https://dx.doi.org/10.24001/ijaems.3.9.9 *ISSN: 2454-1311*

Analyzing the Influence of Various Fuzzification Methods in the Evaluation of Netbeans Java Components' Interface Complexity for Reusability

Ajayi Olusola Olajide, Elemese Tolulope Olawale, Aderele Tolulope Busayo

Department of Computer Science, Faculty of Science, Adekunle Ajasin University, Akungba-Akoko, Ondo State, Nigeria

Abstract— *The prognostic nature of fuzzy has made it a versatile tool in handling uncertainty problem. One of the major components of fuzzy system that plays an important role in its successful interpretability is fuzzification. While many researches have utilized its different forms in the accomplishment of their evaluations, especially in the domain of component based software development; it remains to be seen, the application and effects of these different membership functions in the assessment of components a singular solution. The research work examined the interface complexity of two NetBeans Java Components in determining their reusability. The result of the experimentation carried using MATLAB as tool, shows that Trapezoidal returned the highest reusability value, indicating that the components are reusable, and Polynomial fuzzification method returning the lowest reusability value and giving a false alarm that the used components were not reusable. The results underline the indispensable role of fuzzification method in the evaluation of component reusability.*

Keywords— fuzzification, reusability, interface complexity, Java, NetBeans, components, inference engine, membership function.

I. INTRODUCTION

Software reusability is software quality attribute in which software or its module is reused with little or no modification. Software reuse is the development of existing software system using their existing features or component. Since software demand has increase rapidly over the years, with software developer being unable to meet the demand. According to Sommerville (2011), this is due to the increasingly demand for large and more complex system that need to be delivered more quickly. Therefore, the goal of software reusability is to provide higher quality products, less development time, higher scheduling accuracy and Reliability (Kumar et al, 2014).To help the designer and developer to achieve this goals, researchers have proposed a large number methods in the evaluation of component reusability. With vast amount of metrics available it becomes important to understand them in order to acquire a precise and precise understanding of the software being evaluated.

Software Development process contains various phases during the development of software entity. In component based systems development (CBSD), the reusability of a component is an important aspect, which gives the assessment to reuse the existing developed component. If an existing component is used after proper assessment, it reduced the risk, time and cost of the project development process. To be able to reuse the components, it is necessary to predict or asses the component reusability with better accuracy. After assessment of component, if component reusability does not comes out to meet the expected requirement then it may not be good to reuse the reuse the component as it can lead to overwork and may increase the risk, integration time and cost (Sharma et al, 2013). Due to these requirements in software development process, researchers have been trying to find the component reusability using statistical and other conventional techniques. Recently many techniques such as fuzzy logic, Neuro-fuzzy have taken lead due to their power of predictability.

In this paper, various fuzzification methods (fuzzy logic) will be applied in the evaluation of java components reusability with the intent of comparing the effects of the output values returns.

II. BACKGROUND

Fuzzy Logic is the main constitute of the soft computing approaches. Fuzzy logic can be used in conjunction with neural networks, genetic algorithms, probabilistic reasoning

https://dx.doi.org/10.24001/ijaems.3.9.9 *ISSN: 2454-1311*

etc. Fuzzy Logic is a mathematical tool for dealing with uncertainty and also it provides a techniques to deal with imprecision and information granularity (Sivanandam, et al, 2007). Fuzzy logic offers a particularly convenient way to generate a mapping between input and output spaces by using natural expressions (Zadeh, 2002). In direct contrast to neural networks, which take training data and generate opaque models, fuzzy logic is based on if-then rules, which are designed by considering the opinion of experts from that domain. It has been found that the most accurate prediction models are based on analogy and experts opinion. Expert-based estimation was also found to be better than all regression-based models (Musilek et al, 2000). Henceforth the use of fuzzy logic in reusability prediction is desirable since expert knowledge can be incorporated into the fuzzy reusability prediction models.

The major advantage of fuzzy approach is that it is less dependent on historical data. Fuzzy logic models can be constructed without any data or with little data. This makes fuzzy logic superior over data-driven model building approaches such as neural network. Also fuzzy-logic can adapt to new environment when data become available. The most important thing to realize about fuzzy logical reasoning is the fact that it is a superset of standard Boolean logic. Figure 1 shows the general architecture of fuzzy inference system.

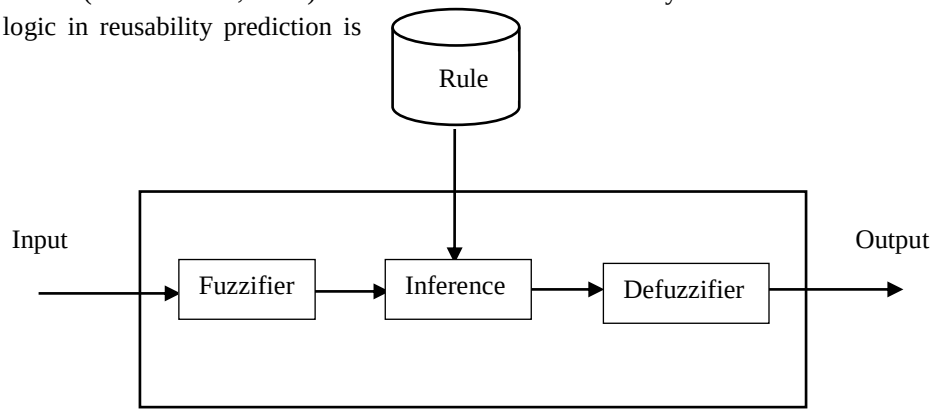

Fig.1: Fuzzy Inference System Architecture

III. RELATED WORKS

In the work of Touil et al (2013),the work focus on the performance and studies of single machine infinite bus using fuzzy power system stabilizer (FPSS), So also in Singh (2015), the paper explores the various metric that affects the reusability of aspect oriented software and estimates it using fuzzy logic approach, the work proved that application of fuzzy logic approach has shown their applicability other than traditional statistical techniques. A furtherance of the work Pooja et al (2015) proposed a model developed using fuzzy logic, the model is effectively used for predicting the degree of reusability of a class in the early phase of SDLC which will result in minimizing the time and effort of the software developers. In Omar et al (2015) the work shows comparison between the effects of different type of membership function on fuzzy logic controller and presents the performance comparison of fuzzy logic controller with three different types of membership functions.

IV. RESEARCH PROBLEM

Fuzzy by nature and purpose has different and wide applicability. The breakdown of the architecture shows the fuzzifier having different types. Researches deploying different fuzzifiers have shown a particular effect as it relates to the fuzzifier applied (Galetakis et al, 2005; Hajighorbani et al, 2014). This study therefore seeks to demonstrate and analyze the effects of the various fuzzification methods in one singular experimentation.

RESEARCH AIM

The study aims to analyze the influence of the various fuzzification methods on the evaluation of NetBeans Java components' interface complexity for determining reusability.

V. RESEARCH METHODOLOGY

The following procedures were taken in the gathering of data, analyzing and implementation in order to actualize the study's goal.

1. Java components were extracted from NetBeans. The features extracted include component's number of methods, number of properties. A total of two (2)

https://dx.doi.org/10.24001/ijaems.3.9.9 *ISSN: 2454-1311*

components, namely AdvancedMedia and HtmlEditorApp were accessed for the experimentation of this work.

2. Suitable Interface Complexity metrics were adopted and applied for the assessment of the components' reusability.
3. Fuzzy Logic. The evaluation was done with the different fuzzification methods identified in this study.Mamdani FIS type was used for this study.

Figure 2 shows the research methodology

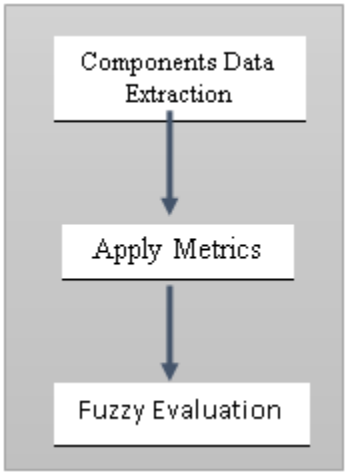

Fig.2: Research Methodology

RESEARCH DESIGN

Architectural design is a representation that enables a software engineer to analyze the effectiveness of the design in meeting its stated requirements (Pressman, 2001). Figure 3 presents the architectural design of the system.

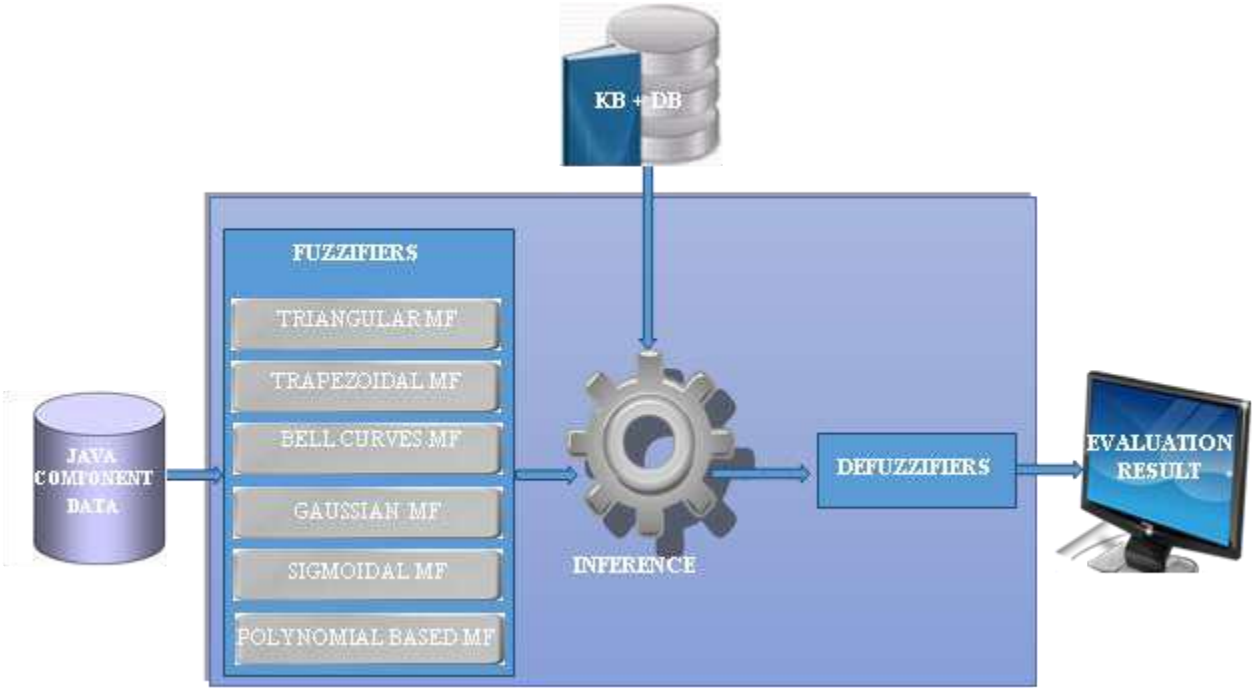

Fig.3: Architectural Design

Fuzzification

https://dx.doi.org/10.24001/ijaems.3.9.9 *ISSN: 2454-1311*

According to Naaz et al (2011), Fuzzification involves mapping the values of the numerical inputs by a function according to a degree of compatibility of the respective fuzzy sets. It could be described as the conversion of a precise quantity to a fuzzy quantity.

Membership Function

A Membership Function (MF) is a curve that defines how each point in the input space is mapped to a membership value (or degree of membership) between 0 and 1.In fuzzy logic, fuzzy set membership occurs by degree over the range [0,1], which is represented by a membership function. There are different types of membership functions, namely:

i. Triangular: This is specified by three (3) parameters; it is curve and linear (a straight line). It has the function name trimf.
Equation 1 presents the mathematical model for triangular membership type.

$$f(x;a,b,c) = \max\left(\min\left(\frac{x-a}{b-a}, \frac{c-x}{c-b}\right), 0\right) \quad \ldots(1) \quad \text{(Zhao et al, 2002)}$$

where:
a, b, c represent values of membership functions low, medium and high respectively.
the output (f) ranges between 0 and 1.

ii. Trapezoidal: This is specified by four (4) parameters; it is curve and linear (also, straight line, but truncated triangular curve). It has the function name trapmf.
Equation 2 present the mathematical model for trapezoidal membership type.

$$f(x;a,b,c) = \max\left(\min\left(\frac{x-a}{b-a}, \frac{d-x}{d-c}\right), 0\right) \quad \ldots(2) \quad \text{(Zhao et al, 2002)}$$

The parameters *a* and *d* locate the "feet" of the trapezoid and the parameters *b* and *c* locate the "shoulders."

iii. Bell curves: This is specified by three (3) parameters; it is smooth and non-linear. It has the function name gbellmf.
Equation 3 present the mathematical model for Bell Curve membership type.

$$f(x;a,b,c) = \frac{1}{1+\left|\frac{x-c}{a}\right|^{2b}} \quad \ldots(3) \quad \text{(Zhao et al, 2002)}$$

The parameter *b* is usually positive. The parameter *c* locates the center of the curve. Enter the parameter vector params, the second argument for gbellmf, as the vector whose entries are *a*, *b*, and *c*, respectively.

iv. Gaussian: This is specified by two (2) parameters; it is smooth and non-linear. It has two (2) functions: gaussmf and gauss2mf.
Equation 4 present the mathematical model for Gaussian membership type.

$$f(x;\sigma,c) = e^{\frac{-(x-c)^2}{2\sigma^2}} \quad \ldots(4) \quad \text{(Zhao et al, 2002)}$$

The parameters for gaussmf represent the parameters σ and *c* listed in order in the vector [sig c].

v. Sigmoidal: This is specified by two (2) parameters; it is suitable for use in neural network for simulating the behaviour of fuzzy. It has three (3) functions type: the basic sigmoidal function (sigmf), the difference in two sigmoidal functions (dsigmf) and the product of two sigmoidal functions (psigmf).
Equation 5 present the mathematical model for Sigmoidal membership type

$$f(x,a,c) = \frac{1}{1+e^{-a(x-c)}} \quad \ldots(5) \quad \text{(Zhao et al, 2002)}$$

Depending on the sign of the parameter *a*, the sigmoidal membership function is inherently open to the right or to the left, and thus is appropriate for representing concepts such as "very large" or "very negative." More conventional-looking membership functions can be built by taking either the product or difference of two different sigmoidal membership functions.

vi. Polynomial based: Several membership functions are found under this group. Three (3) commonly related functions are: Z curve (zmf), S curve (smf) and Pi curve (pimf).

https://dx.doi.org/10.24001/ijaems.3.9.9 *ISSN: 2454-1311*

Equations 6 to 8 present the mathematical model for Polynomial Base membership type.

pimf

$$f(x;a,b,c,d)=\begin{cases}0, & x\le a\\ 2\left(\frac{x-a}{b-a}\right)^2, & a\le x\le\frac{a+b}{2}\\ 1-2\left(\frac{x-a}{b-a}\right)^2, & \frac{a+b}{2}\le x\le b\\ 1, & b\le x\le c\\ 1-2\left(\frac{x-c}{d-c}\right)^2, & c\le x\le\frac{c+d}{2}\\ 2\left(\frac{x-d}{d-c}\right)^2, & \frac{c+d}{2}\le x\le d\\ 0, & x\le d\end{cases} \quad \ldots(6) \quad \text{(Zhao et al, 2002)}$$

Zmf

$$f(x;a,b)=\begin{cases}1, & x\le a\\ 1-2\left(\frac{x-a}{b-a}\right)^2, & a\le x\le\frac{a+b}{2}\\ 2\left(\frac{x-b}{b-a}\right)^2, & \frac{a+b}{2}\le x\le b\\ 0, & x\ge b\end{cases} \quad \ldots(7) \quad \text{(Zhao et al, 2002)}$$

Smf

$$f(x;a,b)=\begin{cases}0, & x\le a\\ 2\left(\frac{x-a}{b-a}\right)^2, & a\le x\le\frac{a+b}{2}\\ 1-2\left(\frac{x-b}{b-a}\right)^2, & \frac{a+b}{2}\le x\le b\\ 1, & x\ge b\end{cases} \quad \ldots(8) \quad \text{(Zhao et al, 2002)}$$

Components Data

With established facts that components could be purchased and extracted from third party rather than built (Sharma et al, 2006; Sharma et al, 2009; Bharwaj, 2010; Kumar et al, 2013), we extracted two (2) components from NetBeans. Table 1 shows the sources, nature and numbers of the components extracted.

Table.1: Components Used

S/N	Component Source	Nature of Components	Number of Components	Date Extracted
1.	NetBeans	Java Components	2	01/09/2016

Interface Complexity

Interfaces are the access points of a component, through which a component can request a service declared in an interface of the service providing component (Kaur and Singh, 2013). That is, interface acts as the means through which application and components interact. Sagar et al (2010) submitted that, interface plays a lead role while measuring the overall complexity of the component. Sharma et al (2008) said complex interfaces will lead to high efforts for understanding and customizing of components; therefore for better reusability, interface complexity should be as low as possible.

For this study, Bounded Interface Complexity Metric (BICM) has been adopted from Tobias et al (2015) for the measurement of black-box complexity based on the specification/signature of the component under study.

https://dx.doi.org/10.24001/ijaems.3.9.9 *ISSN: 2454-1311*

Interface Complexity (IC) =

$$(A \sum_{i=1}^{m}(CIMi)/M)) + (B \sum_{j=1}^{n}(CIPj)/N)) \quad \ldots(9)$$

(Tobias et al, 2015)

where:

CIM_i is the complexity of the i^{th} interface method

CIP_j is the complexity of the j^{th} property.

M and N represents the number of component methods and properties respectively, while A, B are the weight values.

In this study, the customization constants, A and B are equated to 1; thereby modifying the equation to be:

$$IC = (\sum_{i=1}^{m}(CIMi)/M)) + (\sum_{j=1}^{n}(CIPj)/N)) \quad \ldots(10)$$

In determining the interface complexity of components, different weight values are assigned to methods, based on the data type of arguments or return values (e.g. integer, string, date, array list, vector etc.) used in the method. Adopting Kumar et al (2014) the data collected were classified into five (5), namely: very simple, simple, medium, complex, and highly complex. Table 2 represents the weight values of the interface methods.

Table 2: Weight Values of Interface Methods (Kumar et al, 2014)

No of Method / No of Data Types	Very simple (e.g. integer, double, Boolean, float)	Simple (e.g. structured data type)	Medium (e.g. class type, object type)	Complex (e.g. pointers, built-in data types)	Highly Complex (e.g. user-defined data types)
1-3	0.05	0.10	0.15	0.20	0.25
4-6	0.10	0.20	0.30	0.40	0.50
7-9	0.15	0.30	0.45	0.60	0.75
>=10	0.20	0.40	0.60	0.80	1.00

To generate the complexity table for our proposed system, the weight value in Table 2 and the feature extracted values in Table 1 were used. The equations below show how Table 3 was generated.

Let

M = No of Method

P = No of Property

w = weight value

m = No of method's data type

p = No of property's data type

n = number of components

So,

$$CIM = \sum_{i=1}^{n}(m * w)/M \quad \ldots(11)$$

S/N	Component Name	No of methods (M)	Weight values	CIM = ∑ No method data type * their weight values	A = CIM/M	No of property (P)	Weight values	CIP = ∑ No property data type * their weight values	B = CIP/P	IC = A+ B
1.	AdvancedMedia	13	0.10, 0.45	0.32	0.02	3	0.15	0.15	0.05	0.07
2.	HTMLEditorApp	14	0.10, 0.45	0.33	0.02	1	0.15	0.15	0.15	0.17

https://dx.doi.org/10.24001/ijaems.3.9.9 *ISSN: 2454-1311*

$$CIP = \sum_{i=1}^{n}(p * w)/P \quad \ldots(12)$$

Table.3: Complexity Table

VI. IMPLEMENTATION

MATLAB software was used as tool for the implementation of this work. The Input Variable named IC (Interface Complexity) has three input linguistic variables Low (0,0.25,0.50), Medium (0.25,0.50,0.75) and High (0.50,0.75,1.0) while the output variable, Reusability has also three variables as its output linguistic, namely Low (0,0.25,0.50), Medium (0.25,0.50,0.75) and High (0.50,0.75,1.0). With one (1) input variable for the experimentation and three (3) linguistic values, we have 3^1 rules (3 rules) generated. These were formulated as:

If (IC is Low) then (Reusability is High) (1)
If (IC is Medium) then (Reusability is Medium) (1)
If (IC is High) then (Reusability is Low) (1)

For cost effectiveness, Mamdani FIS type was used for this work, with different fuzzification methods deployed for each experimental setup. Table 4 shows the FIS structure for the study.

Table.4: FIS Structure

[System]
Name='IC'
Type='mamdani'
Version=2.0
NumInputs=1
NumOutputs=1
NumRules=3
AndMethod='min'
OrMethod='max'
ImpMethod='min'
AggMethod='max'
DefuzzMethod='centroid'

Figures 4 to 7 further show the FIS specifications for the some of the used fuzzification methods (Input and Output).

Fig.4: Triangular Membership Specification (Input)

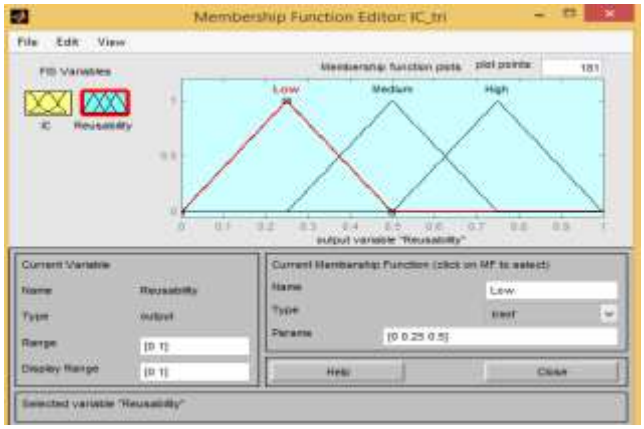

Fig.5: Triangular Membership Specification (Output)

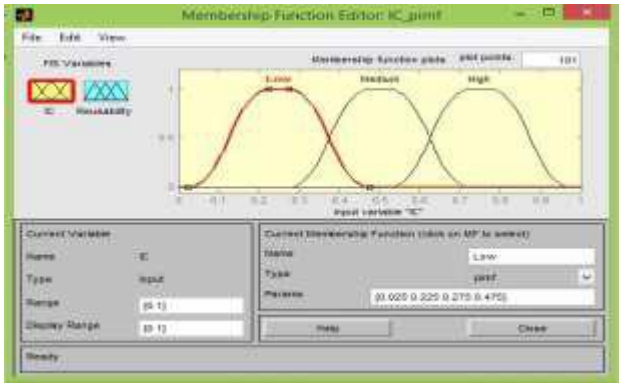

Fig.6: Triangular Membership Specification (Input)

https://dx.doi.org/10.24001/ijaems.3.9.9 *ISSN: 2454-1311*

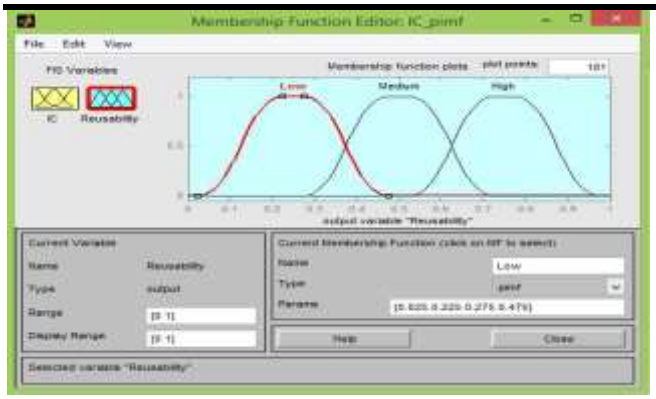

Fig.7: Triangular Membership Specification (Output)

The reusability outputs of some of the different fuzzification methods are presented in Figures 8 to 19, while Table 5 shows the entire results according to the fuzzification methods applied.

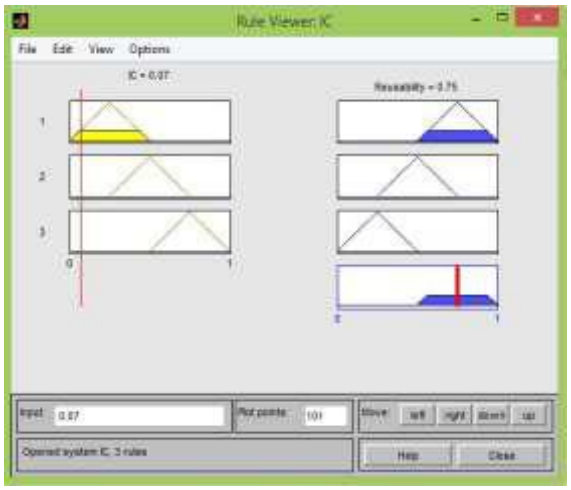

Fig.8: IC Reusability Results (Triangular– Component1)

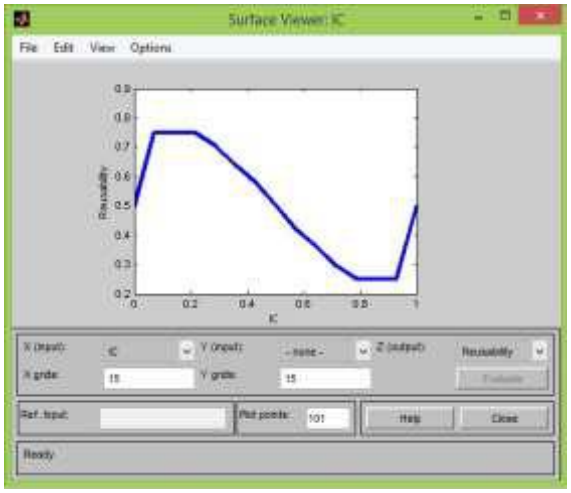

Fig.9: IC Reusability Chart (Triangular– Component1)

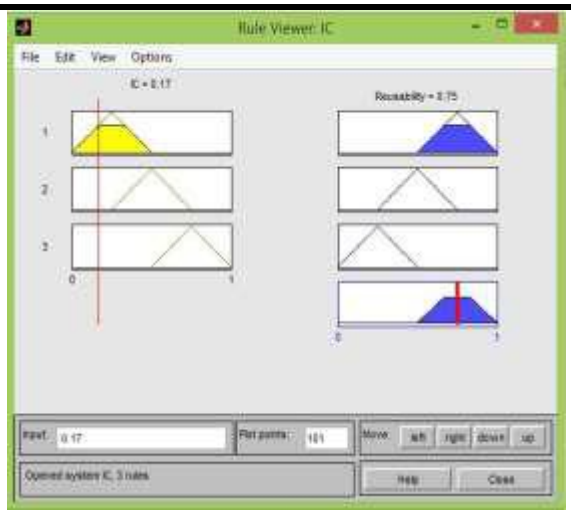

Fig.10: IC Reusability Results (Triangular– Component2)

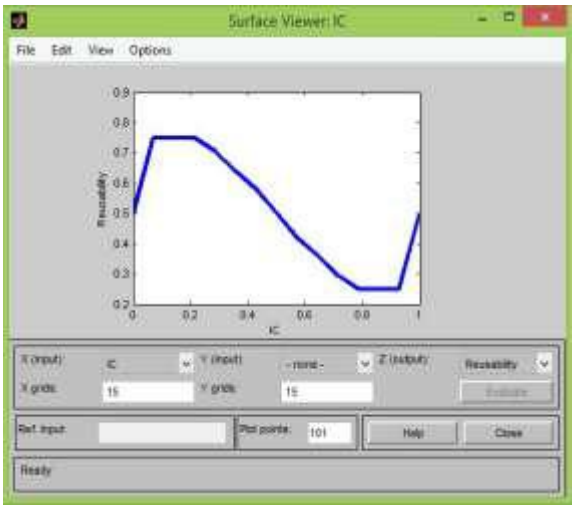

Fig.11: IC Reusability Chart (Triangular– Component2)

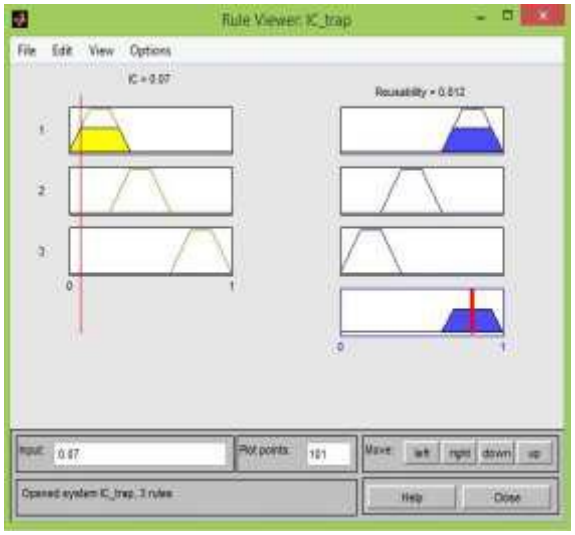

Fig.12: IC Reusability Results (Trapezoidal– Component1)

https://dx.doi.org/10.24001/ijaems.3.9.9 *ISSN: 2454-1311*

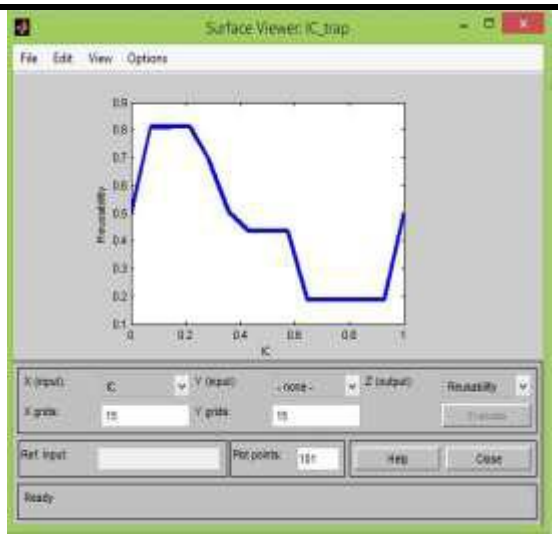

Fig.13: IC Reusability Chart (Trapezoidal– Component1)

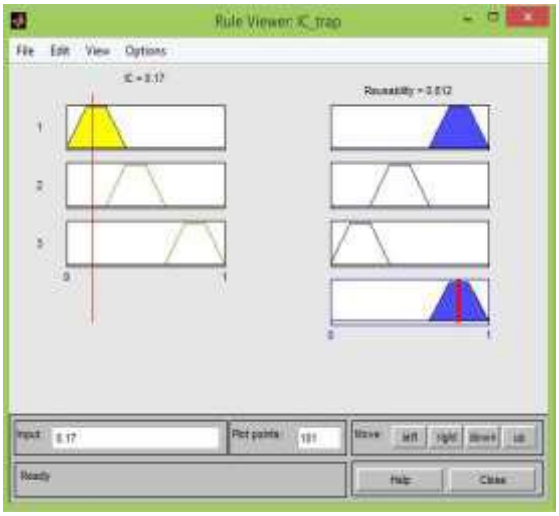

Fig.14: IC Reusability Results (Trapezoidal– Component2)

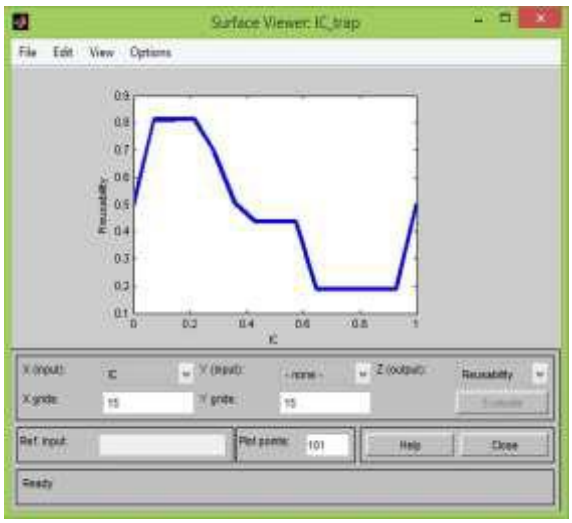

Fig.15: IC Reusability Chart (Trapezoidal– Component2)

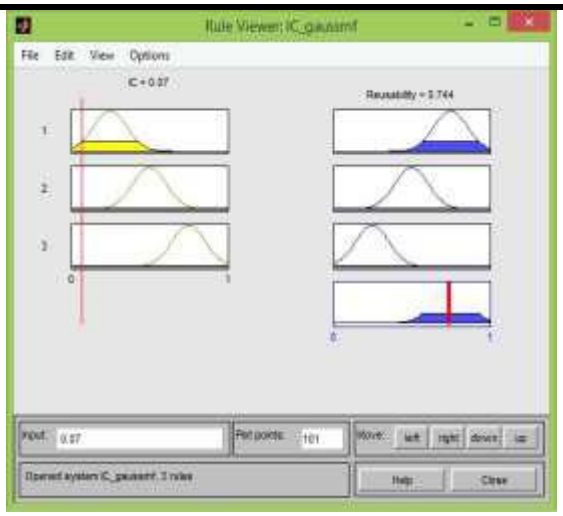

Fig.16: IC Reusability Results (Gaussian– Component1)

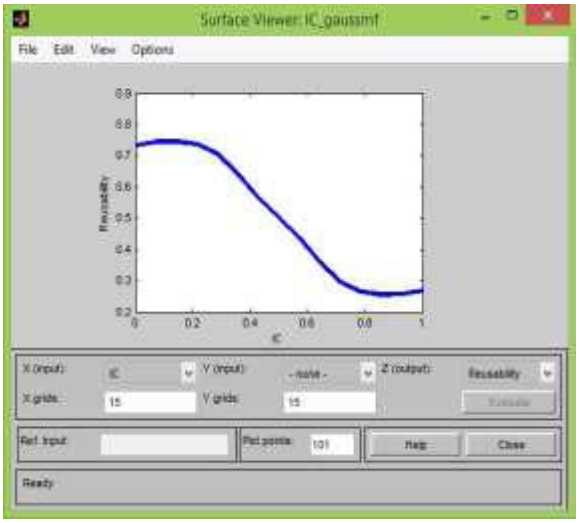

Fig.17: IC Reusability Chart (Gaussian– Component1)

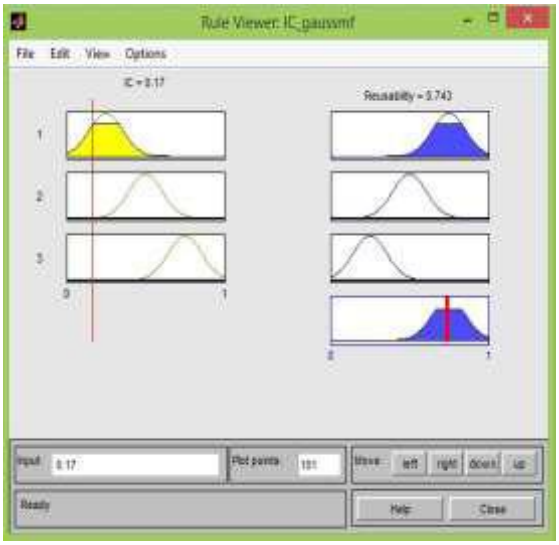

Fig.18: IC Reusability Results (Gaussian– Component2)

https://dx.doi.org/10.24001/ijaems.3.9.9 *ISSN: 2454-1311*

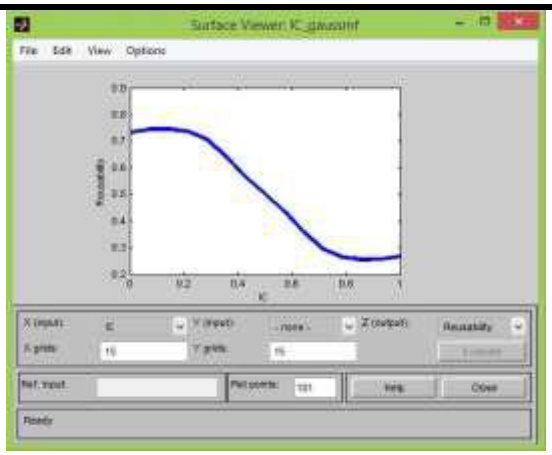

Fig.19: IC Reusability Chart (Gaussian– Component2)

Table.5: Fuzzification Methods and their Interface Complexity Reusability Outputs

Fuzzification Method	Component Type	Input Value	Reusability
Triangular	Component I	0.07	0.750
	Component II	0.17	0.750
Trapezoidal	Component I	0.07	0.812
	Component II	0.17	0.812
Bell Curves	Component I	0.07	0.736
	Component II	0.17	0.744
Gaussian	Component I	0.07	0.744
	Component II	0.17	0.743
Sigmoidal	Component I	0.07	0.758
	Component II	0.17	0.788
Polynomial Zmf	Component I	0.07	0.436
	Component II	0.17	0.436
Polynomial Smf	Component I	0.07	0.780
	Component II	0.17	0.808
Polynomial Pimf	Component I	0.07	0.750
	Component II	0.17	0.750

Highest Reusability Prediction Value

Lowest Reusability Prediction Value

VII. DISCUSSION

Table 5 shows the influences of the different fuzzification approach in the prediction of component reusability based on their interface complexity. As can be deduced from the table, Trapezoidal fuzzification method produced the highest reusability prediction, while Polynomial membership type shows the least reusability value. This implies that while other fuzzification methods resulted into considerably high reusability values, indicating that the components are reusable; the results of Polynomial Zmf suggests components not reusable as it yielded low reusability values.

VIII. CONCLUSION

Certain quality factors such as customizability, interface complexity, portability, understandability etc. determine the reusability of software components. These quality factors are computed using appropriate and related metrics. In this work, we have illustrated with only Interface Complexity. To further stress on the necessary caution to be deployed in the assessment of component reusability, this study has argued for the role of fuzzification methods in ascertaining component reusability. The results presented in the section above proved that indeed, fuzzification method used has effects on the predictability of component reusability.

https://dx.doi.org/10.24001/ijaems.3.9.9 *ISSN: 2454-1311*

FUTURE RESEARCH DIRECTION

Having established the fact that, aside interface complexity, which serves as access gateway to application usage, other quality factors like customizability, portability, understandability do determine reusability of components, this work shall be extended further to analyze the effects of different fuzzification methods utilizing these identified factors in one experimental scenario. We shall also endeavor to use more number of components.

REFERENCES

[1] Galetakis, M., Vasiliou, A., Roumpos, P. F. (2005). Developing Fuzzy Inference System (FIS) for the evaluation of multiple layer Liguite deposits, International Workshop in "Geoenvironment and Geotechnics", Milos Island, Greece.

[2] Hajighorbami,S., Radzi, M.A.M., Abkadir, M.Z.A., Shatie, S., Ichanaku, R., and Maghami, M.R. (2014). Evaluation of FuzzyLogic Subsets Effects on Maximum Power Point Tracking for PhotoVoltaic System. International Journal of Photoenergy. http://dx.doi.org/10.1155/2014/719126

[3] Kumar, A., Chaudhary, D., and Kumar, A. (2014). Empirical Evaluation of Software Component Metrics. International Journal of Scientific & Engineering Research. Vol. 5, Issue 5: 814- 820

[4] Kumar, V., Sharma A.and Kumar. R. (2013). Applying Soft Computing Approaches to Predict Defect Density in Software Product Releases: An Empirical Study, COMPUTING AND INFORMATICS, volume 32, No.1, pp: 203-224.

[5] Musilek, P., Pedrycz, W., Succi, G., and Reformat, M. (2000). Software Cost Estimation with Fuzzy Models, ACM SIGAPP Applied Computing Review, Vol. 8, pp.24-29.

[6] Naaz, S., Alam A., and Biswas, R. (2011). Effect of Different Deffuzification Methods In A Fuzzy Based Load Balancing Application IJCSI International Journal of Computer science Issues, Vol. 8, Issue 5, No 1, ISSN (online): 1694-0814

[7] OmarAdil,M.A., Aous Y. A., Balasem S. S.(2015). Comparison between the Effects of Different Types of Membership Functions on Fuzzy Logic Controller Performance.International Journal of Emerging Engineering Research and Technology Volume 3, Issue 3, March 2015, PP 76-83 ISSN 2349-4395 (Print) & ISSN 2349-4409 (Online)

[8] Pooja, D., Parwinder, K. D., Jagmohan, M. (2015). Estimating Software Reusability from OO Metrics using Fuzzy Logic. Apeejay Journal of Computer Science and Applications. ISSN: 0974-5742(P), Vol. 3, January 2015

[9] Pradeep, K. S., Om, P. S.,Amar, P. S., and Amrendra, P.(2015). A Framework for Assessing the Software Reusability using Fuzzy Logic Approach for Aspect Oriented Software. Information Technology and Computer Science, 2015, 02, 12-20 Published Online January 2015 in MECS (http://www.mecs-press.org/) DOI: 10.5815/ijitcs.2015.02.02

[10] Roger, S. P.(2001). Software Engineering, A Practitioner approach. McGraw Hill, NY.

[11] Sagar, S., Nerurkar N. W., Sharma A. (2010). A soft computing based approach to estimate reusability of software components, ACM SIGSOFT Software Engineering Notes, Volume 35 Issue 5, September pp:1-5.

[12] Sharma, A., Kumar, V., and Kumar, R. (2013). Applying Soft Computing Approaches to Predict Defect Density in Software Product Releases: An Empirical Study, COMPUTING AND INFORMATICS, volume 32, No.1, pg: 203-224.

[13] Sharma, A., Kumar, R., and Grover, P. S. (2009). Reusability assessment for software components, ACM SIGSOFT Software Engineering Notes, Volume 34 Issue 2, March, pp: 1-6.

[14] Sharma, A., Kumar, R., and Grover, P. S. (2008). Empirical Evaluation of Complexity for Software Components, International Journal of Software Engineering and Knowledge Engineering (IJSEKE), Vol. 18, Issue 5, pp: 519-530.

[15] Sivanandam, S. N., Sumathi, S., Deepa, S. N. (2007). Introduction to fuzzy logic using MATLAB, Springer. New York

[16] Slimane, T., and Djilani, B. A.(2013). Effect of Different Membership Functions on Fuzzy Power System Stabilizer for Synchronous Machine Connected to Infinite Bus. International Journal of Computer Applications (0975 – 8887) Volume 71– No.7

[17] Sommerville, I.(2011). Software Engineering, Pearson, US.

[18] Tobias, M.,Mwangi W., and Michael, K. (2015). Empirical Evaluation of Complexity Metrics for Component Based Systems, Journal of Theoretical and Applied Information Technology. ISSN 1992-8645, Vol. 73 No 2.

[19] Zadeh, L. A.(2002). From Computing with numbers to computing with words-from manipulation of measurements to manipulation of perceptions, International Journal of Applied Mathematics and Computer Science, Vol.12, Issue 3, pp: 307-324