African Journal of Mathematics and Computer Science Research Vol. 2(6), pp. 092-097, July, 2009
Available online at http://www.academicjournals.org/AJMCSR

Full Length Research Paper

Graphics to fuzzy elements in appraisal of an in-house software based on Inter-failure data analysis

J. O. Omolehin[1]*, K. Rauf[1], R. G. Jimoh[2] and O. C. Abikoye[2]

[1]Department of Mathematics, University of Ilorin, Ilorin, Nigeria
[2]Department of Computer Science, University of Ilorin, Ilorin, Nigeria.

Accepted 18 May, 2009

The performance of any software can be measured, approximately, by three parameters; Reliability, Availability and Maintainability. They provide information about the robustness of the software under consideration. In this work, a software is developed to computerize University of Ilorin student's results and is designated as an IN-HOUSE Software. The formalism of fuzzy logic is used to investigate its performance. Our results show that the metrics for performance testing by Pfleeger (1997) perform better when the raw data is refined (fuzzified).

Key words: Software, graphics, fuzzy, decision table, center of area.

INTRODUCTION

Software performance evaluation has been of great concern to software engineers since it takes some degree of expertise to determine whether a software is of good performance regardless of whether it is operational or not, see Leach (2000) and Jawadekar (2004).

There are many ways of testing the system which include function, performance, acceptance and installation testing, see Land (2003), Rankin (2002) and Sommerville (1992).

Fuzzy system

Fuzzy sets were proposed to deal with vagueness related to the way people sense things (e.g. tall versus short, big versus small). A set is defined by its elements and the membership of each element in the set, Sugeno (1985). Fuzzy logic is an area of research, which provides solutions to the problems of vagueness which departs from the all or nothing logic. It logically redefines yes or no ideas in proper form by Zadeh (1965).

This logic constitutes the basis for linguistic approach. Under this approach, variables can assume linguistic values. Each linguistic value is characterized by a label and a meaning. This label is a sentence of a language. The meaning is a fuzzy subset of a universe of discourse. Models, based on this approach, can be constructed to stimulate approximate reasoning. The implementation of these present two major problems namely: how to associate a label with an unlabelled fuzzy set on the basis of semantic similarity (linguistic approximation) and how to perform arithmetic operation with fuzzy numbers.

Zadeh (1965) had distinguished two main directions in fuzzy logic, one is older, better known, heavily applied but does not ask deep logical questions and serves mainly as apparatus for fuzzy control, analysis of vagueness in natural language, control machine, fuzzy traffic controller, fuzzy aggregator and so on. It is one of the techniques of computing. Soft computing is a computational method that is tolerant to sub-optimality, impreciseness and vagueness etc giving quick, simple and sufficient good solutions, Zimmermann et al. (1993).

Fuzzy logic in the narrow sense is symbolic with comparative notion of truth developed fully in spirit of classical logic (syntax, semantics, truth preserving deduction, completeness, etc.) both prepositional and predicate logic. It is a branch of many-valued logic based on the paradigm of inference under vagueness, Zimmermann (1987a).

Fuzzy Transformation

Fuzzy systems input undergo three transformations:

Fuzzification: This is a process that uses predefined membership functions that maps each system input into one or more degree of membership(s).

Rulebase: Rule (Predefined) is evaluated by combining degrees of membership to form output strengths.

*Corresponding author. E-mail: omolehin_joseph@yahoo.com

Deffuzzification: This is a process that computes system outputs based on strengths and membership functions. The two most popular Deffuzzification methods are the Mean-Of-Maximum (MOM) and the Center of Area (COA) methods. For MOM, the crisp output Δq is the mean value of all points ω_i whose membership values $\mu_c(\omega_i)$ are maximum. In the case of discrete universal set W, MOM is defined as:

$$MOM = \sum_{i=1}^{n} \frac{\omega_i}{n}$$

Where,

$$\{\omega_i \mid \mu_c(\omega_i) \leq \mu_c(\omega_j), \omega_i, \omega_j \in W, \omega_i \neq \omega_j\}$$

and n is the number of such support values. As for COA, the crisp output Δq is the center of gravity of distribution of membership function μ_c. In the case of the discrete universal set W, COA is also defined as in Zimmermann (1987b) by:

$$COA = \frac{\sum_{i=1}^{n}(\mu_i(\omega_i)) * \omega_i}{\sum_{i=1}^{n} \mu_i(\omega_i)}$$

Where n is the number of elements of the fuzzy set C, and ω ε W. In this model, the COA method is used for Deffuzzification.

The logic is not just restricted to just the two categories as illustrated above; it can also be applied to any number of the categories. For example, an element x can belong to set A with membership function a, to set B with membership function b, to set C with membership function c and so on. However, it is important to keep in mind that the sum a, b, c etc should equal unity.

This work is mainly on fuzzy approach to performance evaluation (testing) using Reliability, Availability and Maintainability as the yardsticks for the existing metrics. The focal point, in this work, is to calculate the center of area (COA) of the decision table associated with the inter-failure time recorded in the experiment.

Description of variable

The following are the definitions of the variables used:

Software reliability (R_s): The concept of software reliability can be defined as ability of software system to function consistently and correctly over a long period of time, Pfleeger (1997). Software reliability can also be defined as the probability of a failure-free software operation for a specified period of time in a specified environment. Software Reliability is an important factor affecting system reliability. It is quite different from the hardware reliability since it reflects the design perfection rather than manufacturing production as in the case of hardware reliability. Software reliability is a function of execution/operational time and not the clock time. The measure of reliability reflects the system usage.

Software availability (A_s): Availability of software can be guaranteed when the system is operating successfully according to specification at a given point in time by Pfleeger (1997). It is a function of clock time and not operational time.

Software maintainability (M_s): According to IEEE standard computer dictionary (1990), Maintainability can be defined as the ease and speed with which any maintenance activity can be carried out on an item of equipment. It is also defined as the ease with which a software system or component can be modified to correct faults, improve performance or other attributes, or adapt to a changed environment.

Maintainability which is analogous to cumulative failure time in reliability is the probability that a specified maintenance action on a specified item can be successfully performed (putting the item into a specified state) within a specified characteristic using specified tools and procedures by Rosenberg (2000). This shows that for a given condition of use, a maintenance activity can be carried out within a stated time interval and using a stated procedures and resources. It is noted that the three measurements of software performance operate on a scale between 0 and 1. This implies that the closer the values of the measurements to one, the better the performance and the poorer they are, as they approach zero.

MAIN RESULTS

In this work, we consider an IN-HOUSE result computation software used in computing the result of the Postgraduate Diploma students in the department of Computer Science, University of Ilorin, Ilorin, Nigeria. The software has been developed over six years and has been put into effective usage since then. The most important factor in this work is the system failure which might be catastrophic, critical, marginal or minor. Information about software failure is taken with the following two assumptions about the departmental result computation software which is our case study:

- There is a possibility of causing another problem while solving a particular one.
- The inability to predict the next failure.

The case of West African Examination Council (WAEC) and National Examination Council (NECO) in Nigeria was discussed in Omolehin et al. (2009).

Table 1. Inter-failure time Read from Left to Right:

106	133	219	184	218	112	105	194	215	118
240	152	179	126	210	190	772	222	128	216
132	100	102	104	323	161	272	154	553	395
440	170	257	183	437	295	125	1080	715	130
175	580	852	190	115	900	618	925	702	1225
125	238	203	105	1033	712	402	275	292	185
1080	521	442	1315	2212	452	220	185	1020	800

Table 2. The decision table for the fuzzified inter-failure data

0.0479	0.0601	0.9990	0.0831	0.0985	0.0506	0.0474	0.0877	0.0972	0.0533
0.1085	0.0687	0.0809	0.0569	0.0949	0.0859	0.3448	0.1003	0.0578	0.0976
0.0596	0.0452	0.0461	0.0470	0.1460	0.0728	0.1229	0.0696	0.2499	0.01785
0.1988	0.0768	0.1161	0.0827	0.1975	0.1333	0.0569	0.4880	0.3231	0.0587
0.0791	0.2621	0.3580	0.0859	0.0520	0.4067	0.2793	0.4180	0.3172	0.5535
0.0569	0.1075	0.0917	0.0474	0.4668	0.3217	0.1817	0.1243	0.1319	0.0836
0.4880	0.2354	0.1997	0.5942	0.9995	0.2042	0.0994	0.0836	0.4609	0.3615

Interfailure data

Inter-failure data is a data of successive failures of the departmental result computation of an in-house software for an operational environment over a particular period of time.

The inter-failure data of an in-house application, used in the computation of student's result are taken in terms of execution time (seconds) between successive failure of a command-and-control system during an in-house testing, using a simulation of the real operational environment by Musa (1997).

Reliability, Availability and Maintainability are expressed as the attributes of software measured as numbers between 0 (unreliable, unavailable or unmaintainable) and 1 (completely reliable, always available and completely maintainable).

The average of the failure time $t_1, t_2, t_3, \ldots, t_m$ is the mean time to failure (MTTF). T_t is a random variable representing yet-to-be observed time to failure.

The mean time to maintain (MTTM) is the average time it takes to fix a faulty software component.

The mean time to failure (MTTF) can be combined with mean time to maintain (MTTM) to determine how long the system is unavailable. That is, mean time between failures *MTBF = MTTR + MTTM*.

As the system becomes more reliable, its MTTF increases. We can use MTTF as a measure whose value is near zero when MTTF is small, and nears 1 as MTTF gets increased. Considering this relationship, a measure of reliability function can be defined as: $R_s = MTTF/ (1 + MTTF)$

Similarly, we can measure availability function as to maximize the MTBF as:

$A_s = MTBF (1 – MTBF)$.

Also, maintainability function can be measured to minimize the MTTM as:

$M_s = 1/ (1 + MTTM)$.

Table 1 above shows the input data collected from our experiment in its original form.

Decision Table:
Table 1 above is to be fuzzified with the membership function model presented below:

$$f(x_i) = \begin{cases} 0 & if \quad x < 100 \\ (x_i)/(x_m + 1) & if \quad 100 \le x \le 2212 \\ 1 & if \quad x > 2212 \end{cases}$$

Where x_m is the maximum element in the data x_i is the element in the data to be fuzzified and $i = 0,\ldots,n$, n is the number of elements.

Since the focal point of this work is to use the mean of the original data and the center of gravity of our fuzzified data to calculate our test parameters for comparative purpose, our new decision table is now given above In Table 2.

The Center of Area approach used for our decision table is given as:

$$\text{COA} = \frac{\sum_{i=0}^{n} f(x_i) * x_i}{\sum f(x_i)}$$ and is calculated to be 767.94.

Table 3. Descriptive Statistics.

	N	Minimum	Maximum	Mean	Std. Deviation
Mean time to failure	70	100.00	2212.00	403.8000	386.2255
Valid N (listwise)	70				

Table 4. T-Test (One-Sample Test)

	Test Value = 0					
	t	Df	Sig. (2-tailed)	Mean Difference	95% Confidence Interval of the Difference	
					Lower	Upper
Mean time to maintain	.028	69	.978	1.5143	-105.7221	108.7506

Table 5. Descriptive statistics for Table 2 (Descriptive Statistics).

	N	Minimum	Maximum	Mean	Std. Deviation
Mean time to failure	70	.0149	.9995	.156647	.1582064
Valid N (listwise)	70				

Table 6. T-Test for Table 2(One-Sample Test)

	Test Value = 0					
	t	Df	Sig. (2-tailed)	Mean Difference	95% Confidence Interval of the Difference	
					Lower	Upper
Mean time to Failure	8.284	69	.000	.156647	.118924	.194370

The Center of Area approach used for our decision table is given as:

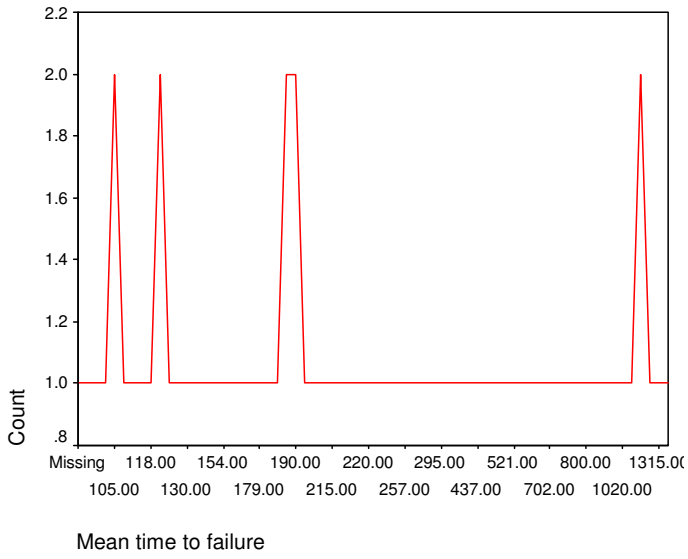
$$\text{COA} = \frac{\sum_{i=0}^{n} f(x_i) * x_i}{\sum f(x_i)}$$ and is calculated to be 767.94.

The descriptive statistics calculated from our original table, Table 1 are tabulated in Tables 3 and 4:
We have the following values from Tables 3 and 4 above:
Mean time to failure (MTTF) = 403.800
Mean time to maintain (MTTM) = 1.5143
Mean time between failure (MTBF) = MTTF + MTTM = 405.3143
Reliability R_s = MTTF (1 + MTTF) = 0.9975
Availability A_s = MTBF (1-MTBF)
= -163874.3675
Maintainability M_s = 1/ (1 + MTTM) = 0.4
The descriptive statistics calculated from our fuzzified table, Table 2, are tabulated in Tables 5 and 6:
We have the following values from our fuzzified Table 2:
Mean time to failure (MTTF) = 0.156647
Mean time to maintain (MTTM) = 0.156647
Hence,
Mean time between failure (MTBF) = 0.313294
Reliability R_s = 0.1354318
Availability A_s = 0.2138563
Maintainability M_s = 0.364568.

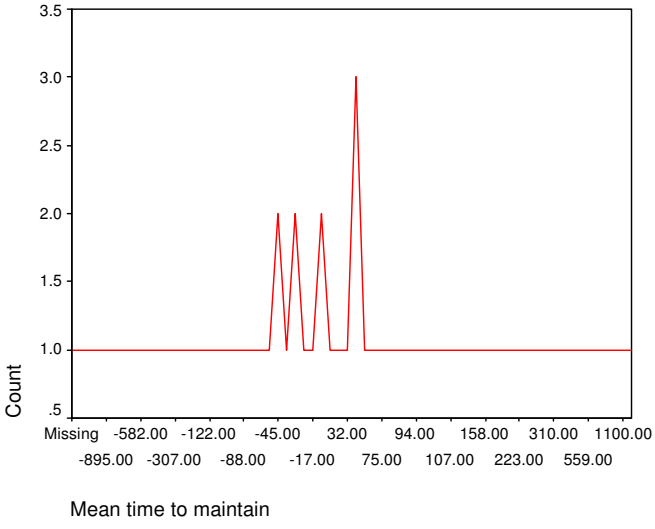
Analysis and Conclusion

In the original data in Table 1, the reliability value retuned by our metrics is 0.9975. However, the value of reliability calculated by our metrics in the fuzzified Table 2 is 0.135431. The result obtained from the fuzzified data in Table 2 is more reliable than that of the original Table 1, since our measurement in Table 2 is based on the relationship (membership function) between the data items. The implication of this is that the result in Table 1 is not absolutely reliable.

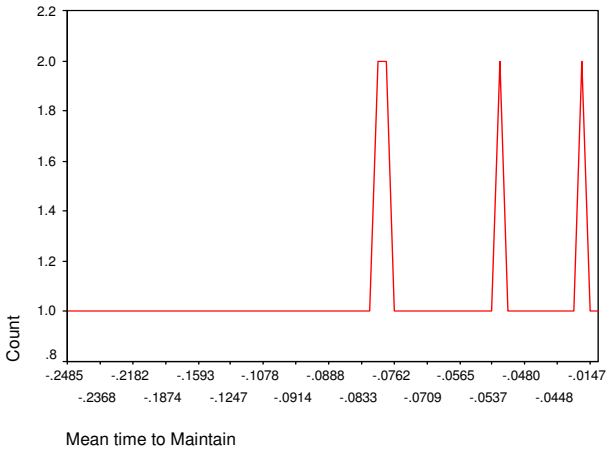
The statistics generated from the original Table 1 did not give us information about the availability, since the calculated value is negative (-591.20946) but the fuzzified Table 2 gives us the value 0.218563. This means that the



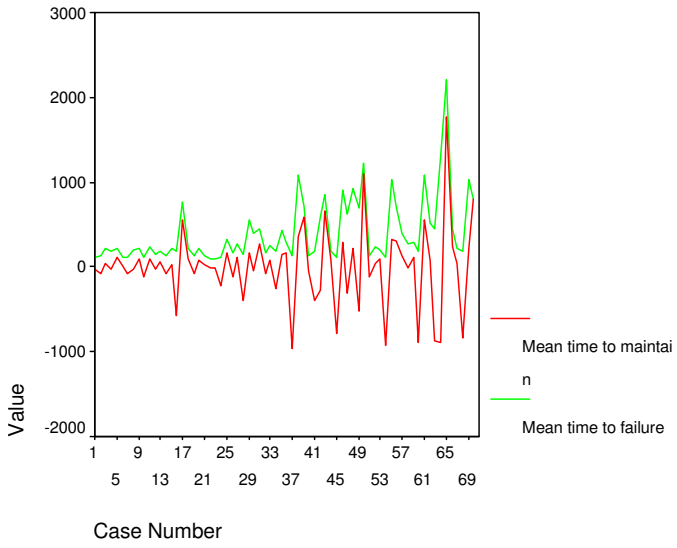
Graph 1. Mean Time to failure.



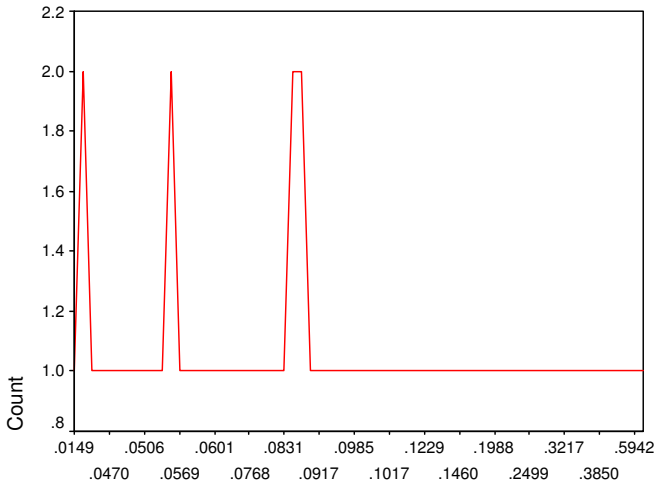
Graph 2. Mean Time to maintain.



Graph 5. Mean time to maintain



Graph 3. Relationship between MTTF and MTTM.

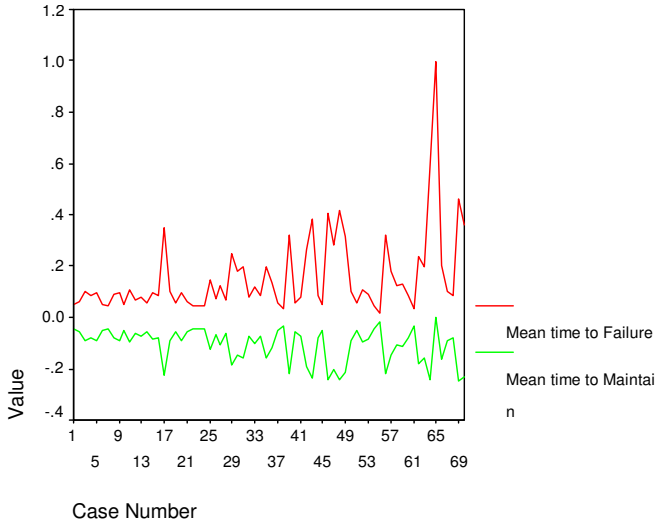


Graph 4. Mean time to failure

software is not well known. On maintainability, the Original Table 1 and fuzzified Table 2 return 0.4 and 0.364568 respectively, this shows that the rate at which it can be maintained is relatively low.

Graph 1 and Graph 4 are favorably comparable which shows that the mean time to failure is very high. However, Graph 4 shows that the mean time to failure can be minimized or even eliminated over a long period of time. Graph 2 and Graph 5 show that it might be expensive to maintain the software.

Graph 3 and Graph 6 show that MTTF and MTTM are closely related. The implication is that more work should be carried out on the software to meet up with international standard. The computer program written in FORTRAN 90 is listed in the Appendix.



Graph 6. Relationship between MTTF and MTTM

REFERENCES

IEEE Standard Glossary of Software Engineering Terminology (1990).

Jawadekar WS (2004): Software Engineering, Principle and Practice, Tata McGraw-Hill Publishing Company Limited, New Delhi.

Land R (2003). Measurement of Software Maintainability, Malardalen University Press.

Leach RJ (2000): Introduction to software engineering, CRC Press, Boca Raton, London New York Washington D.C.

Musa JD (1997). Introduction to Software engineering and testing 8th international symposium on software reliability engineering.

Omolehin JO, Enikuomehin AO, Jimoh RG, Rauf K. (2009): Profile of conjugate gradient method algorithm on the performance appraisal for a fuzzy system, Afr. J. Maths. Comput. Sci. Res. 2(3): 030-037.

Pfleeger SL (1997). Software Engineering, theory & practice, Pearson Education, Washington D.C.

Rankin C (2002). The software Testing Automation frame work, IBM syst. J.

Rosenberg J (2000). Can we measure maintainability, Sun Microsystem, California.

Sommerville IAN (1992). Software Engineering, Addison-Wesley Publishing Company, pp. 389 – 396.

Sugeno M (1985). Industrial Applications of Fuzzy Control, North-Holland, Amsterdam, London, New York 1985.

Zadeh LA (1965). Fuzzy Sets, Information and Control 8, pp. 338-353.

Zimmermann HJ (1987a). Fuzzy Sets in Pattern Recognition, Pattern Recognition Theory and Applications. Springer Verlag, Berlin, Heidelberg, New York 1987. pp.383-391.

Zimmermann HJ (1987b). Fuzzy Sets, Decision Making, and Expert Systems, Kluwer Academic Publishers, Boston, Dordrecht, Lancaster.

Zimmermann HJ, Becker K, Kamahi H, Juffernbruch K, Rau G, Kalff G (1993): An Intelligent Alarm System for Decision-Support in Cardioanaesthesia, Knowledge Base and User Interface. First European Congress on Fuzzy and Intelligent Technologies, Aachen, September 7-10, 1993, pp. 1023-1026.

APPENDIX

```
C D FOR DATA ELEMENTS
C FD FOR FUZZIFIED DATA
C FX MODEL FOR FUZZIFICATION
C SUMFD SUM OF FUZZIFIED DATA
C DFD FUUZZIFIED MULTIPLIED BY FUZZIFIED DATA
C XMAX IS THE MAXIMUM OF THE DATA ELEMENTS
 DIMENSION D (70), FD (70), DFD (70)
 OPEN (5, FILE='IN.PUT')
 OPEN (6, FILE='OUT1.PUT')
 READ (5, 20, END=70) (D (I), I=1, 10)
 READ (5, 20, END=70) (D (I), I=11, 20)
 READ (5, 20, END=70) (D (I), I=21, 30)
 READ (5, 20, END=70) (D (I), I=31, 40)
 READ (5, 20, END=70) (D (I), I=41, 50)
 READ (5, 20, END=70) (D (I), I=51, 60)
 READ (5, 20, END=70) (D (I), I=61, 70)
 XMAXD=2212.0
 SUMDFD=0.0
 SUMFD=0.0
 DO 21 I=1, 70
 FD (I) =D (I)/ (XMAXD+1)
 DFD (I) =D (I)*FD (I)
 SUMDFD=SUMDFD + DFD (I)
 SUMFD=SUMFD + FD (I)
 21 CONTINUE
 CG=SUMDFD/SUMFD
 WRITE (6, 14)
 14 FORMAT (//)
 DO 24 I =1, 70
 WRITE (6, 33) I, D (I), I, FD (I)
 33 FORMAT (1X,'D (', I2,') =', 2X, F7.2, 3X,'FD (', I2,') =',
F6.4)
 24 CONTINUE
 WRITE (6, 15)
 15 FORMAT (///)
 WRITE (6, 34) CG
 34 FORMAT (20X, 'CENTER OF AREA =', F7.2)
 20 FORMAT (70F6.1)
 70 CONTINUE
 STOP
 END
```