Life Cycle of Software Development Maintainability Attributes Model using Fuzzification: A Review

Poonam[1], Rekha[2]

[1]M. Tech. Scholar, [2]Assistant Professor
CBS Group of Institutions, Jhajjar, Haryana
Maharishi Dayanand University, Rohtak.

***Abstract*: Software Development goes through a number of phases. These phases together make a Life Cycle of Software Development. It is estimated that more that there are more than 100 billion lines of code in production in the world. As much as 80% of it is unstructured and not well documented. Maintenance can lessen these problems. Maintainability is the ability to keep the system up to date after deploy to the customer site. We studied a number of software maintainability measurement metrics and also new proposed techniques. In our research we focused on how to measure the software. After considering these factors we can conclude that how much software is maintainable This means that how the maintenance cost can be reduced and how much efforts will be required to reduce the cost. So we will use a fuzzy logic to implement these factors. We found that fuzzy logic can be used to model uncertainty for these factors. Fuzzy logic is a way to deal with reasoning that is approximate rather than precise. Then by fuzzy logic we measured the maintainability. In our research, we considered experimental data. First we applied these factors on data and then by fuzzy logic we measured the maintainability. This work based on Rule Base consists of number of rules. Rule structure is like "If this and/or If this then this.**

***Keywords*: Software Development, Life Cycle of Software, software maintainability, fuzzy logic**

Introduction

Maintenance is the concept most commonly correlated with more robust infrastructure and considerably lower long-term costs. Description of maintenance commonly find, for example: "The effort required to fix errors, boost efficiency or other attributes after delivery of a software program or component, The climate has been modified. Or "A program may be sustained if only minimal efforts are made to fix minor bugs."

Perhaps too easy, naturally. The second is particularly misleading, since in reality it is a tautology rather than a concept. The response would be "its correction needs little effort if you wondered what a minor bug is." In addition to this very native description, Specific metric methods aim to characterize main tenability as conformity with a collection of rules that suit the observable characteristics of the system, such as high consistency, minimal coupling, etc. The key concern is the absence of clear rationale for the chosen requirements, which sometimes appear to address some technological elegance instead of enhancing program management efficiently. We published a report in German business organization on network maintenance activities in 2003. Among the 47 interviewees, 60% said that software maintenance is regarded as an "important issue", but only 20% have carried out certain quality checks on maintenance. The maintainability testing requirements used by these 20% range from object-oriented, cyclical complexity, and a small number of lines per process, to a brief description to OMG's service-oriented and model-driven architecture.

Moreover, there is nothing in common about what is actual "maintenance", whether it can be measured and whether it can be performed.

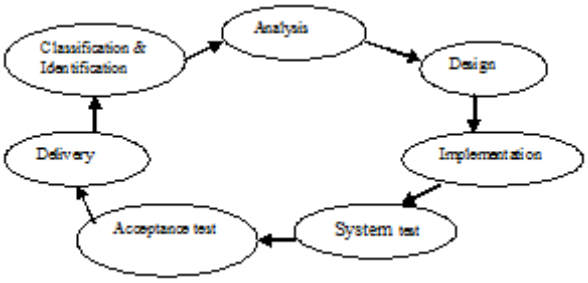

Figure1: Maintenance Process

There is some uncertainty about "standing," whether it can be reached and whether it can be done. There are some misunderstanding. Through using "maintainability" as a concept, this ambiguity can quickly be clarified and overcome. The "fitness" finish is used to turn the "maintainable" term into a product and thereby mark it as a machine object. In addition, the term "retention" applies to the following concepts: the act of "retention" (verb) is an entity or software program that we discover. However, the assumption that the program is very unmaintainable is the characteristics of the machine, and there are many other considerations, such as training

maintenance personnel, managing operational knowledge, and appropriate resources affecting software maintenance activities. Ignore this flaw is most obvious in terms of "readability".

Review of Literature

Chandrashekhar Rajaraman, Micheal R Lyu [1] Describe certain problems that may result in software unreliability in evaluating and managing C + + applications. Likewise, logic and other analytical facts quantify the complexity of C++ systems written in other object-oriented languages (such as lines of code, loop reduction, and computer science indicators) because they cannot solve the subject. I tried to explain what is not enough to have limitations such as legacy and encapsulation. For C++ systems, certain steps are identified with a definition of functional decomposition – coupling. The three commonly employed difficulty indices are measured for five C++ systems, which are contrasted in two of the class couplings and AMC and parallels. Our preliminary findings indicate that compared to the three commonly used sensitivity indicators, paired testing is more directly related to testing and maintenance issues.

Aggarwal et. al. [3] The amount of time and energy taken to maintain the program in service absorbs about 40%-70% of all development cycle costs. This thesis suggests an automated device repair calculation of four parameters using a fuzzy model. The analysis also provides analytical evidence on project maintenance periods used to test the model suggested.

Alain April et. a. [4] Discuss the assessment and optimization of software sustaining feature. A shortage of design frameworks enables the assessment, monitoring and quality development of program repair functions. The SMmm discusses program engineering for the particular tasks while maintaining a structure close to the CMM maturity model. It is intended as an extension to this pattern. Practices are the foundation of the SMmm. Computer engineering experience, universal norms and landmark literature. Our aim, scale, framework, architecture and initial validation are provided.

Shyam R. Chidamberand Chris F. Kemerer The motivation of the evaluation process is an important part of quality control. With its core position in the implementation and usage of information technology, administrators rely more and more on process development of information technology.

Design field of applications. There were two impacts on this focus. The first is that this desire has stimulated a range of different and/or improved solutions to software creation, maybe with object-orientation (OO) the most popular. Third, an growth in demand for electronic metrics or instruments used to monitor operations has been centered on process management. The need for these interventions is particularly evident as a company adopts a modern technology that has yet to acquire proven procedures.

This thesis tackles these criteria by creating and applying a modern series of OO architecture metrics. Prior performance measurement work has typically been subject to one or more forms of critique, while adding to the awareness of software engineering processes in the region. Those involve the absence of a theoretical basis, inadequate generalization, or development relies too much on the calculation of suitable resources, and too much energy for selection. The Bunge ontology became the scientific basis between Wand and Weber for the OO concept approaches. Six concept indicators were developed and analytically tested against a collection of measuring criteria previously indicated. The set of requirements for program metry assessment and a short overview of the scientific data collection sites are given for Weyuker. To order to show viability and propose an observational comparison of such interventions on two field locations, an interactive data collection method was eventually established and introduced. How administrators may use these metrics to enhance operations.

Jane Huffman Hayeset. al. The Adaptive Maintenance Effort Modell (AMEffMo) develops a concept for the calculation of adaptive program maintenance in person, for the purpose of estimating adaptive maintenance in one-to-one period. The regression trends have been effective in estimating proactive maintenance practices and the valuable knowledge for managers and maintainers.

Jane Huffman Hayes et. al. Introduce the paradigm of observation and adoption (OMA) which assists organizations, without committing themselves and implementing large-scale, enhancement of their software's creation processes. In specific, the technique was used to enhance maintenance-oriented development procedures. The idea that tech teams naturally report about issues that perform or don't operate well is based on this innovative perspective. Then, teams searched their objects and recollections for incidents to identify components of apps, procedures, measurements, etc. For the management of devices, any calculation will instead be performed to insure that the process contributes to better management. To order to meet the requires, we introduce two preventive steps, a substance preventive and expected maintenance.

Also investigated were other maintenance steps that can be included in the mine phase. Finally, the team formalizes and adopts mining operations that contribute to established observations of procedures, strategies or behaviors that improve the software product. Two development ventures and a web-based healthcare infrastructure operated by a wider commercial tech company, have been experimentally analyzed in OMA.

Rikard Land explains the work that we have just started. We'll look at how the The "maintenance" of the program is modified as time passes and statistics on manufacturing devices are retained. They discuss the idea of 'maintenance,' our theories and solution.

Warren Harrison et. al. Create a modern software maintenance model focused on an impartial judgment law that defines whether or not a specified software module can be effectively changed. The paper indicates that the early detection of shifting systems may be useful strategies for the productive distribution of maintenance capital by the usage of adjustment steps during release cycles.

Scott L. Schneberger and Ephraim R. Mclean The operating network expense for the management of information technology applications represents the largest single life cycle. More recently, the field of computing has begun experiencing a major transition from the clustered to the centralized or distributed systems of computing. This paper explores a recent study field for software maintenance concentrating on how and how specifically software maintenance is influenced by emerging technologies in distributed computing environments. The topic appears to rely on two diameters of the information system design, based on the trade journal articles: the simplicity of modules and the sophistication of the structures. The bigger the machine elements, the better they are to control each other, but the tougher they are to tackle the entire program. This work was focused on a modern statistical paradigm on part numbers and variety, amount and selection, and the average pace of change for information management sophistication. For purposes addressed here, a report on the topic of the IS system and application-level developers, creators, programmers and consumer relations was undertaken on the secondary source details such as accounting costs for centralized device maintenance.

The field research found that the complete sophistication of distributed systems analyzed exceeded their components' ease of use and usability. The paper also provides an outline of the evaluation and study of the new distributed computing technologies, including proposed areas of specialized work required based on the research findings and implications of the author.

H. Dieterro Mbach and Bradfordt. Ulery Ulery has a big part in both efficiency and management issues in the nature in large-scale computing goods. Improving software maintenance needs better maintenance methodologies, better approval of product requirements before maintenance is published, and better designing methodologies to achieve the quality levels required to satisfy these parameters of approval. Systematic progress in maintenance involves awareness of current problems, capacity to change established practices and a willingness to track their performance. Measurements in information are a tool that helps the development cycle if implemented correctly.
A top-down approach to correct implementation of metrics would be required where oriented changes seek to decide what data is to be gathered and how they are to be interpreted. Within this report, a realistic solution is proposed to enhance management of applications by measurements. This method is focused on general indicators and changes templates. Models, application and functional recommendations are provided for converting them to industrial maintenance. Finally, some descriptions of implementations of the real-world management method are addressed.

George E. Stark It describes the central role of many organizations in the maintenance of software. Those facets of products and processes which tend to influence the expense, timetable, efficiency and functionality of a software maintenance distribution are common for managers to define and calculate. This paper refers to the specific concerns of a single organization's technical service and addresses those actions on the basis of their responses. Attriving to maintain and engineering the areas of development, track progress over time, and help render choix among alternatives is assessed, both in the software maintenance phase and the resulting product.

M. Burgin et. al. Describes the significant and fairly recent reuse of information Computer systems strategy. Application. It intends to establish more technical criteria for product reusability evaluation methods and mathematical theory. After the introduction, reusability is seen as a factor of usefulness in the second part. It allows you to take advantage of expertise in software reuse metric creation and usage. The third part describes and describes various formats and levels of software indicators. The fourth part is a formal description of the software indicators and their attributes. The work focuses on the development of information engineering and, in particular, on creating more effective measures of reuse.

Melis Dagpinar and Jens H. Jahnke A great deal of criteria for evaluating object-oriented program characteristics, such as height, shield, stability and connection, have been suggested. The findings reveal that direct coupling parameters of the scale and import are important predictors of class retention, whereas the measurements of descent, unity and indirect / export coupling are not.

Robert Lagerström , Pontus Johnson The theoretical model consists of organizations with corresponding features to construct business models of architecture. The Nose Such models offer guidance for the quantitative management review of the knowledge. In these evolving programs, IT decision makers use the model should be able to forecast the costs of evolving for specific software projects and collect risk analysis data. It encourages IT leaders to consider their project lists a focus and to prioritize reform programs.

Priyanka Dhankhar and Harish Mittal Describe the management of applications is a measure of how easily a software program or feature can be adjusted for the purpose of fixing bugs, enhancing efficiency or other attributes or adapting to a changing setting' We provide an overview of the design of object-oriented applications in this article. They are necessary for ensuring reusability and expandability. . Through empirical review, to prove that object-oriented difficulties are usually not enough to quantify content written in other object-oriented languages, we will discuss issues such as encapsulation inheritance and text. , Information Sciences Metrics of Halstead and Cyclomatic difficulty of Mc Cabe.

Priyanka Dhankar and Harish Mittal Provides product servicing as a function for all engineering departments as the program is shipped, configured and usable to the location of the client. The amount of time and energy taken to maintain the program in service absorbs about 40%-70% of all development cycle costs. A systematic test of object oriented computing focused on two criteria, class binding and cyclomatic complexity utilizing fused logic, is suggested in this report. In addition, this analysis provides observational evidence on class maintenance times used to test the method suggested.

Megha and Harish Mittal: The easiest way to fix errors, boost efficiency or other characteristics is to calculate software maintenance, or adjust to the changing environment, utilizing a software program or part. Maintenance relies very much on the software form, as is commonly known. Maintenance of applications is a time consuming, expensive step of the life cycle of a software system. Over the full life cycle, the time and resources needed to run software consume approximately 40% to 70% of the cost. In this article, we will discuss the manner in which a paradigm introduced decreases the uncertainty and operational costs of programs and actions. Reduces or eliminates expensive downtimes and efficient uptime improvements. At times unplanned preventive works that have less effect on development may be carried out.

Megha and Harish Mittal: Maintenance of software is a challenge undertaken by every designing party as the program is shipped, activated and usable at the customer's location. Maintenance relies very much on the software form, as is commonly known. Maintenance of applications is a time consuming, expensive step of the life cycle of a software system. This report suggests a 4-parameter comprehensive analysis for software maintenance estimation. The time invested and resources required for the management of software are roughly 40% to 70%. Using these criteria the analysis should determine how repair expenses and resources are popular. We have therefore established a blurry model for device repair measurements.

Conclusion

Various techniques have been developed, including various major ductility factor measurement factors, such as Chandrasekhar. Consider four factors, such as the average number of real-time variables, average real-time span, and comment rate to measure the main persistence. We found that these variables provide a more detailed view of software maintenance. A fuzzy model can be used to estimate maintenance, and the results use empirical results to prove that the comprehensive maintenance value produces better results than a single input indicator. Since different values of the four parameters are considered, the values of these parameters should be small to keep maintenance costs low. Reduce the work of calculating sustainability.

Future Scope

Further work to increase the accuracy of measurement in this field may be undertaken to build such a framework. We suggest that this model be tested in real time. The time needed to correct this error in the maintenance period is determined when any error is found in the project.

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 in2009978-1-4244-4841-8/09/$25.00 ©2009 IEEE
[8]Priyanka Dhankhar1, Harish Mittal '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.