NEURO FUZZY LOGIC MODEL FOR COMPONENT BASED SOFTWARE ENGINEERING

Harpreet Singh [1], Vishal Kumar Toora [2]
singh.harpreet89@hotmail.com , vishal.alawalpuria@gmail.com

Abstract

Fuzzy logic has proved its mettle in last few decades and has been used in various applications to improve the performance and embeds some intelligence into the system. Fuzzy logic also solves the problem of non linear systems and handles them with great efficiency and provides robustness to the system. However, our aim always lies in achieving the improved solution and there are different hybrid algorithms. In this paper, the recent data based artificially intelligent techniques like Fuzzy have been customized and used .The application/case study has been taken from a research paper which appeared in a reputed general. The case study deals with reusability of software components. The attributes are coupling, volume, complexity, regularity and reuse frequency. In such data search application the design and developed Neuro-fuzzy hybrid algorithm has shown its superiority because it includes the advantages of Fuzzy as well as neural networks. Neuro -fuzzy algorithms is definitely superior to Fuzzy algorithm as it inherits adaptability and learning. From the simulation and the result obtained, it has been shown that the percentage average error is less in Neuro-fuzzy model. Neuro-fuzzy algorithm has yielded accuracy greater than the accuracy levels as in the case of Fuzzy logic for software reusability.

Keywords: Fuzzy logic, Neuro-fuzzy, Software reusability

I. Introduction

Component-based software engineering (CBSE) intends to build large software systems by integrating pre-built software components. The high productivity is achieved by using standard components. The principles of CBSE can be best described by the following two guiding principles: reuse but do not reinvent (the wheel); assemble pre-built components rather than coding line by line. Software assets [5], or components, include all software products, from requirements and proposals, to specifications and designs, to user manuals and test suites. Anything that is produced from a software development effort can potentially be reused. That quality of a piece of software, that enables it to be used again, be it partial, modified or complete is called reusability. The process of reuse consists of four major activities viz. manage the reuse infrastructure (MRI), produce reusable assets (PRA), broker reusable assets (BRA) and consume reusable assets (CRA). A few terms are defined to facilitate our discussion: Producers are those who create reusable assets with the specific goal of reusability. Function of Manage the Reuse Infrastructure (MRI) is to establish the reuse rules, roles, and goals in the infrastructure to support reuse. The Produce Reusable Assets (PRA) activities develop, generate, or reengineer assets with the specific goal of reusability. PRA includes domain analysis and domain engineering. The Broker Reusable Assets (BRA) activity aids the reuse effort by qualifying or certifying, configuring, maintaining, promoting and brokering reusable assets. The Consume Reusable Assets (CRA) activity occurs when systems are produced using reusable assets.

In the field of artificial intelligence, neuro-fuzzy refers to hybrids of artificial neural networks and fuzzy logic. Neuro-fuzzy hybridization results in a hybrid intelligent system that synergizes these two techniques by combining the human-like reasoning style of fuzzy systems with the learning and connectionist structure of neural networks. Neuro-fuzzy hybridization is widely termed as Fuzzy Neural Network (FNN) or Neuro-Fuzzy System (NFS) in the literature. Neuro-fuzzy system (the more popular term is used henceforth) incorporates the human-like reasoning style of fuzzy systems through the use of fuzzy sets and a linguistic model consisting of a set of IF-THEN fuzzy rules. The main strength of neuro-fuzzy systems is that they are universal approximators with the ability to solicit interpretable IF-THEN rules. The strength of neuro-fuzzy systems involves two contradictory requirements in fuzzy modeling: interpretability verses accuracy. In practice, one of the two properties prevails. The neuro-fuzzy in fuzzy modeling research field is divided into two areas: linguistic fuzzy modeling that is focused on interpretability, mainly the

Mamdani model and precise fuzzy modeling that is focused on accuracy, mainly the Takagi-Sugeno-Kang (TSK) model. Although generally assumed to be the realization of a fuzzy system through connectionist networks, this term is also used to describe some other configurations including, fuzzy logic based tuning of neural network training parameters, fuzzy logic criteria for increasing a network size representing fuzzification, fuzzy inference and defuzzification through multi-layers feed-forward connectionist networks, realising fuzzy membership through clustering algorithms in unsupervised learning in SOMs and neural networks deriving fuzzy rules from trained RBF networks. It must be pointed out that interpretability of the Mamdani-type neuro-fuzzy systems can be lost. To improve the interpretability of neuro-fuzzy systems, certain measures must be taken.

II. Component based software Engineering

Component based software engineering is the process of implementing or updating software systems using existing software assets. Software assets, or components, include all software products, from requirements and proposals, to specifications and designs, to user manuals and test suites. Anything that is produced from a software development effort can potentially be reused. That quality of a piece of software, that enables it to be used again, be it partial, modified or complete is called reusability. The following steps for implementing Component based software Engineering process

Step1: Assess organizational readiness: Understand the people, process, product, technology, asset, economic, metric and management facets of the organization and how reuse will impact each of these aspects.

Step2: Identify and collect metrics: While this activity is done throughout the reuse effort as necessary, collecting metrics early will enable us to benchmark the organization and show the impact when reuse is implemented.

Step3: Identify domains in the organization: Enumerate a list of domains that are in common within the organization.

Step4: Analyze the domain: An informal domain analysis may be conducted for the chosen domain. This analysis includes determining features common to systems in the domain and assessing the range of variability.

Step5: Examine the existing organizational structure: Consider establishing an independent producer group. This would dedicate resources to ensure that the necessary assets are created, managed and supported.

Step6: Create and manage reusable assets: Make, buy or re-engineer existing assets for users. Bring these assets under a source control and configuration management system.

Step7: Utilize tools, technology, and standards: Examine whether to create or use existing tools, technology and standards for your reuse program.

Step8: Conduct reviews and walkthroughs to reinforce reuse: Throughout the product development life cycles, perform reviews to ensure adherence to reusability objectives.

In this research paper the considered data table for software reusability is given in table1, below. Software reusability has shown to be dependent on 5 attributes viz. coupling, volume, complexity, regularity and reuse frequency.

III. Neuro Fuzzy logic model

The process of fuzzy inference involves all of the pieces that are described in the previous sections: Membership Functions, Logical Operations, and If-Then Rules. There are two types of fuzzy inference systems that can be implemented in Fuzzy Logic Toolbox: Mamdani-type and Sugeno-type. These two types of inference systems vary somewhat in the way outputs are determined. See the Bibliography for references to descriptions of these two types of fuzzy inference systems. Fuzzy inference systems have been successfully applied in fields such as automatic control, data classification, decision analysis, expert systems, and computer vision. Because of its multidisciplinary nature, fuzzy inference systems are associated with a number of names, such as fuzzy-rule-based systems, fuzzy expert systems, fuzzy modeling, fuzzy associative memory, fuzzy logic controllers, and simply (and ambiguously) fuzzy systems.

Mamdani's fuzzy inference method is the most commonly seen fuzzy methodology. Mamdani's method was among the first control systems built using fuzzy set theory. It was proposed in 1975 by Ebrahim Mamdani as an attempt to control a steam engine and boiler combination by synthesizing a set of linguistic control rules obtained from experienced human operators. Mamdani's effort was based on Lotfi Zadeh's 1973 paper on fuzzy algorithms for complex systems and decision processes. Although the inference process described in the next few sections differs somewhat from the methods described in the original paper, the basic idea is much the same. Mamdani-type inference, as defined for Fuzzy Logic Toolbox, expects the output membership functions to be fuzzy sets. After the aggregation process, there is a fuzzy set for each output variable that needs defuzzification. it is

possible, and in many cases much more efficient, to use a single spike as the output membership function rather than a distributed fuzzy set. This type of output is sometimes known as a singleton output membership function, and it can be thought of as a pre-defuzzified fuzzy set. It enhances the efficiency of the defuzzification process because it greatly simplifies the computation required by the more general Mamdani method, which finds the centroid of a two-dimensional function. Rather than integrating across the two-dimensional function to find the centroid, you use the weighted average of a few data points. Figures 1, 2, 3, 4 and 5 show fuzzy sets taken for each input variable in Mamdani model in this case.

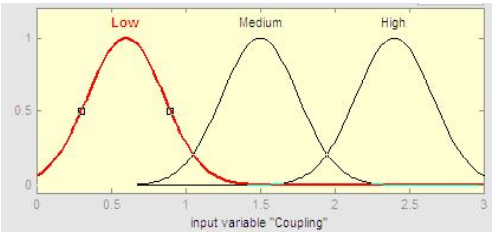

Figure1: Input variable "coupling" fuzzy sets in Mamdani model

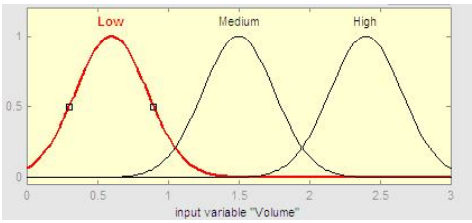

Figure2: Input variable "volume" fuzzy sets in Mamdani model

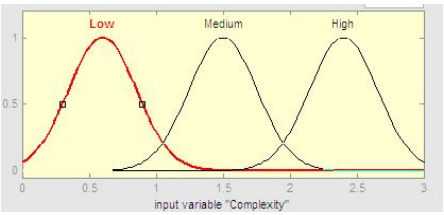

Figure 3: Input variable "complexity" fuzzy sets in Mamdani model

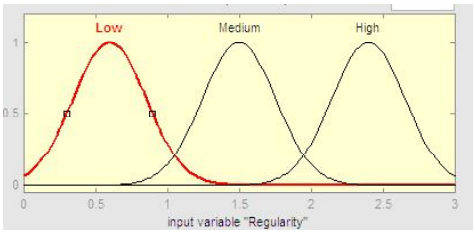

Figure 4: Input variable “regularity” fuzzy sets in Mamdani model

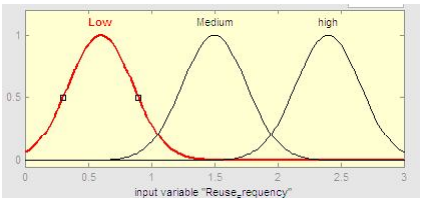

Figure 5: Input variable “reuse frequency” fuzzy sets in Mamdani model

Figure 6 shows fuzzy sets taken for output variable in Mamdani model in this case. Figure 7 is

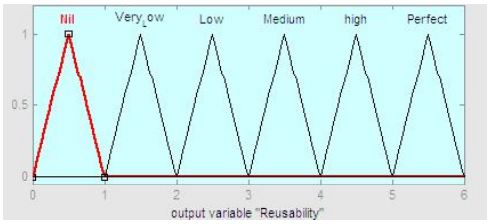

Figure 6: Output variable “software reusability” fuzzy sets in Mamdani model

the snapshot of few sample rules of the rule base for fuzzy model for software reusability.

1. If (Coupling is Low) and (Volume is Low) and (Complexity is Low) and (Regularity is Low) and (Reuse_frequency is Low) then (Reusability is Nil) (1)
2. If (Coupling is High) and (Volume is High) and (Complexity is High) and (Regularity is Low) and (Reuse_frequency is Low) then (Reusability is Nil) (1)
3. If (Coupling is High) and (Volume is Low) and (Complexity is Low) and (Regularity is Low) and (Reuse_frequency is Low) then (Reusability is Nil) (1)
4. If (Coupling is High) and (Volume is Low) and (Complexity is Low) and (Regularity is Low) and (Reuse_frequency is Low) then (Reusability is Verylow) (1)
5. If (Coupling is High) and (Volume is High) and (Complexity is High) and (Regularity is Medium) and (Reuse_frequency is Low) then (Reusability is Verylow) (1)
6. If (Coupling is High) and (Volume is High) and (Complexity is Low) and (Regularity is Medium) and (Reuse_frequency is Low) then (Reusability is Verylow) (1)
7. If (Coupling is High) and (Volume is High) and (Complexity is Low) and (Regularity is Medium) and (Reuse_frequency is Low) then (Reusability is Verylow) (1)
8. If (Coupling is High) and (Volume is Medium) and (Complexity is Medium) and (Regularity is Medium) and (Reuse_frequency is Low) then (Reusability is Low) (1)
9. If (Coupling is High) and (Volume is Low) and (Complexity is Medium) and (Regularity is Medium) and (Reuse_frequency is Low) then (Reusability is Low) (1)
10. If (Coupling is High) and (Volume is Medium) and (Complexity is Low) and (Regularity is Medium) and (Reuse_frequency is Low) then (Reusability is Low) (1)
11. If (Coupling is Medium) and (Volume is High) and (Complexity is Medium) and (Regularity is Medium) and (Reuse_frequency is Low) then (Reusability is Low) (1)

Figure 7: Rules in Mamdani model

Sugeno-type systems support this type of model. In general, Sugeno-type systems can be used to model any inference system in which the output membership functions are either linear or constant.A typical rule in a Sugeno fuzzy model has the form, "If Input 1 = x and Input 2 = y, then Output is z = ax + by +c". For a zero-order Sugeno model, the output level z is a constant (a=b =0). The output level z_i of each rule is weighted by the firing strength w_i of the rule. For example, for an AND rule with Input 1 = x and Input 2 = y, the firing strength is w_i = AndMethod ($F_1(x)$, $F_2(y)$), where $F_{1,2}$ (.) are the membership functions for Inputs 1 and 2. The final output of the system is the weighted average of all rule outputs, computed as

$$\text{Final Output} = \frac{\sum_{i=1}^{N} w_i z_i}{\sum_{i=1}^{N} w_i}$$

Figures 8, 9, 10, 11 and 12 show fuzzy sets taken for each input variable in Sugeno model in this case.

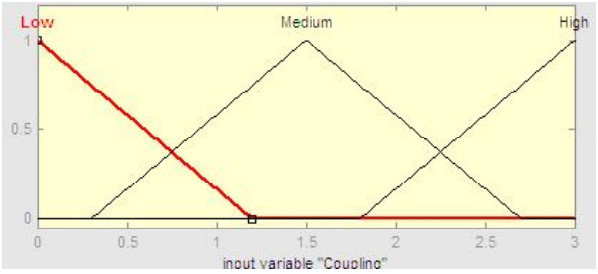

Figure 8: Input variable "coupling" fuzzy sets in Sugeno model

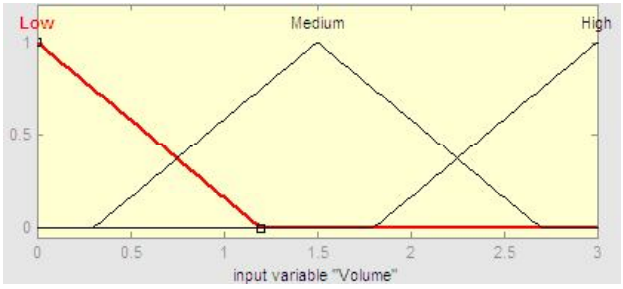

Figure 9: Input variable "volume" fuzzy sets in Sugeno model

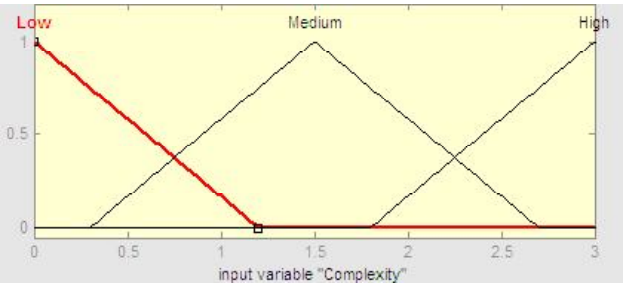

Figure 10: Input variable “complexity” fuzzy sets in Sugeno model

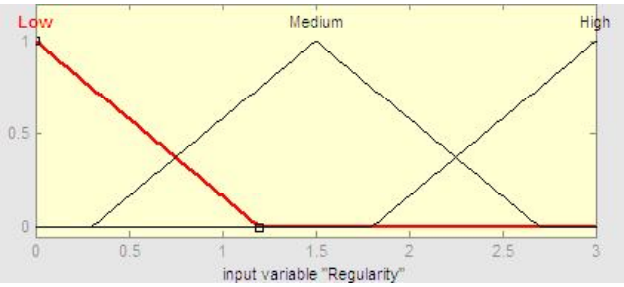

Figure 11: Input variable “regularity” fuzzy sets in Sugeno model

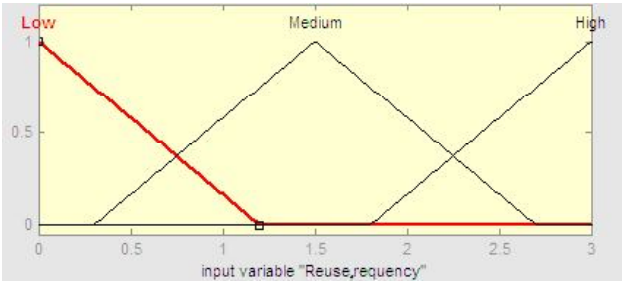

Figure 12: Input variable “reuse frequency” fuzzy sets in Sugeno model

Figure 13 below shows membership function editor for output variable in Sugeno model in this case.

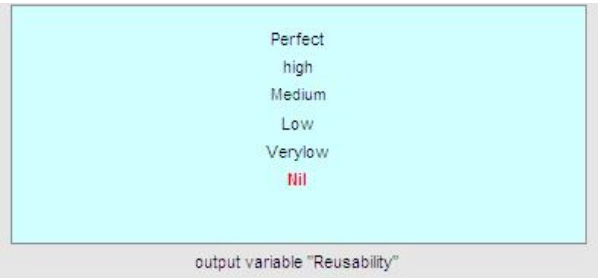

Figure 13: Membership function editor for output variable in Sugeno model

Figure 14 below shows neural network providing support to the fuzzy rule base in Sugeno model. This is the essence of Neuro-fuzzy model. The neural network as shown below has been trained and tested for the ideal data as shown in table 1 for software reusability.

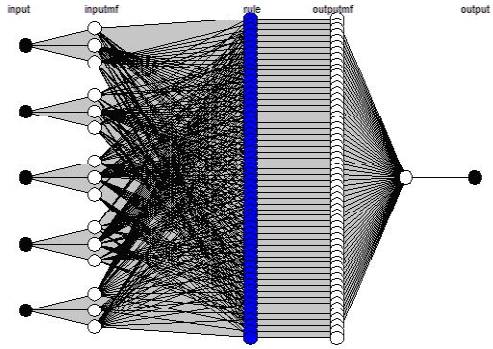

Figure 14: ANFIS model structure for Neuro-fuzzy in Sugeno

IV. Simulation and testing

The developed Neuro-fuzzy model for software reusability in Sugeno has been simulated in MATLAB. It has been tested for various input entries and error has been calculated in each case. Figure 16 below gives test run of developed Neuro-fuzzy model for software reusability for one sample case.

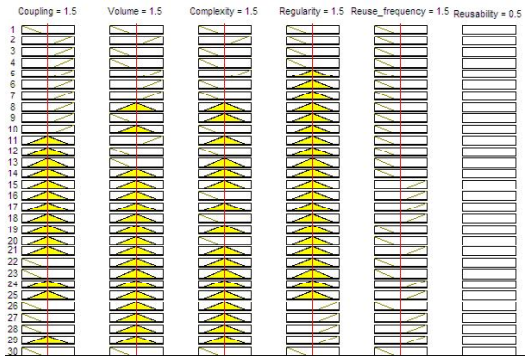

Figure 16: Sample testing of Neuro-fuzzy model (Sugeno model) for software reusability

V. Results and discussions

Figure 18 below shows surface for the developed Neuro-fuzzy model in Sugeno for software reusability in this case**.**

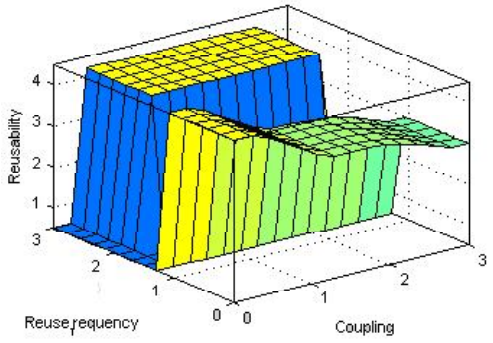

Figure 18: Execution surface in Sugeno model

The table 2 as given below shows fifteen test values for software reusability. Percentage error has been calculated in each test case to discuss accuracy of developed Neuro-fuzzy model.

Table 2: Neuro-fuzzy output

Coupling	Volume	Complexity	Regularity	Reuse-frequency	Reusability	Neuro-fuzzy output	% Error
2	2	2	3	2	1	1.59	0.49
3	2	1	3	1	3	1.59	0.245
2	1	2	1	2	3	1.88	0.02
1	1	3	1	2	3	1.88	0.02
2	1	2	2	1	2	1.88	0.02
2	3	2	2	1	3	2.56	0.12
2	1	2	2	1	3	2.72	0.03
3	2	1	2	2	3	3.10	0.03
2	2	2	2	1	4	3.48	0.105
2	3	2	2	1	4	3.35	0.0375
1	3	2	3	2	6	5.25	0.142
1	3	2	3	3	5	5.25	0.03
2	2	2	2	1	3	3.0	0.03
1	3	2	1	3	4	4.10	0.025
1	3	3	2	3	3	3.10	0.20

VI. Conclusions

I this paper a Neuro-fuzzy hybrid algorithm is proposed for the component classification. In such data search application the design and developed Neuro-fuzzy model has shown its superiority because it includes the advantages of fuzzy as well as neural network. On one hand fuzzy provides a robust inferencing mechanism with no learning and adaptability while on the other hand the neural algorithms provide learning and adaptability. Neuro -fuzzy algorithm is definitely superior to fuzzy algorithm as it inherits adaptability and learning. From the simulation and the result obtained in this paper .it has been shown that the percentage average error is less in the case of neuro-fuzzy algorithms.

VII. References

[1] Y. F. Chen, M. Y. Nishimoto and C. V. Ramamoorty, "The C Information Abstraction System," IEEE Trans. on Software Engineering, Vol. 16, No. 3, March 1990,pp. 471-483.

[2] Caldiera, Gianluigi and Victor R. Basili, "Identifying and Qualifying Reusable Software components," IEEE Software, vol.24, No.2, February 1991, pp-61-70.

[3] J. C. Esteva and R. G. Reynolds, "Identifying Reusable Components using Induction," International Journal of Software Engineering and Knowledge Engineering, Vol. 1, No. 3 (1991) 271-292.

[4] S.R. Chidamber and C.F. Kemerer, "Towards a Metrics Suite for Object Oriented Design," Proc. Conf. Object Oriented Programming Systems, Languages, and Applications (OOPSLA'91), vol. 26, no. 11, pp. 197-211, 1991.

[5] G. Boetticher, K. Srinivas and D. Eichmann, "A Neural Net-based Approach to Software Metrics", Proc. of the 5th International Conference on Software Engineering and Knowledge Engineering, San Francisco, CA, 14-18 June 1993, pp 271-274.

[6] S.R. Chidamber and C.F. Kemerer, "A Metric Suite for Object Oriented Design", IEEE Trans. on Software Engineering , Vol. 20, pp. 476-493,1994.

[7] S. V. Kartalopoulos, Understanding Neural Networks and Fuzzy Logic-Basic Concepts and Applications, IEEE Press, 1996, pp. 153-160.

[8] H. Ishibuchi, T. Yamamoto, and T. Nakashima, Fuzzy data mining: Effect of fuzzy discretization, Proc. of 1st IEEE International Conference on Data Mining, 241- 248, 2001.

[9] T. -P. Hong, C. -S. Kuo, and S. -C. Chi, Trade-off between computation time and number of rules for fuzzy mining from quantitative data, International Journal of Uncertainty, Fuzziness and Knowledge- Based Systems 9, 587-604, 2001.

[10] Richard W. Selby, "Enabling Reuse-Based Software Development of Large-Scale Systems", IEEE Trans. on Software Engineering, Vol. 31, No. 6, June 2005 pp. 495-510.