

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 6N5

Abstract

Cause-Effect Graphing (CEG) is used to derive test cases from a given natural language specification to validate its corresponding implementation. This paper surveys some known drawbacks to CEG analysis and shows how these can be overcome. It also shows how the CEG technique can be used to derive discriminating customer-directed scenarios (use-cases) for validating requirements.

Keywords: requirements engineering, cause-effect analysis, use-cases

1. Introduction

Cause-Effect Graphing is basically a hardware testing technique adapted to software testing by Elmendorf [3] and further developed by others [1, 2, 4-14]. The CEG technique is a black-box method, i.e, it considers only the desired external behavior of a system. As well, it

is the only black-box test design technique that considers combinations of causes of system behaviors.

In CEG analysis, first, identify *causes*¹, *effects*² and *constraints*³ in the (natural language) specification. Second, construct a CEG as a combinational logic network which consists of nodes, called *causes* and *effects*, arcs with Boolean operators (*and*, *or*, *not*) between causes and effects, and *constraints*. Finally, trace this graph to build a decision table which may be subsequently converted into use cases and,

¹ A cause represents a distinct input condition or an equivalence class of input conditions. A cause can be interpreted as an entity which brings about an internal change in the system. In a CEG, a cause is always positive and atomic.

² An effect represents an output condition or a system transformation which is observable. An effect can be a state or a message resulting from a combination of causes.

³ Constraints represent external constraints on the system.

eventually, test cases.

Myers proposed a CEG tracing algorithm to systematically select a *high-yield*⁴ set of test cases [7]. Some researchers [1, 14, 2] have pointed out ambiguities in Myers' approach. We investigate Myers' approach, illustrate strengths and weaknesses, and present possible corrections.

The quality of a system depends heavily on the validity of requirements. We show how to use CEG analysis as a requirements validation technique.

The paper is organized as follows: in the next section, we review Myers' approach to CEG analysis. Section 3 gives some resolutions to drawbacks in Myers' method. Section 4 shows how CEG analysis can be used as a requirement validation technique. We describe work in progress and conclude the paper in Section 5.

2. Analysis of Myers' CEG Approach

In this section, we illustrate Myers' approach, and discuss five apparent problems with this approach.

⁴High-yield set of test cases denotes a set of test cases adequate to detect major errors found in the implementation.

2.1 Example - Sendfile Command

In a given network, the sendfile command is used to send a file to a user on a different file server. The sendfile command takes three arguments: the first argument should be an existing file in the sender's home directory, the second argument should be the name of the receiver's file server, and the third argument should be the receiver's userid. If all the arguments are correct, then the file is successfully sent; otherwise the sender obtains an error message.

The above informal specification is first traced to derive causes and effects. Each one of these causes and effects are given a unique identification number (see Table 2.1).

From these causes and effects, along with the semantics that can be gathered from the specification, a CEG representation of the specification is built (Figure 2.2). We then add environmental constraints to the graph. In this case we do not have any.

Causes	Effects
1. The first argument is the name of an existing file in the sender's home directory.	100. The file is successfully sent.
2. The second argument is the name of receiver's file server.	101. The sender obtains an error message.
3. The third argument is the receiver's userid.	

Table 2.1 - List of causes and effects

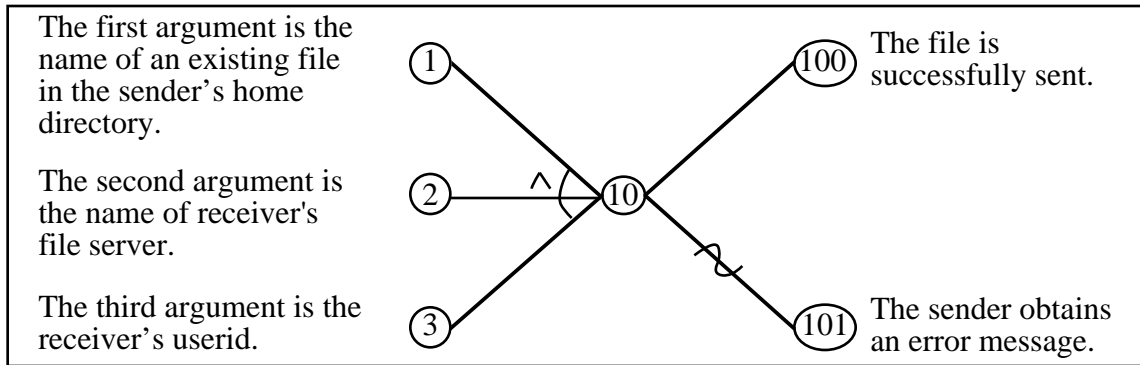


Figure 2.2 - The Cause-Effect Graph obtained

Figure 2.3 shows the decision table obtained from the CEG given in Figure 2.2 using Myers' rules of decision table derivation[7]. Each column in the decision table is then converted into a test case. For instance, suppose file f , server r , and userid u are all correct arguments, then the test case derived from column 1 in the decision table will be (*Sendfile f r u*) and the expected result is that file f is successfully sent.

2.2 An Analysis of Myers' Approach

Some authors [1, 2, 14] have noted some inconsistencies in Myers' CEG technique.

We discuss the more serious ones here.

	1	2	3	4	5	6	7	8
Causes								
1	1	0	1	0	0	1	1	0
2	1	0	0	1	0	1	0	1
3	1	0	0	0	1	0	1	1
Effects								
100	1	0	0	0	0	0	0	0
101	0	1	1	1	1	1	1	1

Figure 2.3 - Decision Table produced from CEG given in Figure 2.2 using Myers' rules

Common arguments against the CEG approach in general are the difficulty in identifying *causes* and *effects* because of their often vague description in natural language and the error-proneness in general of any method based on the *semantics* of a natural language specification. However, our experience is that a system domain

expert can often easily identify *causes* and *effects* even from a natural language specification and the corresponding *semantics*. Furthermore, since a CEG is built directly from an informal specification, this representation tends to be user-accessible and therefore promotes collaboration with the customer to detect ambiguities and incompleteness in the requirements. Finally, deriving *customer-directed scenarios*⁵ (use-cases) to check the specification itself (shown later in the paper) makes this method less error-prone and, hence, more cost-effective. Some problems still remain, however, and these are discussed below.

2.2.1 Observation 1

It can be argued that Myers' process of deriving a decision table from a CEG is described in an imprecise, difficult, and contradictory fashion [7]. Specifically, an initial statement of three considerations (pp. 65 in [7]) to be followed in the derivation the decision table contradict a summary (p. 67 in [7]). In consideration 2, Myers states that when tracing back through an

and node whose output should be 0, all combinations of inputs leading to a 0 output must be enumerated, except in the case where when one is exploring the situation where one input is 0 and one or more of the others are 1. Then, it is not necessary to enumerate all conditions under which the other inputs can be 1. In the summary, only one situation is enumerated under which one of the inputs is 0; this contradicts the previous statement. This is illustrated in Figure 2.4(a). In the derivation of the decision table for the CEG using Myers' consideration 2, when cause *c* is set to 1 all the combinations of causes *a* and *b* leading to output 0 at node *d* should be considered. The decision table in Figure 2.4(b) is thus obtained. When using the rules given in Myers' summary, (when cause *c* is set to 1 only one situation where output *d* is set to 0 should be considered), we derive the decision table in Figure 2.4(c). Thus, two different decision tables are obtained when applying the derivation rules. Such ambiguity will surely mislead the user and lead to inconsistency in application of the technique and difficulties in training.

⁵ A customer-directed scenario is a meaningful sequence of causes and effects used for validation purposes.

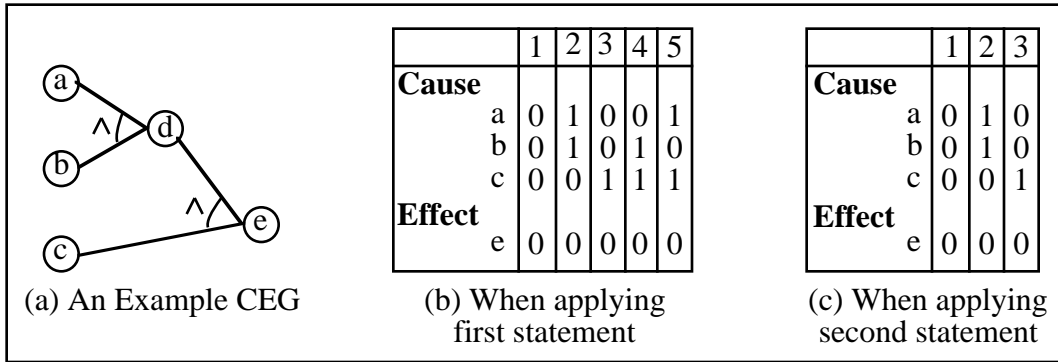


Figure 2.4 Contradiction in Myers' rules

This also affects the claim that situations are eliminated that tend to generate low-yield test cases.

2.2.2 Observation 2

Analyzing the rules given by Myers to derive a decision table showed that, if a CEG is written differently, i.e. the syntax is different while the semantics are the same, a different decision table could be obtained. This contradicts the reasonable requirement that equivalent semantics be tested equivalently. For instance, suppose we want to construct a CEG representing an effect e given by

$$e = (a \text{ and } b) \text{ or } (c \text{ and } d)$$

This effect is illustrated by the CEG given in Figure 2.5(a). Now, one of the other ways of writing effect e , using De Morgan's law, is as follows.

$$e = (a \text{ or } c) \text{ and } (a \text{ or } d) \text{ and } (b \text{ or } c) \text{ and } (b \text{ or } d)$$

If the new formula of e is used the CEG given in Figure 2.5(b) will be obtained. When Myers rules are used on the two CEGs, the decision tables given in Figure 2.5(c) and Figure 2.5(d) are obtained. Since the two decision tables are different they will give rise to two different sets of test cases. Therefore, with these rules, the set of test cases obtained from a CEG will depend on how effect e is formulated. Thus, these rules contradict a reasonable requirement for logical consistency.

2.2.3 Observation 3

When building a decision table, it often happens that some entries are left blank, meaning that they are irrelevant for the presence/absence of a given effect. For instance, in the decision table given in Figure 2.3 columns 5 to 8 contain some

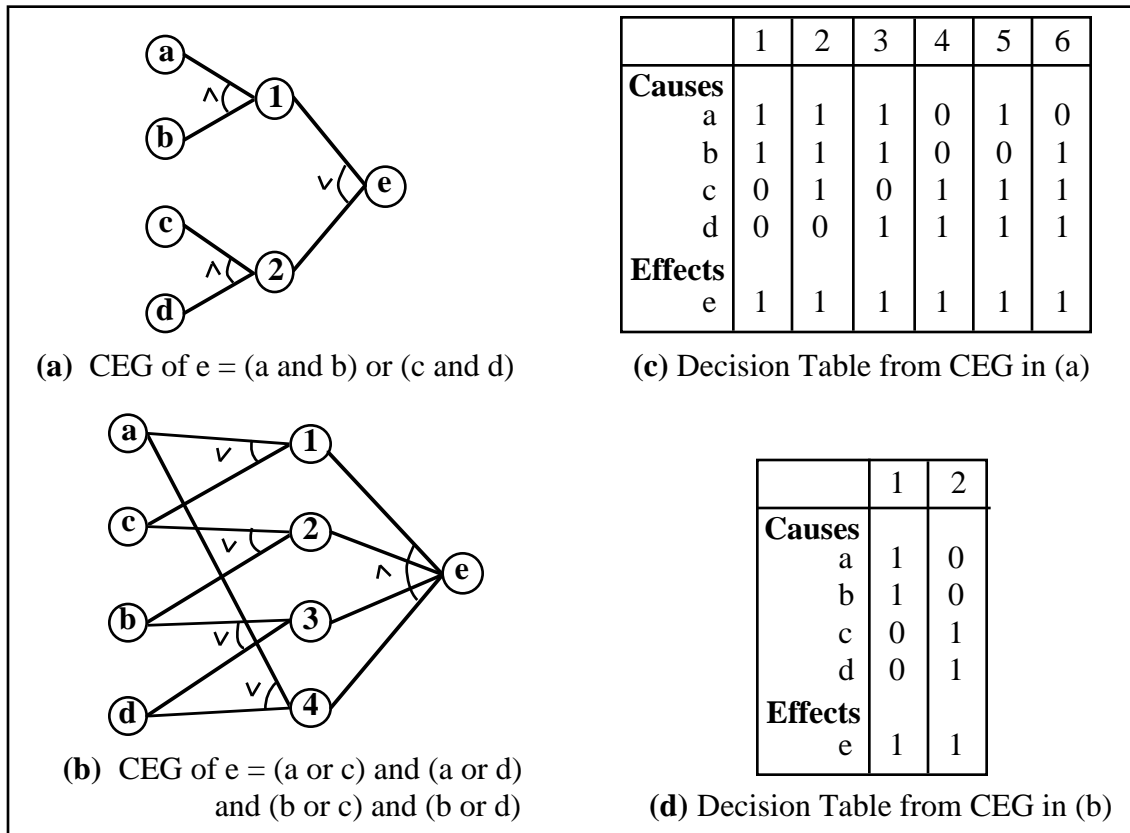


Figure 2.5 Logical Inconsistency in Myers' Rules

empty entries which represent *don't care*⁶ causes. It is reasonable that these *don't care* causes be instantiated before the test cases are derived. However, there is no obvious pattern to the way in which values are assigned to *don't care*. Again, leaving this exercise to the user's discretion may lead to low-yield test cases.

2.2.4 Observation 4

⁶ Don't cares refer to the set of causes whose presence/absence is not relevant for the presence/absence of a given effect.

One important aspect missing in the description in [7] of the CEG technique is that nothing is said about verifying conformance of the CEG to its corresponding natural language specifications. Overlooking verification may lead to faulty and incomplete test cases. For example, during the derivation of a decision table no considerations can be applied to detect any constraint omission in the CEG. This example is illustrated in Figure 2.6. It can be observed that the same decision table is obtained whether

or not the One-and-only-one, Inclusive, or Exclusive constraints are omitted. This may mislead the designer at the design phase if he/she

uses only the CEG to obtain insight into the specification.

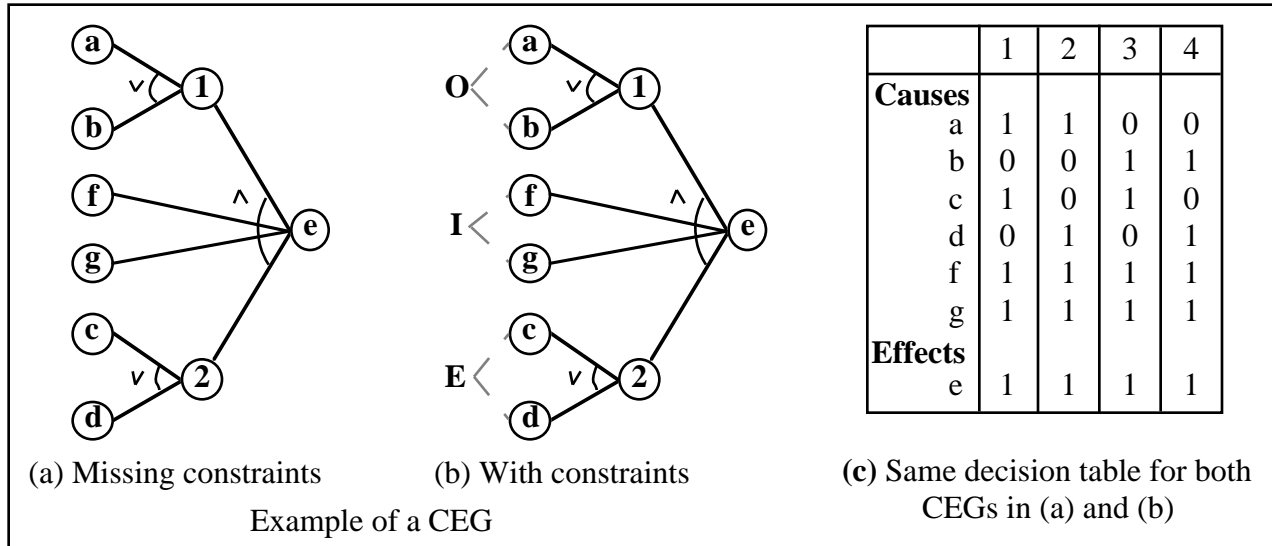


Figure 2.6 Superfluous Constraints Analysis

2.2.5 Observation 5

One of the other aspects that Myers does not consider when deriving test cases is the forcing of effects to be *absent* (corresponds to hardware stuck-at-one testing). He concentrates only on their presence. Setting an effect explicitly to *absent* may help to determine if that effect is always present or always absent.

NOTE : Even if some ambiguities and incompleteness have been noted in Myers' approach, the natural concepts of the CEG approach make it attractive from a human

usability perspective. The CEG technique is an improvement over informal, ad-hoc specifications of systems. It is systematic even though subjective in the first stage of constructing the CEG, and therefore more uniform, repeatable, and reliable. It is based on a graphical form of propositional logic which gives the user some degree of confidence in the specification power of the graph [1]. The following section will give a possible resolution to the above drawbacks in Myers' approach.

3. A Possible Solution to Drawbacks in Myers' Approach

Most of the defects in Myers' approach occur in the derivation of a decision table from its corresponding CEG. Therefore a better technique must be used during this phase. We found methods based on *Path Sensitization technique* [e.g. 2, 14] to be useful for this purpose. These methods also trace a CEG-like representation of the specification to build a decision table. The advantages of Path Sensitization Techniques are that they are all algorithmic and they consist of straightforward and unambiguous rules for deriving a decision table from a CEG. In Path Sensitization Technique combinations of causes are derived from the CEG by sensitizing each effect of the system to each cause in its corresponding *cause set*⁷. The combinations of causes obtained can be used to derive actual test data when the implementation of the system is tested. During the derivation of combinations of causes using the Path Sensitization Technique each

⁷ A cause set of an effect is the set of causes that can affect the presence or absence of that effect.

cause/effect is checked individually to see if it is *stuck_at_0* (always absent) or *stuck_at_1* (always present) [2].

While Path Sensitization methods solve the problems of contradicting rules and lack of test cases representing absence of effects in Myers' approach, they do not solve the other problems mentioned in Section 2. In this paper we present an intuitive approach, which we called the *Basic Path Sensitization Technique (BPST)*, to derive a decision table from a CEG. This technique has most of the advantages of Path Sensitization methods and at the same time it helps to resolve the drawbacks in Myers' approach.

Briefly, in BPST, a set of cause combinations are derived from a CEG for each effect, e , in such a way that effect e is sensitized⁸ to each cause, c , present in its cause set. The basic sensitization rules are given in Figure 3.1 and the algorithm is shown in Figure 3.2.

⁸ An effect e is said to be sensitized to a cause c when only cause c determines the presence or absence of effect e . We do this by setting all the other causes in the cause set of e to the specific values.

3.1 Analysis of BPST Algorithm

In the algorithm it can be observed that when an effect e is sensitized to a cause c there may be several ways of assigning values to the other causes in the cause set of effect e . When this case arises, we should select one cause combination where most of the causes are present (in the case where cause c is set to absent) and one cause combination where most of the causes are absent (in the case where cause c is present). Our choice is to test for the most unusual case, hence increasing the **importance of the contribution of cause c** . By this we mean that if most of the causes in the cause set of e are present and setting cause c to *absent* makes effect e absent too, then it can be said that absence of cause c has overridden the presence of all other causes. Hence, the importance of the contribution of cause c is said to be increased. Similarly, when effect e is present, we choose a combination of causes where the maximum number of causes is absent in order to determine if presence of cause c overrides the absence of all the other causes. This way of selecting combinations of causes brings an additional

high-yield guideline to the Basic Path Sensitization Technique, i.e **when sensitizing an effect to a cause, the combinations of causes that will increase the importance of the contribution of that cause are chosen.**

Figure 3.3 shows the decision table obtained from the CEG given in Figure 2.2 using the Basic Path Sensitization technique. Entries of causes and effects in bold represent the sensitized causes and effects. Also, column 5 has been added to detect any missing constraint (a case where most causes are present need not be added, as column 1 represents this case). For instance, consider column 2 in the decision table. This column represents two different sensitizations: the first is the sensitization of cause 1 to effect 100 (absence of cause 1 will lead to absence of effect 100 if we preset both causes 2 and 3 to present), and the second represents the sensitization of cause 1 to effect 101 (absence of cause 1 will lead to presence of effect 101 if we preset both causes 2 and 3 to present). Therefore, when deriving test cases from column 2 in the decision table, one should consider both

sensitizations and verify both absence and

presence of outputs.

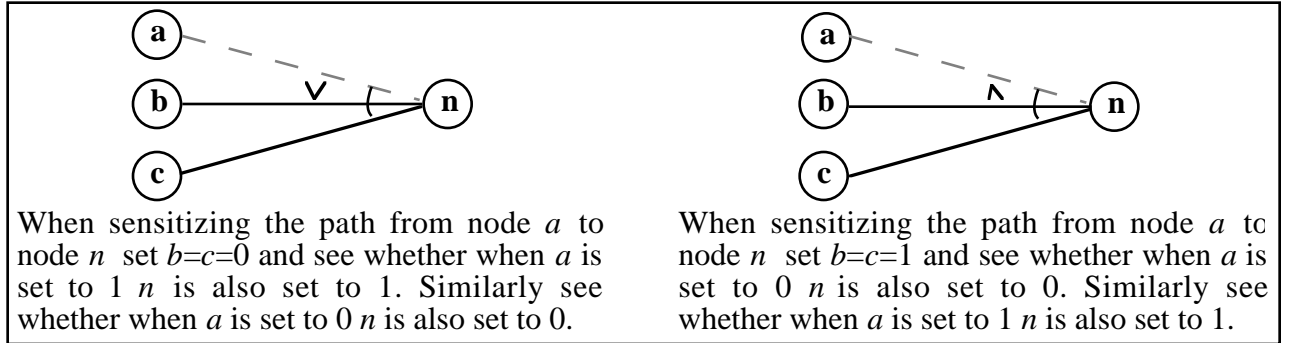


Figure 3.1 Rules for Path Sensitization Technique

In BPST, two cause combinations are derived for each cause c found in the cause set of a given effect e -- in one cause combination c will be present and in the other it will be absent. Moreover, two additional cause combinations are later derived for the entire graph to detect unintended absence of constraints. We can therefore come up with an upper bound for the number of cause combinations (hence, test cases) derived from a CEG using BPST. The upper bound is given by the following straightforward lemma.

Lemma 1

When using the Basic Path Sensitization technique, if a CEG has m effects and each effect e_i ($1 \leq i \leq m$) has n_i causes in its cause set then an upper bound on the number of

cause combinations that can be derived from the CEG is given by $2 + 2 \sum_{i=1}^m n_i$.

Besides, as only two cause combinations are derived for each cause in the cause set of an effect and the cause set of an effect is the same whatever the representation of the CEG, it follows that the number of cause combinations derived from a CEG is the same, independent of the CEG representation. This is summarized as the lemma below.

Lemma 2

The number of cause combinations derived from a CEG using the Basic Path Sensitization technique is the same whatever the representation of the CEG.

Proofs of these lemmas are given in [9]. Only $O(n)$ test scenarios are required, rather than 2^n .

```

Step 1 (* building a cause set for each effect *)
  Let g be a CEG.
  For each effect  $e_i$  in g do
    Backtrack g from  $e_i$  to build the Cause Set of  $e_i$ .
    Name this Cause Set  $C_i$ .
    At the same time build a set  $N_i$  to indicate all the intermediate nodes found on the path from  $e_i$  to its causes.
  endfor
Step 2 (* tracing from causes to effects to derive cause combinations for one effect *)
  For each cause  $c_i$  in Cause Set  $C_j$  (*of effect  $e_j$ *) do (* we are sensitizing  $e_j$  to cause  $c_i$  *)
    Present_Node =  $c_i$  (* node from where we start tracing *)
    Output_of_Node = x (* x denotes either presence or absence of Present_Node*)
    Repeat
      From Present_Node go to its forward node n such that  $n \in N_i$  or  $n = e_j$ 
      If relation at node n is "and" then
        Set all incoming arcs to n (except the one from  $c_i$ ) to 1
        (* We should therefore backtrack each one of these arcs to set values to the appropriate causes responsible for
           the arcs being set to 1. Note that two special cases may occur: i) several combinations of the causes may
           set the arcs to 1, ii) a given cause may be set to different values through separate paths*)
      Else (* relation at node n is "or" *)
        Set all incoming arcs to n (except the one from  $c_i$ ) to 0
        (* We should therefore backtrack each one of these arcs to set values to the appropriate causes responsible for
           the arcs being set to 0. The same special cases noted above may arise here also *)
      endif
      If the sensitized link from Present_Node to node n is set to "not" then
        Output_of_Node = not Output_of_Node
      Else
        Output_of_Node = Output_of_Node (* no change in value *)
      endif
      Present_Node = n
    until Present_Node = effect  $e_j$ 
    If Output_of_Node = x then
      Presence of cause  $c_i$  = Presence of effect  $e_j$  (* also absence of cause  $c_i$  = absence of effect  $e_j$ *)
    Else
      Presence of cause  $c_i$  = Absence of effect  $e_j$  (* also absence of cause  $c_i$  = presence of effect  $e_j$ *)
    endif
    Create two combinations of causes, one with cause  $c_i$  set to 0 (absent) and another one with  $c_i$  set to 1 (present)
    Place each combination of causes in a distinct column in the decision table (DT).
  endfor
Step 3 (* deriving cause combinations for all effects *)
  Repeat Step 2 for each effect  $e_j$  in g
  Remove any duplicate column in the DT
  Apply the constraints in the graph to each column in the DT to see if it is a valid cause combination
  Add two additional cause combinations one where most of the causes are present and one where most are absent (* to
  check for any missing constraints *)
Step 4 (* filling empty entries in the DT *)
  Fill in any empty entries in the DT by deducing their values from the values of the causes and effects in the given
  column and by looking at the constraints in the CEG.
  If there are still some empty entries then fill them with arbitrary values and keep a record of this (*Useful when
  generating test data *)

```

Figure 3.2 Basic Path Sensitization Technique (BPST) algorithm

The Basic Path Sensitization technique possesses several advantages. Besides consisting of a set of unambiguous rules, this technique tests for both presence and absence of effects and at the same time it determines if constraints are missing from a CEG. Moreover, this is the only technique that gives an upper bound on the number of test cases derived from a CEG. In addition the algorithm proposed is simple, easy to implement, and it works for every CEG.

	1	2	3	4	5
Causes					
1	1	0	1	1	0
2	1	1	0	1	0
3	1	1	1	0	0
Effects					
100	1	0	0	0	0
101	0	1	1	1	1

Figure 3.3 - Decision Table using BPST algorithm

4. The CEG as a Requirement Validation Technique

The CEG technique is a useful testing technique to test the implementation of a system by using test cases derived from the natural language specification of the system. This

Scenario number :- When CauseList Then EffectList
CauseList :- [Not] Cause
CauseList :- [Not] Cause and CauseList
EffectList :- Effect <i>present/absent</i>
EffectList :- Effect <i>present/absent</i> and EffectList

Table 4.1 - BNF of user-directed scenarios

technique is valid only if the natural language specification satisfies the customer's intentions. Thus, if the specification is incorrect we will end up with a set of incorrect test cases. Therefore, we must validate the specification itself. One way to do this is via *customer-directed scenarios (use-cases)*. A use-case represents a set of absent/present causes that determines the absence/presence of an effect. In a customer-directed scenario representation, the complete English text of each cause and effect will be used instead of their identification numbers. The scenarios are derived from their corresponding decision table where each column of the latter is converted into one scenario. Table 4.1 shows a BNF of customer-directed scenarios. The scenarios are called customer-directed because they are shown to the customer who should approve or disapprove each scenario. Therefore, these scenarios should be easy to read, understandable and unambiguous.

Figure 4.2 shows the list of customer-directed scenarios obtained from the decision table given in Figure 3.3. The derivation was straightforward. The scenario number corresponds to the column number in the decision table.

Scenario #	Requirements
1	<p>When the first argument is the name of an existing file in the sender's home directory and the second argument is the name of receiver's file server and the third argument is the receiver's userid</p> <p>Then the file is successfully sent and the sender does not obtain an error message</p>
2	<p>When the first argument is not the name of an existing file in the sender's home directory and the second argument is the name of receiver's file server and the third argument is the receiver's userid</p> <p>Then the file is not successfully sent and the sender obtains an error message</p>
3	<p>When the first argument is the name of an existing file in the sender's home directory and the second argument is not the name of receiver's file server and the third argument is the receiver's userid</p> <p>Then the file is not successfully sent and the sender obtains an error message</p>
4	<p>When the first argument is the name of an existing file in the sender's home directory and the second argument is the name of receiver's file server and the third argument is not the receiver's userid</p> <p>Then the file is not successfully sent and the sender obtains an error message</p>
5	<p>When the first argument is not the name of an existing file in the sender's home directory and the second argument is not the name of receiver's file server and the third argument is not the receiver's userid</p> <p>Then the file is not successfully sent and the sender obtains an error message</p>

Figure 4.2 - Test scenarios derived from decision table given in Figure 2.3

For instance, scenario 2 is an English description of column 2 in the decision table. Note that we express absence of a cause or effect by giving its corresponding negative form in order to enhance readability of the scenarios.

Also, note that “**or**” is not used, since test scenarios must not be ambiguous (ambiguity in causes leads to “retest” problems, and ambiguity in effects leads to confusion about test purpose. In both cases, ambiguity increases the

complexity of the test scenarios and is therefore undesirable).

5. Conclusion

In this paper we have shown some points of inconsistency and ambiguity in the Myers' approach to cause-effect graphing technique. We have also presented a possible reason for these drawbacks. In addition, we have shown how the cause-effect graphing technique can be used as a requirement validation technique too.

Even if some ambiguities and incompleteness exist in Myers' approach, the concepts of the CEG approach used in specifying the functional behavior of a system make it attractive from a usability perspective. The CEG technique is an advance over informal, ad-hoc specifications of systems. It is systematic even though subjective in the first stage of constructing the CEG, and therefore relatively uniform, repeatable, and reliable. It is based on a graphical form of propositional logic, which gives the user some degree of confidence in the specification power of the graph [1].

Several CEG tools exist presently on the market, for example, the SoftTest tool [12].

However, their objective is to derive test cases from the cause-effect graph in order to validate an implementation. Our goal is to validate the informal specification as well. Presently, we are developing a tool to derive the decision table and the customer-directed scenarios automatically from a given Cause-Effect Graph. The CEG is input in a text format; then our tool automatically draws the graph and outputs the decision table and the customer-directed scenarios.

In addition, we have developed an extension of this technique for validating requirements for highly interactive systems, called Extended CEG analysis. ECEG allows the direct encoding of timing and synchronization constraints. For more details, please see [10].

This technique possesses many benefits. However, it does possess some drawbacks too. The main drawback is probably the up-front cost of deriving causes, effects, and constraints from a given informal specification. This process requires domain knowledge and training in the CEG method. However, these up-front costs are small compared to the potential major downstream savings because they avoid

unnecessary rework and operational problems.

6. Acknowledgment

This research was partially supported by the Telecommunications Research Institute of Ontario (TRIO). The authors gratefully acknowledge the assistance of Mrs. L. Desrochers and Mr. A. Boni Bangari.

7. About the Authors

K. Nursimulu is a software engineer at BNR in Ottawa. Robert Probert is Professor of Computer Science at U. of Ottawa <bob@csi.uottawa.ca> or fax (613) 562-5185

References

- [1] Adler, M and Gray, M. A., "A Formalization of Myers Cause-Effect Graphs for Unit Testing", ACM SIGSOFT, Vol. 8, No.5, Oct. 1983, pp. 24-32.
- [2] S. Arabatzis, "Path Sensitization Technique for Test Case Design", Master's Thesis, University of Ottawa, 1986.
- [3] Elmendorf, W. R., "Cause-Effect Graphs in Functional Testing", TR_00.2487, IBM, Poughkeepsie, N.Y., Nov. 1973.
- [4] Elmendorf, W. R., "Functional Analysis using Cause-Effect Graphs", IBM, Poughkeepsie, N. Y. 1974
- [5] Elmendorf, W. R., "Functional Testing of Software Using Cause-Effect Graphs", IBM, Poughkeepsie, N. Y., 1975
- [6] Howden, W. E., "Functional Testing and Design Abstraction", Journal of System and Software 1, 1980, pp. 307-313.
- [7] Myers, G. L., "The Art of Software Testing", Wiley-Interscience, New-York, 1979.
- [8] Myers, G. L., "Software Reliability : Principles and Practices", Wiley-Interscience, New York, 1976.
- [9] K. Nursimulu, "Cause-Effect Validation of Distributed Systems", Masters thesis, University of Ottawa, February 1994.
- [10] K. Nursimulu and R L. Probert, "Cause-Effect Validation of Telecommunications Service Requirements", 6th Int. Conf. on S E and Applications, Paris, Nov. 15-19, 1993.
- [11] Probert, R. L., Ural, H. and Smith, E., "CEG Advisor User's Guide", CSI, University of Ottawa, July 1985.
- [12] Softest, "Requirements Based Testing CASE Tool", Release 3.2, Bender and Associates, inc., 1991.

[13] Tai, K.C, M.A. Vorcle, A.M. Paradkar,
P. Lu, "Evaluation of a Predicate-based
software Testing Strategy", IBM Systems
Journal, vol 33, No. 3, 1994.

[14] Walsh, P. J., "An Analysis of Test case
Selection", IBM , Endicott, New York 13760.