test cases in order to determine the answer to the following question: "Are we building the right system"?
- The developers review the test cases to clarify their understanding of the requirements. The developers learn what they will be tested on, and can therefore develop the software to succeed.

By performing these reviews, everyone on the project team can obtain the same understanding of what will be built.

Traditional Test Case Design Techniques

There are a number of test case design techniques, such as Equivalence Class Partitioning and Boundary Value Analysis. These techniques rely on the test case designer to manually work out the proper combinations of test cases. Often, the test case designer does not use a formal test case design technique and relies on his/her "gut feel" to assess whether test coverage is sufficient.

While these techniques do generate combinations of test cases, they often fall short on providing full functional coverage. Too often the normal flow or "go path" functionality has overlapping, redundant test cases, while exceptions and error conditions go untested.

Cause-Effect Graphing

The Cause-Effect Graphing technique was invented by Bill Elmendorf of IBM in 1973. Instead of the test case designer trying to manually determine the right set of test cases, he/she models the problem using a cause-effect graph, and the software that supports the technique, BenderRBT, calculates the right set of test cases to cover 100% of the functionality. The cause-effect graphing technique uses the same algorithms that are used in hardware logic circuit testing. Test case design in hardware insures virtually defect free hardware.

Cause-Effect Graphing also has the ability to detect defects that cancel each other out, and the ability to detect defects hidden by other things going right. These are advanced topics that won't be discussed in this article.

The starting point for the Cause-Effect Graph is the requirements document. The requirements describe "what" the system is intended to do. The requirements can describe real time systems, events, data driven systems, state transition diagrams, object oriented systems, graphical user interface standards, etc. Any type of logic can be modeled using a Cause-Effect diagram. Each cause (or input) in the requirements is expressed in the cause-effect graph as a condition, which is either true or false. Each effect (or output) is expressed as a condition, which is either true or false.

A Simple Cause-Effect Graphing Example

I have a requirement that says: "If A OR B, then C."

The following rules hold for this requirement:

- If A is true and B is true, then C is true.
- If A is true and B is false, then C is true.
- If A is false and B is true, then C is true.
- If A is false and B is false, then C is false.

The cause-effect graph that represents this requirement is provided in Figure 1. The cause-effect graph shows the relationship between the causes and effects.

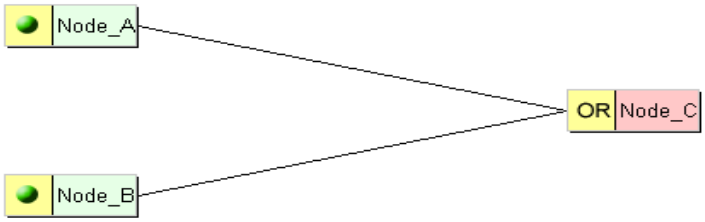


Figure 1 – A Cause-Effect Graph

In Figure 1, A, B and C are called nodes. Nodes A and B are the causes, while Node C is an effect. Each node can have a true or false condition. The lines, called vectors, connect the cause nodes A and B to the effect node C.

All requirements are translated into nodes and relationships on the cause-effect graph. There are only four possible relationships among nodes, and they are indicated by the following symbols:

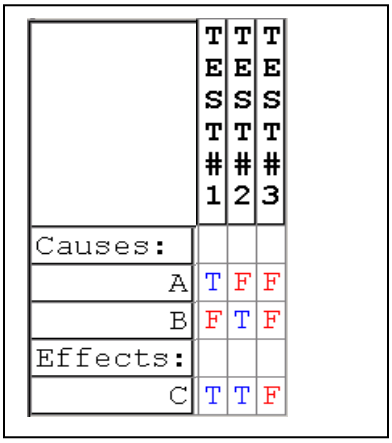
If A always leads to C, there is a black line that connects the nodes.
If A or B lead to C, there is an "OR" relation.
If A and B lead to C, there is an "AND" relation.
If not A leads to C, there is a red line that connects nodes.

The Decision Table

The cause-effect graph is then converted into a decision table or "truth table" representing the logical relationships between the causes and effects. Each column of the decision table is a test case. Each test case corresponds to a unique possible combination of inputs that are either in a true state, a false state, or a masked state (the masked state will be described in Figure 4 below).

Since there are 2 inputs to this example, there are 2 ** 2 = 4 maximum combinations of inputs from which test cases can be selected.

Figure 2 represents the decision table derived for the cause-effect graph in Figure 1.



	TEST #1	TEST #2	TEST #3
Causes:			
A	T	F	F
B	F	T	F
Effects:			
C	T	T	F

Figure 2 – The Decision Table

Notice that there are only three test cases. However, these three test cases cover 100% of the functionality for this example. The fourth combination of inputs (A= true and B = true) does not add any additional functional coverage, and is a redundant test case. Each relation is tested with the right combinations of causes so that all defects are covered for that relation, resulting in 100% coverage.

For more technical information about Cause-Effect Graphing, refer to the following:

Richard Bender
Bender RBT Inc.
17 Cardinale Lane
Queensbury, NY 12804
518 743-8755
www.benderrbt.com

G. J. Myers,
The Art of Software Testing,
Wiley, 1979.

Cause-Effect Graphing Analysis and Validation of Requirements
Khenaidoo Nursimulu and Robert L. Probert
Bell-Northern Research and Telecommunications Software Engineering Research Group
Department of Computer Science
University of Ottawa, 150 Louis Pasteur/Priv., Ottawa
Ontario, Canada K1N 6NA
www.cs.ubc.ca/local/reading/proceedings/cascon95/pdf/nursimul.pdf

Cause-Effect Graphing
Theresa Hunt
http://www.westfallteam.com/Papers/Cause_and_Effect_Graphing.pdf

Let's Create a Cause-Effect Graph

In the previous article on Requirements-Based Testing, we performed an Ambiguity Review and revised our original set of requirements to the following:

The Postal Regulation for South America (Version 1)

The following postal regulation determines postage surcharges. This regulation applies only to parcels, not other types of postal items, and only to parcels sent to South America. Other postal items are envelopes and post cards. There is no postage surcharge for envelopes and post cards. For parcels which people mail to South America between December 1 and December 24 of each year, the system will apply the surcharges in Table 1 in addition to the standard postage of $5.00 US:

Table 1 - Country & Weight Surcharge between December 1 and December 24

Argentina	All weights	$11 US
Brazil	Weight >= 34 pounds	$21 US
Brazil	Weight = 33 pounds	$19 US
Brazil	Weight <= 32 pounds	$17 US
Columbia, Ecuador, Peru, Bolivia, Chile, French Guiana, Guyana, Paraguay, Suriname, Uruguay, Venezuela, and Falkland Islands	All weights	$15 US

For parcels which people mail to South America outside of December 1 to December 24 of each year, the system will apply the surcharges in Table 2 in addition to the standard postage of $5.00 US:

Table 2 - Country & Weight Surcharge outside the dates December 1 to December 24

(In a real life problem, the specific details of Table 2 would be supplied here. For this example, they have been excluded.)

Figure 3 – The Postal Regulation Requirements Corrected for Ambiguity

The test case designer translates the requirements into the cause-effect graph shown in Figure 4 (created using BenderRBT).

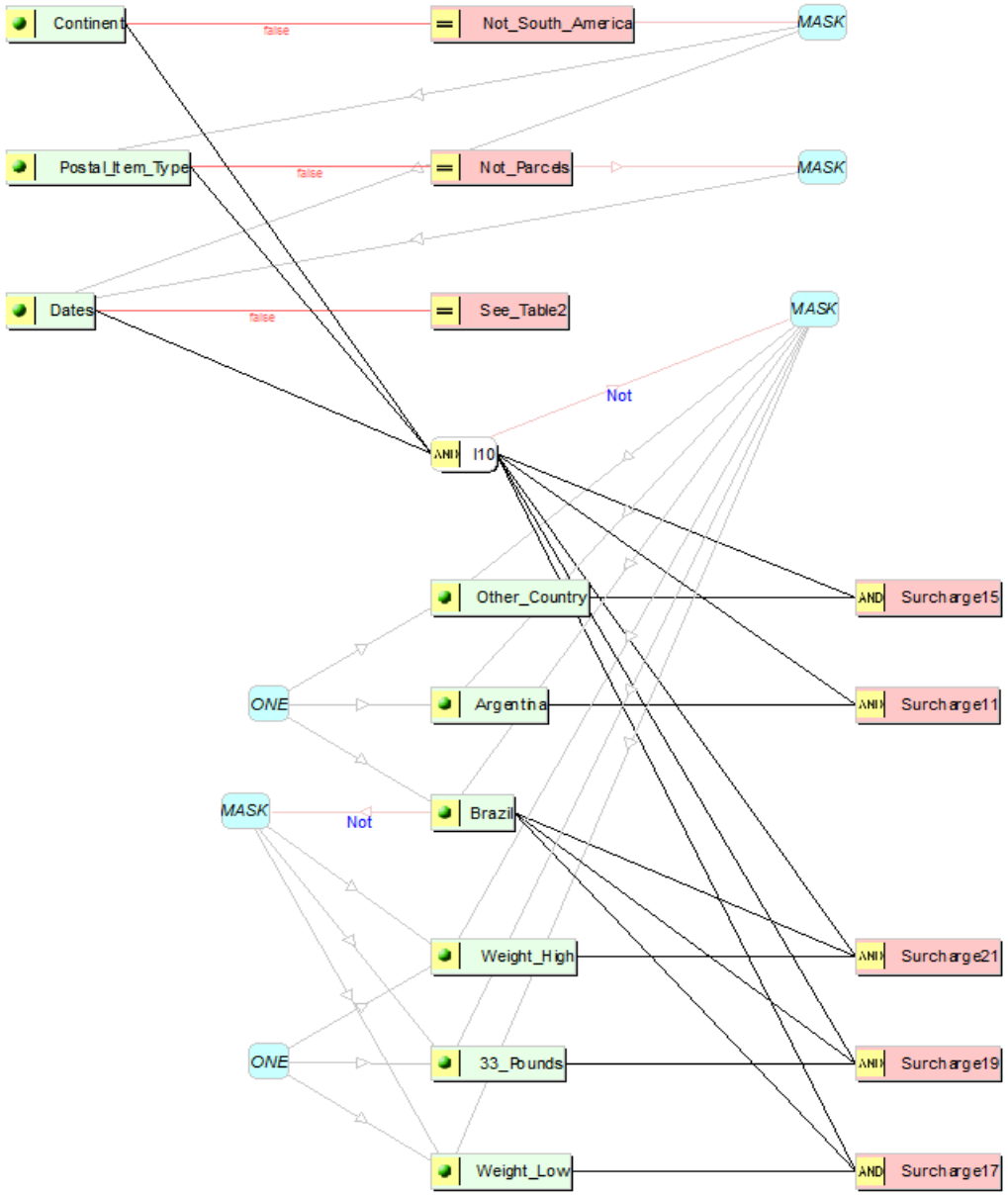


Figure 4 – Cause-Effect Graph for the Postal Regulation Requirements

The node I10 represents the state in which the item is sent to South America AND the postal item is a parcel AND the item is mailed between December 1 and December 24. There are a number of mask statements in the cause-effect graph. The mask statements define constraints on the system under test. As an example, the mask (Argentina, Weight_High, 33_Pounds, Weight_Low) means that when Argentina is the country, then the weight divisions associated with Brazil are irrelevant inputs.

The Cause-Effect Graph is translated into the following collapsed decision table (output from BenderRBT):

	↓	TEST#1	TEST#2	TEST#3	TEST#4	TEST#5	TEST#6	TEST#7	TEST#8
Causes:									
Continent		F	T	T	T	T	T	T	T
Postal_Item_Type		M	F	T	T	T	T	T	T
Dates		M	M	F	T	T	T	T	T
Argentina		M	M	M	T	F	F	F	F
Other_Country		M	M	M	F	T	F	F	F
Brazil		M	M	M	F	F	T	T	T
33_Pounds		M	M	M	M	M	T	F	F
Weight_Low		M	M	M	M	M	F	T	F
Weight_High		M	M	M	M	M	F	F	T
Effects:									
Not_South_America	{obs}	T	F	F	F	F	F	F	F
Not_Parcels	{obs}	m	T	F	F	F	F	F	F
See_Table2	{obs}	m	m	T	F	F	F	F	F
I10		F	F	F	T	T	T	T	T
Surcharge11	{obs}	F	F	F	T	F	F	F	F
Surcharge15	{obs}	F	F	F	F	T	F	F	F
Surcharge19	{obs}	F	F	F	F	F	T	F	F
Surcharge17	{obs}	F	F	F	F	F	F	T	F
Surcharge21	{obs}	F	F	F	F	F	F	F	T

Figure 5 – Decision Table for the Postal Regulation Requirements

Each column in Figure 5 represents a test case. The inputs are listed under the Causes, while the outputs are listed under the Effects. Each value in the matrix can be either T = true, F = false, or M = masked (an input that is masked is irrelevant to that test case). Each column represents a unique combination of inputs.

The following 8 test cases are derived for the Postal Regulation requirements using the Cause-Effect Graphing technique (output from BenderRBT):

TEST#1 -- The Postal Regulation

Cause(s):
The item is NOT sent to South America

Effect(s):
These conditions are out of scope for the Postal Regulation problem.

TEST#2 -- The Postal Regulation

Cause(s):
The item is sent to South America
The postal item is NOT a parcel

Effect(s):
These conditions are out of scope for the Postal Regulation problem

TEST#3 -- The Postal Regulation

Cause(s):
The item is sent to South America
The postal item is a parcel
The item is NOT mailed between December 1 and December 24

Effect(s):
See Table 2 for postage surcharges

TEST#4 -- The Postal Regulation

Cause(s):
The item is sent to South America
The postal item is a parcel
The item is mailed between December 1 and December 24 inclusive
The country is Argentina

Effect(s):
The postage surcharge is $11

TEST#5 -- The Postal Regulation

Cause(s):
- The item is sent to South America
- The postal item is a parcel
- The item is mailed between December 1 and December 24 inclusive
- The country is Neither Argentina NOR Brazil

Effect(s):
- The postage surcharge is $15 US.

TEST#6 -- The Postal Regulation

Cause(s):
- The item is sent to South America
- The postal item is a parcel
- The item is mailed between December 1 and December 24 inclusive
- The country is Brazil
- The item weighs 33 pounds

Effect(s):
- The postage surcharge is $19 US

TEST#7 -- The Postal Regulation

Cause(s):
- The item is sent to South America
- The postal item is a parcel
- The item is mailed between December 1 and December 24 inclusive
- The country is Brazil
- The item weighs 32 pounds or less

Effect(s):
- The postage surcharge is $17 US

TEST#8 -- The Postal Regulation

Cause(s):
- The item is sent to South America
- The postal item is a parcel
- The item is mailed between December 1 and December 24 inclusive
- The country is Brazil
- The item weighs 34 pounds or more

Effect(s):
- The postage surcharge is $21 US

In Summary

Figure 5 shows there are 9 inputs to the Postal Regulation requirements (i.e., Continent, Postal_Item_Type, Dates, Argentina, Other_Country, Brazil, Weight_High, 33_Pounds, Weight_Low). It is interesting to note that with 9 inputs, there are 2 ** 9 = 512 combinations of inputs from which test cases can be selected. The Cause-Effect Graphing technique derived only 8 test cases, but these 8 test cases cover 100% of the functionality described in the requirements. No other test cases are required to improve functional coverage. Additional test cases would only provide redundant coverage already supplied by some portion of each of the 8 original test cases.

Cause-Effect Graphing, in conjunction with a powerful analytical tool such as BenderRBT, provides a rigorous, consistent and cost effective approach to deriving test cases. Since the technique determines the minimum number of test cases, the test effort is minimized, and yet the test case designer can be confident that once these tests are successfully run, then testing is complete, and no untested functionality will move into production.

Notes: [1] Functional requirement - A functional requirement specifies what the system must be able to do, in terms that are meaningful to its users.

Gary E. Mogyorodi, B.Math., M.B.A.
Certified Tester, Foundation Level (CTFL)
Certified Tester, Advanced Level – Functional Tester (CTAL-FT)
Certified Tester, Advanced Level – Test Manager (CTAL-TM)

President
Software Testing Services
Toronto, Ontario, Canada M8V 0A5
Phone: (647) 692-5040
Email: garym@softestserv.ca
Web: www.softestserv.ca

