Predicting Software Maintainability Using Fuzzy Logic

Neha Prashar[1], Chhavi Rana[2]

Student[1], Assistant Professor[2]
Department of Computer Science and Engineering,University Institute of Engineering and Technology,Maharshi Dayanand University, Rohtak, Haryana, India
[1]iamnehaparashar@gmail.com, [2]chhavi1jan@yahoo.com

Abstract- Ascertaining , efforts and budget spent in the maintainability phase of a software development Maintainability has always drawn the attention of researchers. Software Maintainability is the measure of ease with which a software upgrades, enhances or debugs itself. The software which adapts itself quicker to the upgrades or changes is more maintainable compared to the softwares which takes more time to adapt themselves. The relation between the amount of time taken by the developer to change the software and maintainability is inverse. The time spent and efforts required for keeping software operational consumes about 40-70% of cost of the entire life cycle This study proposes a four parameters that integrated to measure the software maintainability. This study will evaluate how to reduce the maintenance cost and the efforts by using these parameters.

Keywords- Software Maintenance, Fuzzy Logic, MATLAB etc.

I. INTRODUCTION

It is certain that software maintenance represents a lot of the general software spending plan for a data information association. With the proceeding increment in software production an ever increasing number of assets are spent on maintenance. The Maintenance of existing software can represent 70 percent of all efforts exhausted by a software association. It is assessed that numerous organizations will spend near 80 percent of their software allowance on Maintenance if nothing is done to enhance the present approach. Maintenance may traverse for a long time though development might be 1-2 year. Since 1972, software maintenance was portrayed as an "iceberg" to feature the substantial mass of potential issue and costs lie under the surface.

One of the real difficulties in software maintenance is to decide the impacts of a proposed modification on rest of the system. Effect investigation is the action of surveying the potential impacts of a change with the point of limiting surprising reactions. The errand includes surveying the propriety of a proposed modification and assessing the dangers related with its execution, including appraisals of the impact on assets, endeavors and scheduling. It additionally includes the distinguishing proof of the system's parts that should be altered as outcomes of the proposed modification.

These surveys also account that maintenance costs are mostly because of modifications, rather than corrections. A number of technical and managerial problems add to the maintenance costs of a software. Some of the most demanding issues of software maintenance are: course compression, impact analysis, and testing. Whenever a modification is made to a portion of software, it is significant that the maintainer gains a absolute understanding of the organization, conduct and functionality of the system being adapted. It is on the basis of this notion that suggestions of enhancements to achieve the maintenance goals can be generated. As a result, maintainers expend a huge sum of their time understanding the code and the associated documentation to understand its logic, rationale, and organization. Existing estimates specify that percentage of time devoted on program maintenance understanding ranges from 60% to 80%.

One of the foremost challenges in software maintenance is to resolve the effects of a projected adaptation on the rest of the system. Impact analysis is the activity of assessing the possible possessions of a alteration with the aim of reducing unanticipated side effects. The undertaking involves assessing the suitability of a anticipated amendment and evaluating the risks coupled with its implementation, including estimates of the effect on resources, efforts and scheduling. It also includes the acknowledgment of the system's parts that requires to be personalized as an outcome of the anticipated adaptation.

Studies propose that the software maintenance process starts without appropriate learning of the software system. This happens in light of the fact that the software maintenance group is ignorant of the prerequisites and plan documentation. Likewise,

traditional models neglect to catch the evolutionary idea of the software. To defeat these issues, software maintenance models have been proposed

This paper studies the concept of software maintenance system. Further, in section II, it provides the related work of various researchers. In Section III, I vehicle collision avoidance technique is discussed . Section IV includes.

II. RELATED WORK

Chandrashekhar Rajaraman et. al. [1] portrayed a couple of troubles that an individual experiences while testing C++ programs, which may realize program flimsiness. Legacy and polymorphism are the key idea in question situated programming (OOP), and are principal for accomplishing reusability and extendibility, yet they in like manner make programs more hard to get it. We have tried to show up by conflicts and by some exact confirmation that for the most part used many-sided quality measurements like line of code, cyclomatic intricacy, and and those written in other protest arranged vernaculars, since they don't address ideas like legacy and epitome, beside having distinctive weakness. A couple of measures using a thought from the universe of utilitarian deterioration – coupling, are described for C++ programs..

K.K. Aggarwal et. al. [2] depicts that Software upkeep is a task that every progression total needs to stand up to when the product is passed on to the clients' place, introduced and is equipped. The time spent and exertion required for keeping programming operational uses around 60% of cost of entire life cycle. This examination proposes a four parameter joined assessment of programming viability with the help of a fluffy model..

Alian April et. Al [3] manages the assessment and change of one's product upkeep venture by proposing moves up to the product support standards and displayed a prescribed development outline for day by day programming upkeep works out: Software Maintenance Maturity Model (SMMM). The product upkeep act encounters a shortage of administration structures to help its evaluation , control, and ceaseless change. The SMMM has a tendency to fortify the previously mentioned plan. The SMM relies upon professionals.

Alain April et. a.[4] adapt to the evaluation and improvement of the product support work by proposing changes to the product upkeep guidelines and presenting an arranged development display for everyday programming upkeep exercises :Software Maintenance Maturity Model. The Software upkeep work acquire with a lack of administration models to advance its assessment, task, and ceaseless change. The SMMM manages the interesting programming upkeep while safeguarding a structure much the same as that of the CMM development display. The SMMM rotates around the experts, Experience, Universal guidelines and the powerful writing on programming support. He recommended the model's objectives, domain, organization took after by its central approval.

Shyam R. Chidamber and Chris F. Kemerer [5] huge module obviously improvement is the ability to evaluate the procedure. They given the imperative undertaking that product advancement plays in the deliverance and pertinence of data innovation, directors are steadily all the more concentrating on strategy upgrade in the product improvement region. This accentuation has had two sound impacts. The essential is that this request has impelled the terms of an amount of creative or potentially better ways to deal with programming improvement, with conceivably the most vital being object-introduction (OO). Following, the edge on strategy improvement has opened up the interest for programming strategies, or measurements with which to administer the system. The need for such measurements is principally sharp when a foundation is embracing an inventive innovation for which understood practices presently can't seem to be created.

Jane Huffman Hayes et. al. [6] deduced an Adaptive Maintenance Effort Model (AMEffMo). Some measurements, were changed and changes were observed to be unequivocally related to support exertion. The relapse models performed sound in foreseeing versatile upkeep exertion and in addition conceivable helpful data for administrators and maintainers.

Jane Huffman Hayes et. Al [7] show the watch mine-embrace (OMA) perspective that helps relationship in rolling out improvements to their product advancements shapes without concentrating on and undertaking far reaching scale clearing various leveled process change. Astoundingly, the approach has been associated with improve programming practices focused on viability. This novel approach depends on the theory that product groups typically specify target actualities about things that do or don't work dmirably. Groups by then mine their old rarities and their recollections of events to find the product items, structures, measurements, et cetera that incited the observation. By virtue of programming viability, it is then vital to play out some estimation to ensure that the technique result in upgraded practicality. We introduce two viability measures, practicality item

and saw practicality, to address this need. Other practicality measures that may be used as a piece of the mine movement in like manner inspected.

Rikard Land [8] depicted the examination we have as of late begun. We will investigate how the "Viability" of a touch of programming changes as time goes on and it is being kept up by performing estimations on current systems. We show the possibility of "viability", our hypotheses, and our approach.

Warren Harrison et. al. [9] creates most recent portrayal of programming upkeep in light of a target determination run which choose whether a given programming unit can be effectively altered, or on the off chance that it ought to rather be reworked. The paper recommends early location of progress inclined modules using change measures crosswise over release cycles can be useful method in capably distribution of support assets.

Scott L. Schneberger and Ephraim R. [10] depicts the biggest single life cycle PC framework cost has been for keeping up data framework programming. All the more starting late, the figuring scene has begun to encounter a basic change from joined PC outlines to non-thought or circled PC models. This paper discusses another locale of research on programming support, focusing on the issue of whether and to what degree the rising advancement of flowed PC working circumstances particularly impacts programming upkeep. In light of trade journal articles, the issue appears to depend upon two diametrics of data framework plans: part straightforwardness and framework many-sided quality.

III. FUZZY SYSTEM MODEL

The Fuzzy Logic instrument was presented in 1965, likewise by Lotfi Zadeh, and is a scientific apparatus for managing vulnerability. It proposes delicate figuring association the vital idea of processing with writing. It furnishes a procedure to manage imprecision and data granularity. The fluffy hypothesis gives an instrument to speaking to etymological develops, for example, "some," "low," "medium," "regularly," "few" as appeared in fig 2.

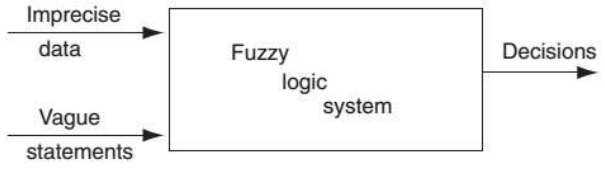

Fig 1: A Simple Fuzzy Logic System [18]

The human cerebrum translates loose and inadequate tactile data gave by discerning organs. Fluffy set hypothesis furnishes a deliberate analytics to manage such data semantically, and numerical calculations are performed by it by utilizing etymological marks predetermined by participation capacities. The Fluffy surmising framework (FIS) when chosen legitimately can adequately display human ability in a particular application. An exemplary set is a fresh defined with a fresh limit. Compared to the traditional set, a fluffy set, is a defined without a new limit. That is, the progress from "has a place with a set" to "does not have a place with a set" is slow, and this smooth change is described by participation works that give fluffy sets adaptability in demonstrating usually utilized phonetic articulations, for example, "the water is hot" or "the temperature is high". The fluffiness does not originate from the irregularity of the constituent individuals from the set, however from the vulnerabilities and loose nature of conceptual musings and ideas. The development of a fluffy set relies upon two things: the distinguishing proof of a reasonable universe of talk and the determination of a suitable enrolment work. In this way, the subjectivity and non-arbitrariness of fluffy sets is the essential contrast between the investigation of fluffy sets and Probability Theory. Fig 3 signifies the participation work for left and right hindrance separate. It helps the vehicle for keeping impact from sides of divider or street.

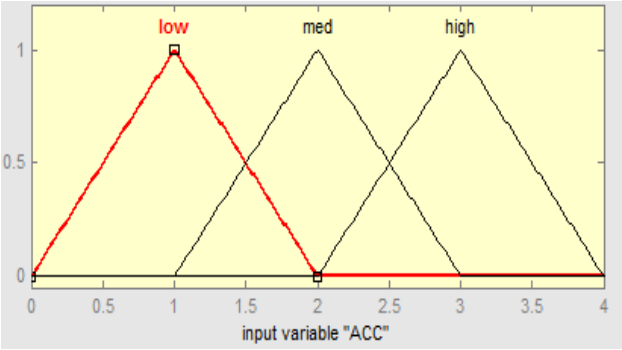

Fig 2: Membership Function of Cyclomatic Complexity

In fluffy framework, the fuzzifier performs estimations of the information factors (input signals, genuine factors), scale mapping and fuzzification (change 1). Hence all the observed signs are scaled, and fuzzification implies that the deliberate signs (fresh information amounts which have numerical qualities) are changed into fluffy amounts. This change is performed utilizing enrollment capacities. In a customary fluffy rationale controller, the quantity of enrollment capacities and the states of these are at first dictated by the client. A participation work has an incentive in the vicinity of 0 and 1, and it shows the level

of belongingness of an amount to a fluffy set. The participation work for speed is appeared in fig 3.

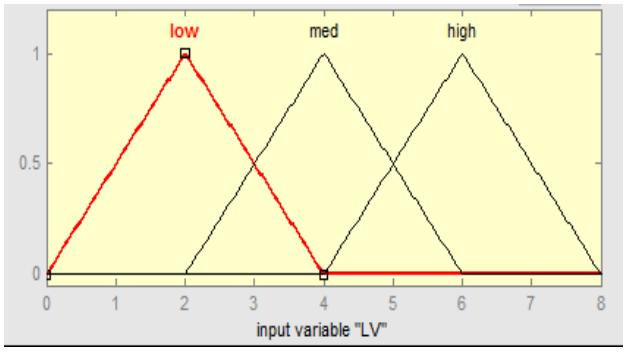

Fig 3: Membership Function for Live Variable

After the phonetic factors and qualities are characterized, the tenets of the fluffy induction framework can be defined. These guidelines delineate fluffy contributions to fluffy yields. This mapping happens through compositional run of induction which depends on Zadeh's expansion of modus ponens which is simply the recognizable if-then restrictive shape. A fluffy if-then administer (otherwise called fluffy run) expect the shape.

If x is A then y is B.

The enrollment capacities can take numerous structures including triangular, Gaussian, ringer formed, trapezoidal, and so on. The learning base comprises of the information base and the etymological control manage base. The information base gives the data which is utilized to characterize the phonetic control rules and the fluffy information control in the fluffy rationale controller. The govern base characterizes (master rules) indicates the control objective activities by methods for an arrangement of semantic standards. At the end of the day, the run base contains guidelines, for example, would be given by a specialist.

Table 1: IF-THEN Rules for Linguistic Variables

IF	THEN
L_Distance is Far and R_Dist is Far	R_Vel is high, L_Vel is high
L_Distance is Near and R_Dist is Near	R_Vel is Slow, L_Vel is high
L_Distance is Near and R_Dist is Medium	R_Vel is Slow, L_Vel is Slw
L_Distance is Near and R_Dist is Far	R_Vel is Slow, L_Vel is Slow

IV. PROPOSED IMPLEMENTATION & RESULTS OF SYSTEM

Programming support is an errand that each improvement assemble needs to confront when the product is conveyed to the client's site, introduced and is operational. We presented an incorporated approach for programming in view of four parameters- Comment Ratio, Average Live Variable, Average Life Variable Span, Average Cyclomatic Complexity utilizing fluffy rationale. Customary measurements like line of code and so forth are not suitable to quantify the intricacy of the product. In this way, we consolidated the customary and more up to date procedures and inferred four elements. All these four parameters are connected to the fluffy model. After the handling of these parameters the yield that will come is practicality.

Maintainability Assessment Metrics

1. Average Number of Live Variable

The normal number of live factors (LV) is the total of the tally of Live Variables isolated by the check of executable articulations (n) LV = LV/n

Where n is an executable statement.

The more, the normal number of live factors, the more troublesome it is create and to keep up a product.

2. Average Life Variable Span

Normal life variable traverse: The traverse can be defined as the number of explanations between two progressive references of the same factor. The normal traverse estimate (LS) for a program can be finished utilizing the condition.

LS program = LS/n

Where n is an executable statement.

3. Comment Ratio

Comment ratio can be defined as

$$CR = (s+c)/c$$

Where s =lines of code and c = number of remark lines. The lesser the proportion, the superior is the intelligibility. Remarks give better meaningfulness and in this manner the Comments Ratio is a vital factor that influences practicality.

4. Average Cyclomatic Complexity

Cyclomatic Complexity has been defined by Maccabe as

$$V = e - n + 2p$$

Where e = the number of edges in a stream chart, n = number of hubs and p = quantity of associated parts.

Fuzzification

The preliminary step is to take the sources of information and settle on how much they have a place with every one of the proper fluffy sets through enrollment's capacities. Our examination in view of eighty standards, and every

one of the principles relies upon settling the contributions to various diverse fluffy etymological sets: If Comment Ratio is low and Average Cyclomatic Complexity is low and Live Variable is low and Life Span is low then viability is great et cetera. Prior to the guidelines can be assessed, the sources of info must be fuzzified as indicated by every one of these phonetic sets.

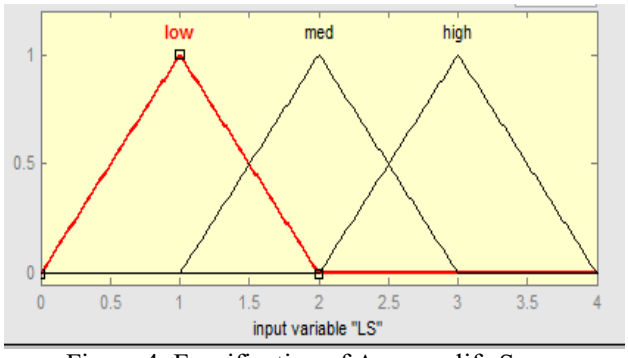

Figure 4: Fuzzification of Average life Span

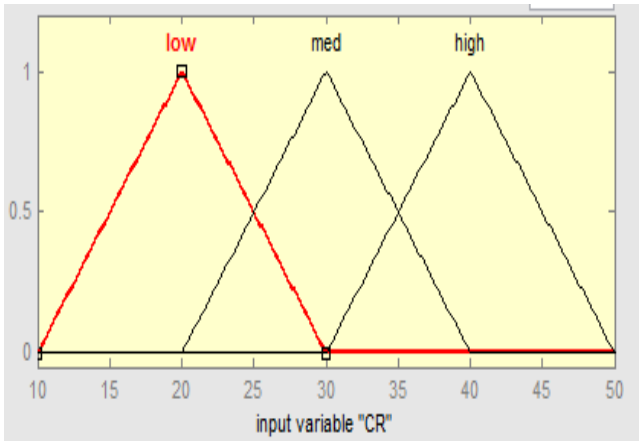

Figure 5: Fuzzification of Comment Ratio

Defuzzification

The contribution for the defuzzification procedure is a set and yield is a solitary number. As much as fluffiness helps the administer assessment amid the middle of the road steps, the last wanted yield for every factor is for the most part a solitary number. Be that as it may, the total of a fluffy set envelops a scope of yield esteems, thus should be defuzzified keeping in mind the end goal to determine a solitary yield an incentive from the set. Maybe the most prevalent defuzzification technique is the centroid figuring, which restores the focal point of region under the bend. There are five worked in techniques upheld: centroid, bisector, center of most extreme, biggest of greatest, and littlest of most extreme.

It is by and large observed that practicality particularly relies upon the sort of information. We have endeavoured to gauge the practicality of programming.. These measurements may not be suitable to gauge the support cost and the endeavours that are utilized to look after programming. In these we have considered the five programming ventures of undergrad building understudies. After thought about the ventures, apply the diverse properties on these tasks, distinctive qualities will be originated from the distinctive activities of all the four parameters. The information is the estimation of the four parameters and the preparing or calculation happens in the fluffy surmising framework and the yield will be the viability. The execution has been done in MATLAB. Keeping in mind the end goal to approve the model, we have considered five programming activities of undergrad designing understudies. They were picked just when appropriate arrangement of info Variables were accessible. The viability was additionally figured utilizing the proposed fluffy model. The outcomes are appeared in Table 1

Table 1: Value of Maintainability

P.No	LV	LS	CR	Cyclomatic Complexity	Maintainability
1	6.5	2.5	45	3.7	9.9995
2	5	0.3	47	3.2	8.001
3	4.8	0.1	9.8	2.4	6.0245
4	5.1	0.01	42	2.2	8.105
5	5.5	0.4	44.5	3.1	9.3456

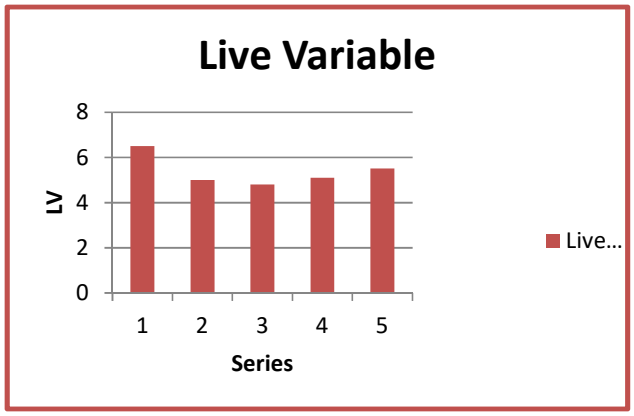

Fig 6: Results for Live Variable

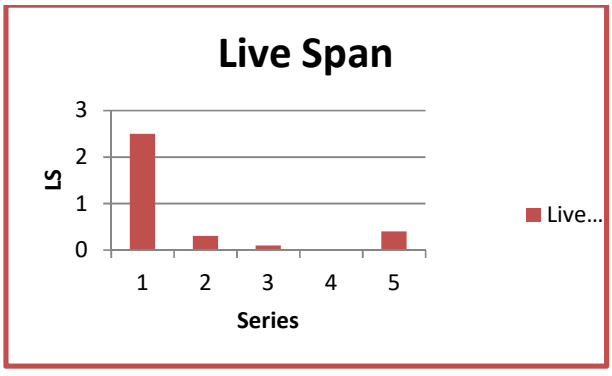

Fig 7: Results for Live Span

VI. CONCLUSIONS

It is usually seen that maintainability depends on the type of Programs. We have attempted to appear by contentions and by some observational examination that broadly utilized conventional measurements may not be fitting to quantify the many-sided quality of the product. This investigation proposes a four parameters that incorporated to quantify the product viability. This examination will assess how to lessen the support cost and the endeavors by utilizing these parameters. Along these lines, we have created fluffy based model for estimating the product practicality. We are about on four elements like Average live Span, Comment Ratio, and Average Cyclomatic Complexity to gauge the viability. We understood that these components will give more exact perspective of viability for programming. Viability can be assessed with the assistance of fluffy model and the outcomes demonstrate that the coordinated estimation of the practicality gives the preferable outcomes over any individual information metric is additionally checked with the assistance of observational outcomes.

Future work that can be done in this field to improve the accuracy of measurement, so as such system can be developed.

REFERENCES

[1]. Rikard Land Mälardalen "Software Deterioration And Maintainability – A Model Proposal" in 1995 University Department of Computer Engineer

[2]. Khairuddin Hashim and Elizabeth Key " A Software Maintainability Attributes Model" Malaysian Journal of Computer Science

[3]. C. van Koten 1 and A.R. Gray 'An application of Bayesian network for predicting object-oriented software maintainability' in 2005 Department of Information Science, University of Otago, P.O.Box 56, Dunedin, New Zealand

[4]. K.K. Aggarwal et. al. 'Measurement of Software Maintainability Using a Fuzzy Model' Journal of Computer Sciences 1(4):538-542, 2005

[5]. P. K. Suri1, Bharat Bhushan2 "Simulator for Software Maintainability" Kurukshetra University, Kurukshetra (Haryana) India IJCSNS International Journal of Computer Science and Network Security, VOL.7 No.11, November 2007

[6]. Markus Pizka and Florian Deißenböck ' How to effectively define and measure maintainability' in 2007

[7]. Mehwish Riaz, Emilia Mendes, Ewan Tempero 'A Systematic Review of Software Maintainability Prediction and Metrics, New

Zealand in 2009 978-1-4244-4841-8/09/$25.00 ©2009 IEEE

[8]. Priyanka Dhankhar1, Harish Mittal2 'SOFTWARE MAINTAINABILITY IN OBJECT ORIENTED SOFTWARE' in 2010 proc.conference 8th may 2010.

[9]. Chikako van Koten Andrew Gray An Application of Bayesian Network for Predicting Object-Oriented Software Maintainability in March 2005 ISSN 1172-6024

[10]. Berns, G., 1984 "Assessing Software Maintainability ."Communications of the ACM, 27: 14-23.

[11]. Baker, A.L.et.al. and R.W.Witty, "A Philosophy for Software Measurement," Journal of Systems and Software, 12, 277-281 (2000).

[12]. Wilde, N. and Ross Huitt: "Maintenance Support for Object- Oriented Programs," Proceedings of IEEE Conference on Software Maintenance Wilde, N. and Ross Huitt: "Maintenance Support for Object- Oriented Programs," Proceedings of IEEE Conference on Software Maintenance

[13]. Booch, G., "Object Oriented Development," IEEE Transactions on Software Engineering, SE-12, 211-221, 1986.

[14]. Halstead,Maurice H. "Elements of Software Science" Elsevier north Holland,New York,1997.

[15]. R. K. Bandi et. al. "Predicting Maintenance Performance Using Object-Oriented Design Complexity Metrics", IEEE T Software Eng, 29, *1*, Jan. 2003, pp. 77 – 87.

[16]. Muthanna, S., K. Kontogiannis and B. Stacey, 2000. 'A maintainability model for industrial software systems using design level metrics.' Proc. Seventh Working Conf.

[17]. Land R.,:Measurement of Software Maintainability", In Proceedings of Artes Graduate Student Conference, ARTES, 2002

[18]. Chandershekhar Rajaraman Michael R. Lyu "Reliability and Maintainability related software metrics in C++" 2003.

[19]. Alain April1 et. al. "Software Maintainence Maturity Model: The software maintainence process model" 2004

[20]. McCabe Thomas j. "A Complexity Measure", IEEE Transaction Software Engineering,Vol2, December 1976.