Reliability Assessment by using Neural Network and Fuzzy Analysis for the Software Cost Effective Evolution Exertion

Lakshmana Rao Padala
Research Scholar, Department of Computer Science and Engineering
JJT University, Chudela, Jhujhunu, Rajasthan, India.

Dr. E Mohan
Principal
P.T.Lee.Chengalvaraya Naicker College of Engineering and Technology
Kanchipuram, Tamil Nadu, India.

Abstract - Fuzzy logic is a mathematical model used for different applications indexing from the control of engineering system to artificial intelligence. Application of fuzzy logic improves a better set of solution. The Software schema, applies the fuzzy logic to increase the quality of product involving the human knowledge and experience. The main theme of the paper is to describe the software engineering principles applied to improve the process of working with fuzzy logic. The system software development process is devised by a standard project failure. Basic reason for the project failure is due to lack of requirement specification engineering, which has to be broadly classified and iteratively discussed and documented. Other factors like impecunious project management exercise, impecunious design tactic and incapable of testing methods also contributing to project failures. The main reason of fit fall is that we use traditional bi-logic structure for decision making. The output of the process is either yes or no in bi-logic system. The Maxim of Uncertainty in Software Engineering (MUSE) states that the unpredictable in genetic and irresistible in software development processes and products. The above fit falls can be easily solved by the Fuzzy logic, because fuzzy logic has ability to deal with unpredictable and multi-logic e.g. an individual problem domain has 0.5 probability or 0.8 probability to be taken as a class, whereas in classical bi-logic, there is only two probability values either 0 or 1. So, measurement levels increases from bi-logic to multi-logic, consequently the measurement error will be reduced and minor data loss at initial stage of evolution. Fuzzy logic uses membership functions to absorb multi variants and values.

Keywords: Fuzzy Logic, bi-logic, multi-logic, Project Management, Requirement Specifications, MUSE.

I. INTRODUCTION: FUZZY-LOGIC HEURISTICS

The selection of using fuzzy logic is to clarify a complication in a precise realm has its own objections. The initial stage is to develop a fuzzy standard from the adroit intellectual for the complication realm. This stage comprises the architecture of the fuzzy result. The architecture leads to model stage, the result needs to be unified into the manufacture environment, and the schema which affords the fuzzy logic based design with aid and applies the yield of the fuzzy logic result. The assimilation stage can involve a series of complications; relatively build up on the target system. The steps ramified in the manufacture of the fuzzy logic solution, lead to slow down the evolution and add important learning loop to the procedure. Based on this reason the approval of fuzzy solution in different fields is denied.

Recently, an ample number of software evolution methods have been imported. Methods aspire at conceive software antiquity through the function of a large number of laws and all the laws are based on classical bi-logic system. For example, the Object-Modeling Technique determines the laws and eliminates the object-oriented simulation antiquities such as classes, associations, attributes and operations, and inheritance relations. These antiquities will not involve in the model and so they results uncertainty and therefore, aim at resolving uncertainty whenever they are encountered.

The multi-logic rule of current object-oriented methods has to be updated. Consider the rule of Candidate Class Identification: If any entity in requisite factor is an ample-value and can remain independent value, independent in the application domain, then select it as an ample-value relevant candidate class.

In this, an entity and a candidate class are the antiquity types to be secular, important and independent are the properties, and relevance-value and autonomy-value indicate the domains of these properties. For example,

ample-value may represent the set of values {Weak, Slight, Fair, Substantial, Strong}, and independent-value may represent the set of values {Dependent, Partial Dependent, Autonomous}. Various domain stages can be assert using disjoint sets conclude in an abrupt division of elements between distinct sets. One of the major defects of disjoint sets is that they do not provide competent means for apprehend the closer values, approximate essence of the software development process. Further we have to examine other models of logic than classical logic.

In fuzzy logic, the concept of ambiguity is introduced by the definition of fuzzy set. A fuzzy set S of a universe of discourse U is characterized by a membership function is μs: U → [0, 1] which associates with each element y of U a number in the interval [0, 1] which represents the grade of membership of y in S. By the definition of fuzzy sets, the concept of linguistic variables is introduced to represent a language typically adopted by a human expert. A lexical variable is a variable whose values, called lexical values, have the form of phrases or sentences in a natural language. An important issue here is to determine the lexical values (i.e. Weak, Strong). Many researchers have explored definitions of lexical values and it has been appear that familiarly used English terms could be defined with a sensible accuracy. Explicit rules of present object-oriented methods using fuzzy-logic is an extension of work, which modeled the heuristic rules of object oriented methods using bi-logic.

II. ARTIFICIAL NEURAL NETWORK-BASED (ANN) APPROACH BY USING BI-LOGIC

Artificial Neural Network-based (ANN) approach for estimating both prior and posterior probability using the historical data of the software organization. For each project objective (O) a threshold function (TO) can be stated, which provides success and failure measures. Note that, TO, in this framework, is a function of many factors as mentioned in Section 3 and as explained below. We could also define metrics on software quality aspects such as Defect Rate, Effort per Defect, Number of Changes per Release (TP1, TP2 … TPn). Therefore, with regard to Figure 3, TB could be a threshold function on Cost Performance Index (CPI) and TS a threshold function on Schedule Performance Index (SPI) or Productivity (ratio between SLOC and Effort). By impacting factor, we mean a variable that can affect the measurement of success metric (e.g., Domain, KSLOC, Complexity, and Developers' Expertise). It is possible that we should take in account a big number of factors to get good results. But, the more this number grows, the more regression function is difficult to be estimated, if we use canonical methods. However, if we use ANN and Back propagation for calculating regression functions, the number of factors does not influence the procedure [6].

Of course, the effects of this grow impact on the computation time, which could be very big. Hence, we are considering ANN as a mean to estimate nonlinear regression functions because we can obtain reliable estimates as automatically as possible, even if we have to consider a high number of impacting factors. Refer to (Bishop, 1995) and (Dreyfus, 2005) for more information about the use of ANN and Back propagation as a mean to estimate non-linear regression functions [9].

Building a Bi-logic Classification Network: So far we have turned our evaluation problem into a bi-logic discrimination problem. Now, we build a new network, for discrimination, as depicted in Figure 6. Such a system is able to classify all possible inputs as belonging to A or B. Actually; we can obtain much more from this kind of network. In fact, this is able to provide the posterior probability said above (Bishop, 1995). This is a very important result because, generally, Bayes' theorem has just theoretical importance, but it cannot be practically applied. The problem is that we cannot know all terms of Equation right side. In order to explain this problem let us consider Equation (2). Let Pr(A | x) be the posterior probability that Bayes' theorem provides, that is, the probability that the considered project belongs to A after its measurements have been executed.Pr(A) and Pr(B) represent prior probability of class A and B respectively, pX(x | A) and pX(x | B) represent the probability density that x occurs conditioned to class A or B, that is, the probability that x occurs with respect to A or B.

$$\Pr(A \mid x) = \frac{p_X(x \mid A)\Pr(A)}{p_X(x \mid A)\Pr(A) + p_X(x \mid B)\Pr(B)}$$

The real problem is that we could, somehow, get estimates for P(A) and P(B), but we are not able to estimate pX(x | A) and pX(x | B), that is, Bayes' Theorem is not useful for real classifications. Based on the mathematical

proof reported in (Bishop, 1995), if we use an ANN like the one, we can directly obtain Pr (A | x) even if we cannot know the conditioned probabilities said above [5].

III. FUZZY LOGIC ENGINEERING SPECIFICATIONS

The main advantage of a fuzzy logic access is that it grants the developers to target on the decision logic of the algorithm. Also, the construction of a fuzzy logic access desires that an algorithm for a explicit issue is to develop on a bottom up approach, where all auxiliary steps of the decision algorithm are explained and edited to reach the final solution. This appears as a cost effective approach.

An uncertainty will be associated while constructing a design and determining to use fuzzy logic. It involves the preceding stage of architecture of the fuzzy logic result for a specific algorithm. Designing a realm specific problem in pseudo code is easier then concluding the clone in a fuzzy logic description. It is very uncertain that the realm expert has knowledge as well as an expert in fuzzy logic. It includes further difficulties in the preservation of the algorithm implementation.

At present many tools are deployed for fuzzy logic solutions and have prudent support for designing the algorithm. These tools help in debugging and fine tuning. The complication is that the development status is rarely the same as the production status, where the fuzzy logic explanation will be applied. Also, classification of the result is not taken into application.

Many tools for fuzzy logic evolution affect a set of accessible fuzzy logic functions (operators, defuzzification, etc.) but they do not equip a genuine way to add additional functions. In most of the cases there is no possibility to add characteristic functions to the solution. This is a serious argument when a used function is disapproved by the fuzzy logic exertion.

Let us consider that N is the number of quantization levels and A is the maximum value of the signal. Then, the Root Mean Square (RMS) value of the quantization error is calculated as:

$$`\epsilon = A/2(N-1) * \sqrt{1/3}$$

The RMS value of the quantization error (ϵ) is inversely proportional to the number of quantization levels. In current software development methods, high quantization errors arise from the fact that rules adopt only bi-logic levels.

The drawbacks with Software Project Management can be easily countered by the fuzzy logic, because fuzzy logic has ability to deal with multi-logic e.g. an entity in a problem domain has 0.5 probabilities or 0.8 probabilities like that to be taken as a class, whereas in classical bi-logic, there are only two probabilities either 0 or 1. So, quantization levels increases from 2 to more than two therefore the quantization error will reduce and so less information loss at beginning phases of development. Fuzzy logic could allow software development especially engineering specification step because it can absorb lexical constraints that is very common in human communication. Many Fuzzy Object Oriented (OOF) modeling has been proposed for modeling estimate specifications, fuzzy objects, and fuzzy typing etc. Gyseghem et al. represent fuzzy information as fuzzy sets and uncertainty by means of generalized fuzzy sets [9]. A universal fuzzy class is imported to grab fuzziness accomplice with aspects.

Lee proposes a Fuzzy Object Oriented (FOOM) modeling technique to abduction and evaluates estimated specifications through the following two steps:

(1) Identify the possible types of fuzziness involved in the modeling of imprecise requirements, such as classes, rules, attributes and associations and

(2) Investigate the potential impacts of incorporating the notion of fuzziness on the features of OO.

FOOM uses fuzzy formation technique to calculate the unity degree between a class and an object, and a class and its subclass. Marin et. al. endeavour for performing the fuzzy type accomplice to a class, handling the problem of ambiguous in the database model, and describing the fuzzy type (structural and behavioural components). A structural component is a fuzzy set describes the set of all the possible aspects and the behavioural component is a fuzzy set describes the set of all the possible methods in the data model. The two

structures of inheritance and instantiation have been altered to take benefit of fuzzy types. The mechanism of instantiation can be allowed to choose the act of aspects of the type to represent objects[12].

IV. FUZZY LOGIC - PROJECT MANAGEMENT

The degree of certitude in a project is not always apt-able. The risk of handling precise outputs leads to beyond essentials. These constraints can be seen as the cumulative decline of software measurements to perform the inherent uncertainties in managers' knowledge of the development products, resources, and processes. The fuzzy logic techniques can help to overcome some of these difficulties by performing the defect in inputs and outputs, as well as availing a more expert knowledge based access to model building. The use of fuzzy logic for project management will not be the same throughout the development life cycle. Different level of accessible instruction and desired attention suggest that it can be used differently based on the current phase; despite of single model can be used for flexibility.

The vital aid of fuzzy logic for software engineering project management is the resilience in terms of the types of input and output variables. Input variables can be expressed as elementary fuzzy design (a huge number of entities in the data model), fuzzy numbers (about 250 entities), or using precise values (265 entities). Relatively, the output can be revealed in the clone, as a label (a short development time), fuzzy number (about 400 developer-hours), or precise values (378 developer hours).

Fuzzy Logic can be used as stature evaluation function. Stature evaluation can be done in advance to make exceptional methods (exceptional stature of the job, it divide the job into dissociable factors), to aid in advance (can range of the job, can improve part of the work), to learn and build estimating skills. Estimation is an uncertain process no one knows how big the product will be. Estimation can be biased by business and other pressures and earlier the estimate, the less is the accuracy of estimation. This is how we can use concept of fuzzy logic in size estimation: 1. Gather size data on previously developed programs, 2. Divide the historical product size data into size ranges as very large, large, medium, small, very small, 3. Compare the planned product with these prior products, 4. Based on this comparison, select the size that seems most appropriate for the new product. Estimation based on Fuzzy logic contributes logically good estimates.

V. FUZZY LOGIC - SOFTWARE TESTING

According to MUSE, ambiguity permeates these processes and products. Plans to test this antiquity, therefore, will carry their uncertainties forward. Software testing, like other development activities, is human intensive and thus introduces uncertainties. These uncertainties may affect the development effort and should therefore be accounted for in the test plan. In particular, many testing activities, such as test result checking, are highly routine and repetitious and thus are likely to be error-prone if done manually, which introduces additional ambiguity. Test planning activities are carried out by humans at an early stage of development, thereby introducing uncertainties into the resulting test plan. Also, test plans are likely to reflect uncertainties that are, as described above, inherent in software antiquity and activities. Test enactment includes test selection, test execution, and test result checking. Test enactment is inherently uncertain, since only exhaustive testing in an ideal environment guarantees absolute confidence in the testing process and its results. This ideal testing scenario is infeasible for all but the most trivial software systems. Instead, multiple factors exist, discussed next, that introduce uncertainties to test enactment activities.

Test selection is the activity of choosing a finite set of elements (e.g., requirements, functions, paths, data) to be tested out of a typically infinite number of elements. Test selection is often based on an adequacy or coverage criterion that is met by the elements selected for testing. The fact that only a finite subset of elements is selected inevitably introduces a degree of ambiguity regarding whether all defects in the system can be detected. One can therefore associate a probability value with a testing criterion that represents one's belief in its ability to detect defects. An example of assigning confidence values to path selection criteria is given below.

Test execution involves actual execution of system code on some input data. Test execution may still include uncertainties, however, as follows: the system under test may be executing on a host environment that is different from the target execution environment, which in turn introduces ambiguity. In cases where the target environment is simulated on the host environment, testing accuracy can only be as good as simulation accuracy. Furthermore, observation may affect testing accuracy with respect to timing, synchronization, and other

dynamic issues. Finally, test executions may not accurately reflect the operational profiles of real users or real usage scenarios. Test result checking is likely to be error-prone, inexact, and uncertain. Test result checking is afforded by means of a test oracle that is used for validating results against stated specifications. Test oracles can be classified into five categories, offering different degrees of confidence. Specification-based oracles install the highest confidence, but still include ambiguity stemming from discrepancies between the specification and customer's informal needs and expectations.

The execution of a fuzzy logic solution in production is usually a black box. The surrounding software solution provides the inputs and receives the output after the processing. The issue with this is that there is no way of accessing partial results. If a result of the fuzzy logic algorithm is incorrect or differs from the development environment results, there is no simple way to pinpoint the source of the error. There is no way to trace the steps the fuzzy logic solution does to get to the final result. The issue might be reproducible with the appropriate steps, but there is no guarantee and it still requires an effort. This is a mandatory requirement in modern software environments which serve numerous clients all around the clock.

VI. FUZZY LOGIC SOLUTION

The mentioned issues in the previous sections are not an issue in many use cases, but still these are important problems which need to be taken into consideration. Fuzzy logic solutions have the possibility to efficiently solve the problem of concurrent programming in modern software environments. By definition a fuzzy logic decision always produces the same output for the same inputs. This means that it is referentially transparent. This property is true for all of the steps taken to reach the final result. This means that with a proper implementation for fuzzy logic tools the decision steps can be executed concurrently to take advantage of modern hardware. The designed fuzzy logic based DSL can be described as a declarative approach to domain logic development which takes advantage of modern hardware environments. This is a significant advantage. There is no need to design the algorithm with parallel [8, 9, 10] execution in mind. The DSL takes care of it where the dependencies make it possible.

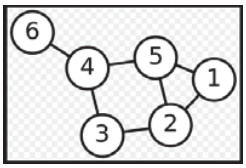

Figure 1 Execution graph

Based on the dependencies, which include partial inputs, partial results and so on it possible to build an execution graph [11]. The execution graph is the workflow for the fuzzy logic decision implementation. Figure 1 show how an execution graph can look like. In case of a fuzzy solution we know ahead of time some initial dependencies. For example all fuzzy rules can be executed concurrently and their results collected. Fuzzy operators and aggregations can be concurrently processed with data parallel algorithms [12, 13].

The fuzzy logic solution proposed by the author is referentially transparent and based on persistent data structures. The advantages are that no race conditions can occur, no need for synchronization between execution threads, also all calculations are done in a different context, meaning every partial and final result is available.

VII. CONCLUSION

Fuzzy logic solutions need to adapt to the requirements of modern software in order to remain a viable approach for the broad field of applications. However Artificial Neural Network-based (ANN) approach is bi-logic function and so fuzzy logic is more adaptable. Fuzzy logic is a complex topic which requires proper software based solution to achieve efficient usage in real life use cases. In some cases, hiding this complexity is the key to success. In other cases by embracing its possibilities and advantages modern software engineering problems can be conquered elegantly and efficiently.Hence, we can say that application of fuzzy logic in software development will definitely improve development processes by considering and in turn reduce information loss and provide greater design alternatives at later stages of development.

REFERENCES

[1] B. Londeix. Deploying realistic estimation (field situation analysis). Information and Software Technology, 37:665-670, 1995.

[2] S. Kumar, B.A. Krishna, and P.S. Satsangi. Fuzzy systems and neural networks in software engineering project management. Journal of Applied Intelligence,4:31-52, 1994.

[3] K. Srinivasan and D. Fisher. Machine learning approaches to estimating software development effort, IEEE Transactions on Software Engineering, 21:126-137, 1995.

[4] T. Mukhopadhyay, S.S. Vicinanza, and M.J. Prietula. Examining the feasibility of a case-based reasoning model for software effort estimation. MIS Quarterly, 16:155-171, 1992.

[5] A.R. Gray, and S.G. MacDonell. A comparison of model building techniques to develop predictive equations for software metrics. Information and Software Technology, to appear, 1997.

[6] G Bellmann, R.E., Zadeh., L.A., Local and fuzzy logic, Modern Uses of Multiple-Valued Logic, edited by Dunn, J.M., Epstein, G., Reidel Publ., Dordrecht,1977, The Netherlands, pp. 103-165

[7] Dubois, D. Prade, H., What are fuzzy rules and how to use them, Fuzzy Sets and Systems 84, 1996, pp. 169-185

[8] Dubois, D. Prade, H., Fundamentals of fuzzy sets, The handbook of fuzzy sets series, Kluwer Academic Publishers, Boston, 1999

[9] Fullér, R., Fuzzy Reasoning and Fuzzy Optimization, Turku Centre for Computer science, TUCS General Publication, N0 9, September 1998., ISBN 952-12-0283-1

[10] Marta Takacs, Approximate Reasoning in Fuzzy Systems Based on Pseudo analysis and Uninorm Residuum, Acta Polytechnica Hungarica, Volume 1, Issue Number 2, 2004, pp.49-62

[11] Sram, N., A hybrid approach to decision making and problem analysis, RAAD 2010, pp. 431 – 433

[12] Erik Wynters, Parallel processing on NVIDIA graphics processing units using CUDA, Journal of Computing Sciences in Colleges, v.26 n.3, p.58- 66, January 2011

[13] Sadaf Qamar , Syed Hasan Adil, Comparative analysis of data mining techniques for financial data using parallel processing, Proceedings of the 6th International Conference on Frontiers of Information Technology, December 16-18, 2009, Abbottabad, Pakistan