Fuzzy Cognitive Map for Software Testing Using Artificial Intelligence Techniques

Deane Larkman[1], Masoud Mohammadian[1], Bala Balachandran[1], Ric Jentzsch[2]

[1] Faculty of Information Science and Engineering, University of Canberra, ACT, Australia,
u950777@uni.canberra.edu.au, Masoud.Mohammadian@canberra.edu.au,
[2] Business Planning Associates Pty Ltd, ACT, Australia
Bus.Planning.Assoc@gmail.com

Abstract - This paper discusses a framework to assist test managers to evaluate the use of AI techniques as a potential tool in software testing. Fuzzy Cognitive Maps (FCMs) are employed to evaluate the framework and make decision analysis easier. A what-if analysis is presented that explores the general application of the framework. Simulations are performed to show the effectiveness of the proposed method. The framework proposed is innovative and it assists managers in making efficient decisions.

Key words: Software testing, Fuzzy Cognitive Maps (FCMs), What-if Analysis

1. Introduction

Software is a key element of systems and devices that support many of the activities that are an accepted part of our modern lifestyle. There is a consequent reliance on the correct behaviour of software, and an expectation that the software will not fail. However, software is an increasingly complex product that requires more and more testing a labour intensive and error prone activity [1]. The consequences of software failure range from the trivial (such as the need to restart a computer program), through the very inconvenient (such as the malfunction of traffic signals), to the catastrophic, where life and property may be affected. To minimise software failure, and its various impacts, high quality software must be created. Testing is a major quality assurance technique to evaluate the quality of software throughout the development cycle [1, 2].

2. Software Testing and Artificial Intelligence

2.1 Research on Software Testing using Artificial Intelligence

The application of AI techniques to testing software has moved beyond the speculative for many techniques, and now receives widespread attention from the research community. AI techniques have been used to explore different characteristics of a range of software, such as system software, real time software, embedded

software, distributed software, and GUI software. The diverse investigations cover many areas. Various test approaches, at different test levels (such as unit, integration and system testing), have been used; for example, white box or structural testing, black box or functional testing, grey box testing, GUI testing and non-functional testing. Different programming paradigms have been examined, including procedural programs, object-oriented programs and aspect oriented programs. Distinct test aspects have been investigated, such as test data generation and test oracle generation [3, 4, 5, 6]. Software programs have been transformed to facilitate the application of AI techniques. AI techniques have been optimised, modified, hybridized with other AI techniques, and adapted.

2.2 Limitations using Artificial Intelligence in Software Testing

Although AI techniques are grounded in theory, these theoretical foundations have rarely been developed to address software testing problems. Investigations about the theoretical basis for using AI techniques to test software are limited. Consequently, AI techniques for testing software lack a firm scientific basis. A recent paper analysed the theories underpinning genetic algorithms, developed these theories for structural test data generation, and empirically validated the predictions of the theories against real world programs [3]. The reason for using an AI technique is commonly analytical. When cited, the reason for using an AI technique is usually because of the similarities between the characteristics of the AI technique, and those characteristics of the software testing problems.

Much of the empirical evidence for the use of AI techniques to test software stems from artificial laboratory programs, which are small and simple programs: real world programs are often not considered for validation. Real world programs are not merely scaled up versions of laboratory programs; but present complex test problems, which frequently require an intensive manual test effort to solve. Many researchers have voiced concern about the need to validate AI techniques against real world programs, rather than against simple laboratory programs [3, 5, 6, 7]. The limitations of laboratory programs have been observed, and it has been noted that toy (simple) programs fail to reveal the limitations of some test data generation techniques [6].

3. Framework Software Testing Evaluating AI Techniques

A framework is at a high level generalised perspective of a domain of interest. It is a way to provide an initial understanding of some directed environment and its concepts or constructs. The framework described in this paper is a decision support framework, and was developed for use at an organisational environment level. It is aimed at test managers, or their equivalents – those people who, amongst other things, construct test plans, which encompass or are underpinned by test strategies.

Test managers are decision makers. The purpose of developing this framework is to support one aspect of that essential activity. Decision making is an increasingly complex activity with considerable potential for error. Decision makers need

to have at their disposal tools to help reduce the risks inherent in their decisions. The complexity of the real world dictates the requirement to have tools that can be used to help lower the decision risk by selecting and analysing amongst multiple alternatives. Test managers need to have a way to help them decide which test technique will be the most beneficial. The selected technique must be consistent with the test strategy or with the test approach used to evaluate the software artefact, and must help reduce the risk of software defects, especially the critical ones, not being found before the software is released to users, customers, clients, or the public at large.

The components of the AI techniques decision support testing framework are Test Management, Test Information, Test Environment, and Technical Support The framework objective is the possible application of AI Techniques to the particular software to be tested. All the components relate to the objective.

4. Fuzzy Cognitive Maps – Application to the Framework

4.1 Fuzzy Cognitive Maps

Fuzzy Cognitive Maps (FCMs) [9, 10] are graph structures that provide a method to capture and represent complex relationships within an environment (which defines a boundary), to improve understanding of that environment. FCMs have been used to model problems with no data [8, 9, 10]. A FCM can be used for what-if analysis, where several alternative scenarios to a given situation are considered [9, 10]. Concept nodes represent the environment behaviour within a FCM. The concepts are connected using arcs (arrows or edges) showing the relations between concepts. The framework's arcs are the influences between the concepts. The development of the FCM is based on the utilisation of domain experts' knowledge, forming a framework within that environment. Expert knowledge is used to identify concepts and the degree of influence between the concepts.

Use of a FCM to analyse the framework can only demonstrate general application of the framework. In contrast, validation of the framework involves specific, concrete applications of the framework. Concrete applications of the framework require case studies of different organisations in an industry environment. Validation of the framework is outside the scope of this paper.

4.2 FCM Construction

The FCM was constructed as follows. The events and the relationships defined by the proposed framework were used to build the graph structure of the FCM. Thus the FCM concepts are Test Management (C2) Test Information (C3), Test Environment (C4), Technical Support (C5) and the Application of AI Techniques to Software Testing (C1). Relationship weight values were assigned arbitrarily according to the authors' judgement, underpinned by industry experience in software testing of one of the authors. The FCM of the software testing framework is shown in Figure 1.

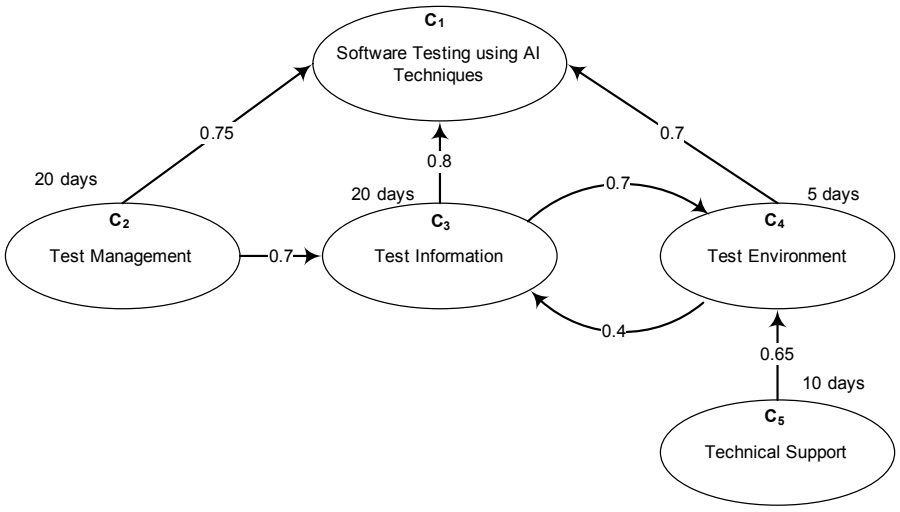

Fig. 1. Fuzzy Cognitive Map with Influences and Temporal

4.3 Static Analysis

If we consider the average effect of the influences from C2 + C3 + C4 (Figure 1) on C1 occurring, there is an estimated up to 75% chance that the software can be tested using an AI technique. Thus there is a 25% chance that some external influence will have an adverse, or even an advantageous effect, on the possibility of using an AI technique to test that particular software. Remember the 75% is based on the current knowledge of the domain expert.

From C2 to C1 shows 0.75. This says that at most (upper boundary) if all the elements that make up Test Management of the software to be tested are positive, then at most there is a 75% chance that AI Techniques will be selected for testing the software.

From C3 to C1 shows 0.80. However, C2 influences C3 by 0.70 and C4 influences C3 by 0.40. If left by itself C3's upper boundary is 80%, but this could be diminished or enhanced by the 70% influence from C2 and the 40% influence from C4.

From C4 to C1 shows 0.70. C4 has two influences on its upper boundary of 0.70: C3 and C5. In this situation C4's upper boundary of 70% is dependent on C3's 70% and C5's 65%. Therefore the influences from the concepts C3 and C5 can potentially provide C4 with constraints that could either adversely or enhance the affect of C4 reaching its upper boundary.

The preceding static analysis shows that a decision maker, in this context, works in a very complex environment. Thus any assistance to reduce the complexity of the environment, or make that complex environment more understandable or easier to interpret, would be a plus for the decision maker. The decision maker needs a way to comprehend the consequences of the interacting concepts of the

framework. They also need to be provided guidance for a better understanding of the elements within their decision, and for a better understanding of the consequences of the environment they are working with. The FCM inference mechanism is a technique that can be used to analyse the interacting knowledge captured by the framework.

4.4 What-If Analysis using the FCM Inference Mechanism

The dynamic dimension of a FCM is suitable for what-if analysis; where alternative scenarios (which involve what-if questions) are considered for a given situation. The solutions to the scenarios are obtained by using the FCM inference mechanism.

The FCM inference mechanism involves standard matrix multiplication. An initial state vector S_0 (a 1×n matrix) is multiplied by the FCM influence matrix, generating a new state vector S_1 as the next step. This process is repeated until the dynamical system reaches equilibrium. The influence matrix is an n x n matrix and each of its elements represents the influence values between the concept nodes. Values of concepts in new state vectors are calculated using the equation $c_i^{t+1} = f\left(W_{ji} c_j^t\right)$. The sigmoid function $f(x) = 1/\left(1 + e^{-\lambda x}\right)$ is used as the activation function. Large values of λ approximate the binary threshold or step function [9, 10]. In the step function f(x) = 1if x > T and f(x) = 0 if x ≤ T, where T is the threshold value, taken from somewhere in the fuzzy interval [0,1]. Thus concepts are either on (1) or off (0). An approximate binary threshold was adopted and the threshold value used was 0.5 [10]. The node to be tested is set to 1 in the input vector and in all the result vectors because it is a sustained input.

To develop a what-if analysis, scenarios are defined and the following process will be followed:

1) Create a connection or edge matrix E, which lists the values of the causal links between the nodes;
2) Define the scenario – select a node to test its effect or influence on the potential decision;
3) Create the initial state vector S_0. Set the node to be tested to1 (on) and set all other nodes to 0 (off). This ensures independent analysis of the test node. The node to be tested is modelled as a sustained input, so the test node is set to1 in all the result vectors;
4) Multiply S_0 by E to obtain the result vector S_1;
5) Repeat step 4 with each result vector (S_n * E) until equilibrium is reached – when a vector is repeated, ie the current iteration $S_{n+1} = S_n$;
6) Take the previous result vector S_n for the analysis; and
7) Repeat steps 2 to 6 for each scenario.

The influences among the concepts in Figure 1 can be displayed using the following influence or edge matrix ***E.***

$$E = \begin{array}{c} \\ C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_5 \end{array} \begin{array}{c} \begin{array}{ccccc} C_1 & C_2 & C_3 & C_4 & C_5 \end{array} \\ \begin{pmatrix} 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.75 & 0.00 & 0.70 & 0.00 & 0.00 \\ 0.80 & 0.00 & 0.00 & 0.70 & 0.00 \\ 0.70 & 0.00 & 0.40 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.65 & 0.00 \end{pmatrix} \end{array}$$

The test manager and other domain experts are required to determine the weights of the different links between the concept nodes, and the initial activation level of each concept. Later research will identify a set of questions to assist the test manager to determine weightings. In this test of the framework the authors have carefully considered the system and provided the weights for the FCM shown in Figure 1. Now what-if analysis can proceed.

Let us first analyse the influence of C2; so C2 is set to 1. Thus C2 can be examined independently of the other nodes and their influences. This situation is represented by $\mathbf{S_0}$ = [0, 1, 0, 0, 0].

S0 = [0, 1, 0, 0, 0]

S0 * E = [0.75, 0, 0.70, 0, 0] – this becomes S1 [1, 1, 1, 0, 0]

S1 * E = [1.55, 0, 0.70, 0.70, 0] – this becomes S2 [1, 1, 1, 1, 0]

S2 * E = [2.25, 0, 1.1, 0.70, 0] – this becomes S3 [1, 1, 1, 1, 0]

S3 = S2: equilibrium has been reached.

Let us next set C3 to 1. Thus C3 can be looked at independently of the other nodes and their influences. This situation is represented by $\mathbf{S_0}$ = [0, 0, 1, 0, 0].

S0 = [0, 0, 1, 0, 0]

S0 * E = [0.80, 0, 0, 0.70, 0] – this becomes S1 [1, 0, 1, 1, 0]

S1 * E = [1.5, 0, 0.40, 0.70, 0] – this becomes S2 [1, 0, 1, 1, 0]

S2 = S1: equilibrium has been reached.

Let us next set C4 to 1. Thus C4 can be looked at independently of the other nodes and their influences. This situation is represented by $\mathbf{S_0}$ = [0, 0, 0, 1, 0].

S0 = [0, 0, 0, 1, 0]

S0 * E = [0.70, 0, 0.40, 0, 0] – this becomes S1 [1, 0, 0, 1, 0]

S1 * E = [0.70, 0, 0.40, 0, 0] – this becomes S2 [1, 0, 0, 1, 0]

S2 = S1: equilibrium has been reached.

Let us next set C5 to 1. Thus C5 can be looked at independently of the other nodes and their influences. This situation is represented by $\mathbf{S_0}$ = [0, 0, 0, 0, 1].

S0 = [0, 0, 0, 0, 1].

S0 * E = [0, 0, 0, 0.65, 0] – this becomes S1 [0, 0, 0, 1, 1]

S1 * E = [0.70, 0, 0.40, 0.65, 0] – this becomes S2 [1, 0, 0, 1, 1]

S2 * E =[0.70, 0, 0.40, 0.65, 0] – this becomes S3 [1, 0, 0, 1, 1]

S3 = S2: equilibrium has been reached.

The framework is flexible and the test manager can modify the concepts and relationships of the framework to match their organisational circumstances, and the characteristics of the software being tested. As indicated the influence weights are test manager dependent and may not represent specific organisational settings.

The FCM converges to a fixed point for each scenario – single vectors are the result when equilibrium is reached. All the equilibrium vectors are *not* null, so

definite answers can be obtained for each scenario. The decision maker needs to be aware that effort into C4 can provide a more positive use of AI Techniques in testing of software. C3 and C5 have approximately the same degree of influence on the potential decision to use or not to use an AI technique on the software being tested. The influence of the Test Management node (C2) on the AI techniques decision is more difficult to determine, because its effect on the framework objective is both direct and indirect.

The information provided from what-if analysis of the framework can be used for decision analysis to support improved decision making by test managers. This approach provides a valuable tool for test managers to evaluate different scenarios for individual concepts, or combinations of concepts in the framework, and apply that evaluation in their organisation.

4.5 Temporal Analysis

FCM has introduced quantitative relationships between concepts to describe the strength of influence between elements. Miao in 2000 has shown that fuzzy cognitive maps generally do not provide a temporal mechanism to represent both the strength of influence and the temporal degree of the effect on the overall objective [11]. The FCM within this framework includes a temporal degree as illustrated in Figure 1. The days are computed by the test manager based on their best estimates of the software to be tested and historical data. The temporal aspect conforms to critical path analysis for the FCM. Combined with strength of influence this provides a solid basis for compressing a software testing life cycle.

5. Conclusion and Future Work

A limitation of the AI testing framework is a potential inability to capture all the important concepts from the software testing domain. Therefore the proposed testing framework may be subject to external influences from concepts overlooked in the software testing domain.

Future work involves:

1) Applying increased granular of the AI testing framework, and analysing any differences in the results between representations of the framework;
2) Using an FCM inference method that is able to map or transform concept values of state vectors to any value of the fuzzy interval [0,1], and comparing the results with the less discriminating FCM inference method illustrated in this paper;
3) Extended temporal to include degree of dependency states; and
4) Constructing a set of questions to help test managers more easily determine the weight values for the relationships between the concepts in the AI testing framework.

6. References

1. Dick, S. & Kandel, A. (2005). Series in machine perception and artificial intelligence, Vol. 63. Computational intelligence in software quality assurance. Hackensack, USA: World Scientific.
2. Hailpern, B. & Santhanam, P. (2002). Software debugging, testing, and verification. IBM Systems Journal, 41(1), USA, pp 4-12.
3. Harman, M. & McMinn, P. (2007). A theoretical & empirical analysis of evolutionary testing and hill climbing for structural test data generation. Proceedings of the 2007 International Symposium on Software Testing and Analysis, pp 73-83.
4. Hermadi, I. & Ahmed, M. A. (2003). Genetic algorithm based test data generator. The 2003 Congress on Evolutionary Computation 1, pp 85-91.
5. Howe, A. E., Von Mayrhauser, A. & Mraz, R. T. (1997). Test case generation as an AI planning problem. Automated Software Engineering, 4(1), USA, pp 77-106.
6. Michael, C. C., McGraw, G. & Schatz, M. A. (2001). Generating software test data by evolution. IEEE Transactions on Software Engineering, 27(12), USA, pp 1085-1110.
7. Kim, J.-M., Porter, A. & Rothermel, G. (2005). An empirical study of regression test application frequency. Software Testing, Verification and Reliability, 15(4), pp 257–279.
8. Smith, E. and Eloff, J. (2000). Cognitive Fuzzy Modeling for Cognitive Fuzzy Modeling for a Health Care Institution, IEEE Intelligent Systems March/April 2000, USA, pp 69-75.
9. Kosko, B. (1997). Fuzzy engineering. Upper Saddle River , USA: Prentice Hall.
10. Kosko, B. (1991). Neural networks and fuzzy systems: A dynamical systems approach to machine intelligence. Englewood Ciffs, NJ: Prentice Hall.
11. Miao, Y. Zhi-Qiang, L. (2000). On causal inference in fuzzy cognitive map. IEEE Trans. Fuzzy Syst., vol 8, pp 107 -119.