MEASUREMENT TECHNIQUES FOR QUALITY IMPROVEMENT OF A SOFTWARE PRODUCT

[1]Renu Yadav, [2]Mr. Ramesh Loar (Guide)

[1]M.Tech. Scholar, [2]Assistant Professor, Department of C.S.E,

M.R.K. Institute Of Engineering & Technology, Saharanwas, Rewari , Haryana , (India)

ABSTRACT

Software Estimation is an approach to develop software systems with the aim to improve software quality and productivity. It is an activity that is difficult to perform and manage effectively. Over the last few years software quality measurement claims a large proportion of organizational resources. It is thought that many problems in software quality improvement result from inadequate practices in software development such as lack of programming standards and guidelines, solving bugs and problems, implementing changes, testing, impact analysis and integration with other systems. So it is very difficult to estimate the software products quality. It has been proved by various researchers that software quality can only be achieved by making appropriate decisions on the strategic characteristics.

Keywords: Quality Measurement, Software Estimation, Software Quality.

I INTRODUCTION

1.1 Software

Software is a set of instructions to acquire inputs and to manipulate them to produce the desired output in terms of functions and performance as determined by the user of the software. It also includes a set of documents, such as the software manual, meant for users to understand the software system.

1.2 Software Project Estimation

Software project estimation is the process of estimating various resources required for the completion of a project. Effective software project estimation is an important activity in any software development project. Underestimating software project and under staffing it often leads to low quality deliverables, and the project misses the target deadline leading to customer dissatisfaction and loss of credibility to the company.

Software project estimation mainly encompasses the following steps:

1. Estimating the Size of Project. There are many procedures available for estimating the size of a project, which are based on quantitative approaches like estimating Lines of Code or estimating the functionality requirements of the project called function point.
2. Estimating Efforts Based on Person-month or Person-hour. Person-month is an estimate of personal resources required for the project.
3. Estimating Schedule in Calendar Days/Month/Year Based on Total Person-month required and Manpower Allocated to the project. Duration in calendar month = Total person-months/Total manpower allocated.
4. Estimating Total Cost of the Project Depending on the Above and Other Resources. In a commercial and competitive environment. Software project estimation is crucial for managerial decision-making. The following table gives the relationship between various management functions and software metrics/indicators. Project estimation and tracking help to plan and predict future projects and provide baseline support for project management and supports decision-making.

1.3 Motivation

Nowadays IT industry is the need of the hour. The use of software is the key factor in IT industry. Any of the software is not 100% accurate. There is always a scope of improvement in it. It is a continuous process to obtain the maximum quality and efficiency of the software. Considering the fact, we can use “fuzzy logic” for quality improvement of a software product because this is not much used for quality improvement of software product.

1.4 Objective

This is a major concern for the software industry, as lack of estimation performance often causes budget overruns, delays, lost contracts or poor quality software. Because of these rather dramatic consequences, there is a high demand for more research on the topic of estimation and measurement in software development projects. The objective of my research works are follows: To analyses existing quality models and compare them. To design a proposed framework for prediction of software size that helps in quality improvement. To design a software quality model using Fuzzy Logic.

II THEORETICAL BACKGROUND

Over the past decades large investments in software engineering research and development have been made by academia and the software industry. These efforts have produced considerably insight in the complex domain of software development, and have paid off in the shape of the improved tools, languages, methodologies and techniques.

Software estimation has always been an active research area. Accurate software estimation is desirable in any software project, not only to properly schedule budget, resources, time and cost and avoid overrun but also to reasonably estimate as software organizations with better estimates and planning will be able to get the projects in bidding. Pre-bid estimation is paramount in getting business for the company. Accuracy of pre-bid estimation governs the smooth running and success of a project. Estimation output forms the basis for subsequent project plan and activities as well as client commitments [1].

2.1 Software Size Computation

- Effort estimation in person-hour is derived from software size.
- Cost & budget calculation.
- Proper scheduling, resource allocation is done as a final step.

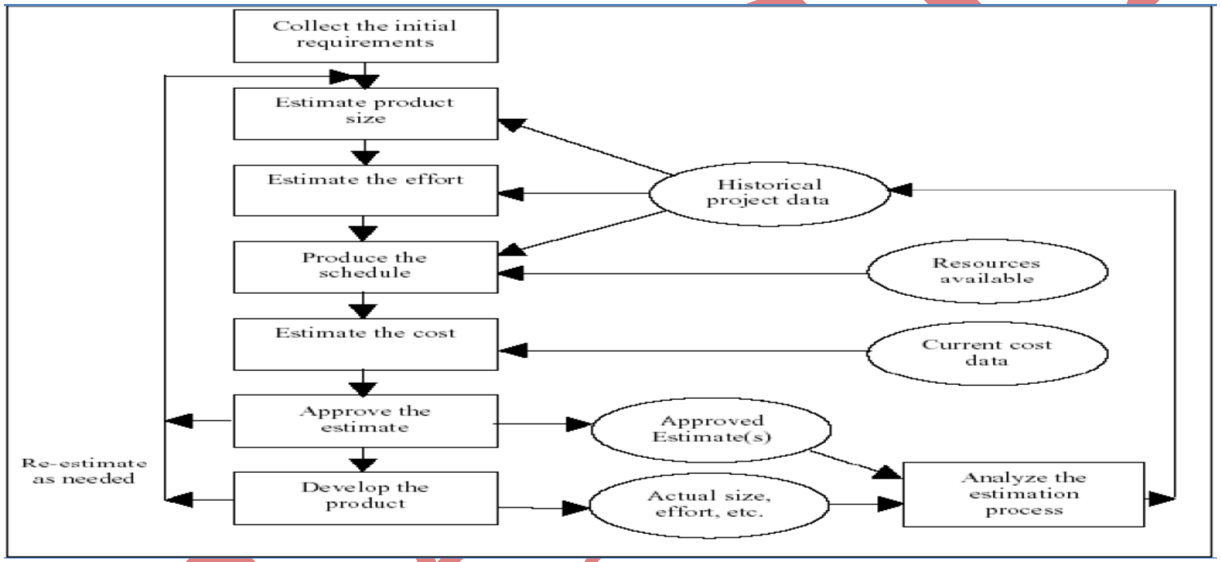

Figure 1: Basic Project Estimation Procedure

2.2 Estimation Methodologies

2.2.1 Analogy Method

In analogy approach the project to be estimated is compared with the already completed projects of that type if exists. The historical data of previously completed projects helps in the estimation. However it works only when previous data is available needs to systematically maintained database.

2.2.2 Top down Method

Top down approach requires less functional and non-functional requirements and is concerned with the overall characteristics of the system to be developed. This estimation is quite abstract at the start and accuracy improves step by step. It can underestimate the cost of solving difficult low-level technical components. However top down approach takes into account integration, configuration management and documentation costs.

2.2.3 Bottom up Method

This method does estimation of each and every individual component and combines all components to give the overall, complete estimation of project. This approach can be an accurate method if the system has been designed in detail. However bottom up method can underestimate the cost of system level activities such as integration and documentation.

2.3 Estimation Techniques

1. Parametric Approach like FP (Function Point), LOC etc and various model based estimations.
2. Heuristic Approach that covers Expertise Based, Learning Oriented estimations etc. All of the heuristic techniques are "soft" in that no model based estimation is used. There are many techniques that come under parametric as well as heuristic approaches. Few are elaborated.

2.3.1 Parametric Approach

a) Line Of Code (LOC): Direct software size can be measured in terms of LOC (Lines of code), one of the oldest techniques. This measure was first proposed when programs were typed on cards with one line per card. Its disadvantage is that accuracy of LOC is highly dependent on the software completion and before that only expert judgment estimates can be given. Also LOC is language dependent.

b) Function Points Metrics: Albrecht of IBM developed the Function Point metrics [2]. In 1984, the nonprofits organization, International Function Point User Group's IFPUG set standards of Function Point Analysis and promoted the metrics and its development.

c) COCOMO and COCOMO-II: Constructive Cost MOdel (COCOMO) was first proposed by Barry W. Boehm [3]. An empirical well-documented, independent model not tied to a specific software vendor, based on project experience is quite popular for software cost and effort estimation. The most fundamental calculation in the COCOMO model is the use of Effort Equation to estimate the number of Person-Months required developing a project. The estimate of a project's size is in SLOC. To get the respective results COCOMO takes LOC. COCOMO- II takes LOC, Function or Use Case points as software size input.

d) COCOMO model is provided for three operational modes:

e) Organic: Applied in projects that have a small, experienced development team developing applications in a familiar environment.

f) Semi-detached: Semi-detached mode is for projects somewhere in between.

g) Embedded: Embedded mode should be applied to large projects, especially when the project is unfamiliar or there are severe time constraints.

2.3.2 Basic COCOMO Model

A brief overview of basic COCOMO is illustrated.

a) Feature Points. To apply FP logic to software such as system software, communication software etc, there is another method called Feature Points. Feature point extends the function points [4] to include algorithms as a new class [5]. It has an additional parameter to measure algorithms with a weight of 3. Feature point model is not in popular use however this measurement is especially useful for systems with little input/output and high algorithmic complexity, such as mathematical software, discrete simulations, and military applications.

Mode	**Effort**	**Schedule**
Organic	$\text{Effort} = 2.4(\text{Size})^{1.05}$	$\text{Time} = 2.5(\text{Effort})^{0.38}$
Semi-detached	$\text{Effort} = 2.4(\text{Size})^{1.02}$	$\text{Time} = 2.5(\text{Effort})^{0.36}$
Embedded	$\text{Effort} = 2.4(\text{Size})^{1.20}$	$\text{Time} = 2.5(\text{Effort})^{0.32}$

b) 3-D Function Points. 3-D function points [6] focuses on the problem of applying function points to scientific and real-time applications. The model uses three dimensions to measure data, function and control. It has also been claimed that the model is suitable for project oriented development. IV) UKSMA Mark II FPA. Symons [7]-[8] proposed a new variant based on Albrecht‟s FPA method in 1988. He concluded that besides 14 influential factors cited by Albrecht 6 more factors would be helpful. This metric was called Mark II FPA. UKSMA (United Kingdom Software Metrics Association) maintains Mark II FPA Standard.

2.3.3 Heuristic Approach

a) Expert Judgment Method: Expert judgment is done based on experience either just by a project manager or by a team of experts involved in the project. Process iterates until some consensus is reached. It works well in situations where no historical data is available. For estimation accuracy industry data can be used as a reference. Very small growing organization often makes use of this technique, however irrespective of the size or maturity of a software house, expert judgment is the wildly used method in the Industry. Several variations are adopted under expert estimation like it can be done in a group of experts of different domains belonging to the same or different projects [9].

Recommendations for Better Estimation

1. For accuracy any estimation technique should be applied through both the top down and bottom up approach.
2. Estimate the software size using a number of techniques, and then average these results to produce a combined estimate.
3. Estimation becomes more accurate as the development progresses.

4. Estimation should be based on several methods. If these do not return approximately the same result, then one has insufficient information available to make an estimate. Some action should be taken to find out more in order to make more accurate estimates.
5. As the metrics program matures, use the data collected from previous projects to develop specific estimating procedures and formulas.
6. Remember to re-estimate as the project progresses. Software estimates usually increase over the life of the project, and one should adjust cost and effort estimates accordingly.
7. Estimate the size of each program component (function block or software component) independently and relate this size to similar products and project components

III PROPOSED WORKS

Objective 1: -To analyses existing qualities models and compare them.

There are a number of quality models in software engineering literature, each one of these quality models consists of a number of quality characteristics (or factors, as called in some models). These quality characteristics could be used to reflect the quality of the software product from the view of that characteristic. Selecting which one of the quality models to use is a real challenge. In this paper, we will discuss the contents of the following quality models:

1. McCall's Quality Model.
2. Boehm's Quality Model.
3. Dromey's Quality Model.
4. FURPS Quality Model.
5. ISO 9126 Quality Model.

In addition, we will focus on a comparison between these quality models, and find the key differences between them. The rest of this paper is structured as follows: presents an overview of the five common quality models used in software engineering. It contains a detailed analysis and comparison between the five quality models.

Objective 2: - To design a proposed framework for prediction of software size that helps in quality improvement.

Objective 3: -To design a software quality model using Fuzzy Logic.

Fuzzy Logic

The Fuzzy Logic Toolbox for use with MATLAB is a tool for solving problems with fuzzy logic.

The concept of Fuzzy Logic (FL) was conceived at the beginning of the 70s by Lotfi Zadeh, a professor at the University of California at Berkley, and presented not as a control methodology, but as a way of processing data by allowing partial set membership rather than crisp set membership or non-membership. Fuzzy logic is a convenient way to map an input space to an output space. The mathematical concepts behind fuzzy reasoning are very simple. What makes fuzzy nice is the "naturalness" of its approach and not its far-reaching complexity. The FL model is empirically-based, relying on an operator's experience rather than their technical understanding of the system. It provides a convenient way to represent Linguistic variables and subjective probability. Linguistic variables are the variables whose values are not numbers but words or sentences in a natural or artificial language. The use of fuzzy set theory allows to incorporate the unavoidable imprecision of the data.

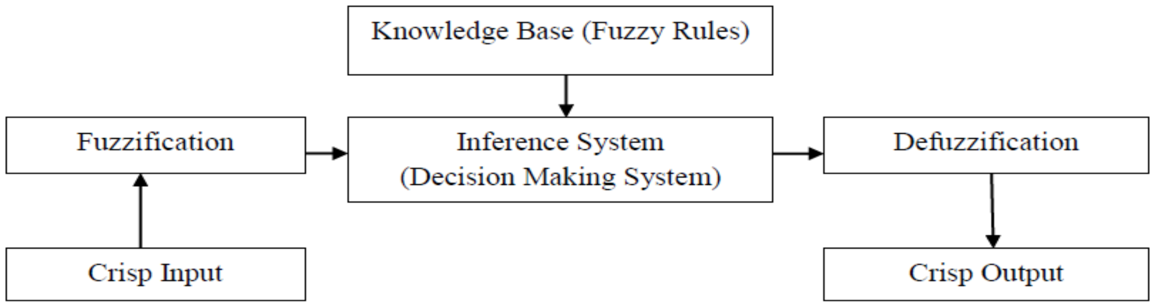
Knowledge Base (Fuzzy Rules)
Fuzzification
Inference System (Decision Making System)
Defuzzification
Crisp Input
Crisp Output

Figure 2: Fuzzy Inference System

3.1 Implementation of Fuzzy Logic

Steps for execution of rule base fuzzy system

Fuzzification

- Identify the inputs (factors in our case) and define their ranges based on domain expert.
- Identify the outputs(quality) and define their ranges based on domain expert

- Select a membership function and create the fuzzy partitions (degree of fuzzy membership function) for each input and output.

Fuzzy inference System

- Construct the rule base (based on input range)

In our case

There are five factors each having three values “Low”, “Medium”, “High”. So no of rule=>35 =243 rule

- These membership values are then processed in fuzzy domain by inference engine based on knowledge base (rule base and data base) supplied by domain experts

Defuzzification

- Combine the rules and defuzzify them to get crisp output

Implementing a fuzzy system requires that the different categories of the different inputs be represented by fuzzy sets which, in turn, is represented by membership functions.

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. The input space is sometimes referred to as the *universe of discourse*, a fancy name for a simple concept. There are total 11 membership functions available in MATLAB. We considered Triangular Membership Functions (TMF) for our problem, because of its simplicity and heavy use by researchers for prediction models.

Triangular Membership Functions (TMF) - It is a three-point function, defined by minimum a, maximum *c* and modal value *b* i.e. TMF (*α, b, c*), where (*α* ≤ *b*≤ *c*).

a: lower boundary and c: upper boundary where membership degree is zero

b: the centre where membership degree is 1

$$\mu_A(x) = \left\{ \begin{matrix} 0 & \text{if } x \le a \\ \dfrac{x-a}{b-a} & \text{if } a \le x \le b \\ \dfrac{c-x}{c-b} & \text{if } b \le x \le c \\ 0 & \text{if } x \ge c \end{matrix} \right\}$$

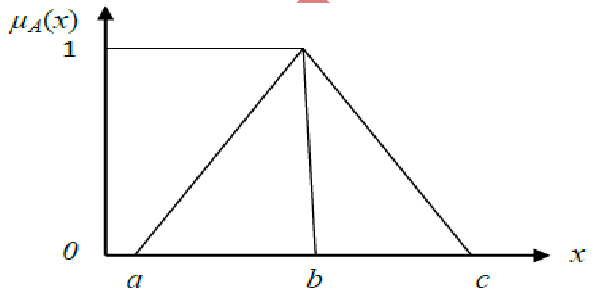

Figure 3: Triangular Membership Functions

IV CONCLUSION

We compared the existing software quality models. We gone through these models and studied about the similarities and differences among them. Also we gone through the different factors related to their qualities. Further we proposed framework for prediction of software size that helps in quality improvement.

In our work we proposed a fuzzy model where, Quality was a measure of characteristics of Software Product. Output shows that these factors have strong correlation with Quality. Result of this model can be a direction for software developer to improve the quality of a software product. It can be concluded that the quality the software product we found to be medium.

REFERENCES

[1] Marcos Kalinowski (2011). From Software Engineering Research to Brazilian Software Quality Improvement, 25th Brazilian Symposium on Software Engineering IEEE.

[2] Albrecht (1979). Measuring Application Development, in Proceedings of Joint share/guide/ibm Application Development Symposium.

[3] B.W. Boehm (1981). Software engineering economics, Englewood Cliffs, NJ: Prentice Hall.

[4] Changli Sun (2010). Software Documents Quality Measurement- A Fuzzy Approach IEEE.

[5] Côté, M.-A, Suryn, W, Martin, R. A, LaporteC. Y (2004b). The analysis of the industrial applicability of software product quality ISO standards: the context of MITRE's Software Quality Assessment exercise, in Proceedings of the 12th International Software Quality Management & inspire Conference (BSI), Canterbury, Kent, United Kingdom.

[6] Côté, M.-A., Suryn, W., Martin, R. A., Laporte, C. Y (2004a). Evolving a Corporate Software Quality Assessment Exercice: A Migration Path to ISO/IEC 9126.Software Quality Professional.

[7] Crosby (1979). P.B. Quality is free: The art of making quality certain. New York .

[8] D. V. Dzung and A. Ohnishi (2009). Improvement of quality of software requirements with requirements ontology, in the 9th International Conference on Quality Software.

[9] L.Pfleeger (1996). Software Quality: The Elusive Target. *IEEE Software.*