Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 3, Issue. 3, March 2014, pg.807 – 814

RESEARCH ARTICLE

A Fuzzy Based Model for Software Quality Estimation Using Risk Parameter Assessment

Anjali Kinra

Department of Computer Sciences, ITM University, Gurgaon, India

Kinra.anjali@yahoo.in

Abstract—Software Cost Estimation is the challenging factor in project management. Accurate cost estimation helps to complete project with in time and budget. Due to this behavior of the project it is considered as a risky project .Under these conditions risk management is mandatory. Large numbers of estimation models have been proposed over the last 30 years. Constructive Cost Model (COCOMO) is one of existing model, which is used for estimation and also for fuzzy based analysis. In this paper, we are using fuzzy based approach which is used for software quality estimation. Fuzzy Logic was primarily bestowed in to check however rule based system can solve the software effort estimation drawback. The aim of this paper is to analyze the process, product and platform based attribute by applying rule based system. Analysis is divided in to two stages; the fuzzification of individual risk and a collaborative analysis with fuzzy modeling. These will be performed to conclude the software quality.

Keywords- Fuzzification; COCOMO; SLIM, Function Point; Risk; Risk attribute

I. INTRODUCTION

Risk invariably involves uncertainty and also the potential for loss. Risk in a software development project is additionally known as "software risk" and is defined as "a live of probability of an unacceptable outcome affecting the software project, process, or product". Risk management also plays a vital role here, considering the fact that a software project will be used in an environment where the results are intangible and subject to a higher level of uncertainty compared to the other types of projects. In the Project Planning Phase, risk management activities focus mostly on risk assessment, which is a discovery process of identifying the potential risks, analyzing or evaluating their risk effects, and prioritizing the risks. Risk identification activity focuses on enumerating the possible risks, creating a risk statement, and establishing the context of the possible risks as deliverables. Based on the risk statement and the risk context, all aspects of the risks can then be analyzed and prioritized, so that a project manager can determine where action should be taken to manage such risks . Hence, the risk assessment phase will provide the information about the number of risks and an estimate of the risk-exposure relationship of each. In order to support project managers in a software development project, several models have been developed to assist in the Effort

Estimation and Software Risk Assessment. The most significant effort estimation models[1] that have been utilized within the programming development projects will be the Constructive Cost Model (COCOMO), the System Evaluation and Estimation of Resource Software Evaluation Model (SEER-SEM) ,and the Software Life Management (SLIM) model[2]. Barry Boehm developed COCOMO model in 1980's, is the most widely used estimation model for software project. Even though the risk assessment (identification and analysis) is conducted together with the software estimation in the Project Planning phase, risk identification and analysis is usually done separately from cost estimation. The Expert-COCOMO model improves on this process by utilizing the information taken from the effort estimation step to establish a risk assessment for a particular software project. The cost factors in the form of scale factors and effort multipliers in the COCOMO model become the inputs for the Expert-COCOMO model. The output for the model is a list of software risks that are related to the COCOMO price variables, such as: Schedule Risk, Product Risk, Platform Risk, Personnel Risk, Process Risk, and Reuse Risk[3]. The software risk taxonomy within the Expert-COCOMO model is presented in Fig 1.1. In second and third section of the paper we have a tendency to discuss regarding COCOMO, and in fourth section, we describe the proposed work based on fuzzy technique.

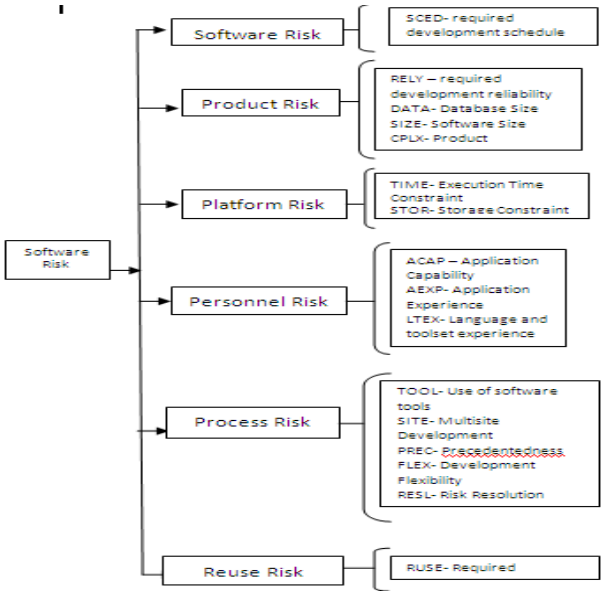

Fig 1.1 Risk Attributes [4]

II. COCOMO

The term COCOMO means "Constructive Cost Model". It helps to calculate effort, cost and schedule for software project [4]. This model was proposed by Barry W. Boehm. COCOMO first version named as "COCOMO81" and it became one of the popular cost estimation models of 1980's.COCOMO Model may apply to three different modes of software project [5]. These modes are as follows:-

1) Organic: In this mode, a small software team is conducted for software development in an extremely acquainted environment.

2) Semi-detached: This mode is sounds like a bridge between the organic and embedded modes. It will either be seen as a combination of characteristics of both modes or a middle level of project characteristic.

3) Embedded: In which, the product is developed inside an extremely coupled complex of hardware, Software and operational constraints. Now , we will discuss the three different levels of COCOMO in detail:

A. Basic COCOMO

Basic COCOMO gives an estimate of effort for a software project using a single property. This property utilized for the computation of the corresponding evaluation is that size of the product. This property is communicated regarding KLOC (Kilo Lines of Code) and is ascertained utilizing the following equation:

$Effort = X \times (Size)^{y}$,

Where;

X, Y: Constants (depend upon the software project mode).

Size: Size of the software (in KLOCs).

Effort: Estimated effort in units of PMs).

When calculating the evaluated development time in this level, the measure of the estimated effort is utilized as the main parameter. The following equation is used to calculate the development time:

The point when ascertaining the evaluated advancement time in this level, the measure of the assessed exertion is utilized as the fundamental parameter. The accompanying equation is utilized to ascertain the improvement time:

$$Time_{dev} = S \times (Effort)^{T}$$

Where,

S, T: Constants (depend upon the software project mode).

Effort: Previously calculated estimated effort (expressed in PMs).

$Time_{dev}$: Estimated development time (expressed in Months)

Table 2.1

Constant values for different classes

Types of Software	**Values of Constants**			
	A	**B**	**C**	**D**
Organic	2.4	1.05	2.5	0.38
Semi-detached	3.0	1.12`	2.5	0.38
Complete	3.6	1.20	2.5	0.38

B. Intermediate COCOMO

The intermediate COCOMO is sort of a wrapper level for the basic COCOMO and it uses the calculable values from the basic COCOMO adds some additional parameters to the estimation formula and recalculates the estimation. This estimation is claimed to be more realistic as it contains numerous aspects that have effect on the amount of needed effort and time to comprehend a software project rather than one aspect.

In the perspective of intermediate COCOMO, there are 15 different factors that influence a software project. These factors are arranged in 4- different classes:

Software Product

- Reliability Requirement
- Database Size
- Product Complexity

Computer System

- Execution Time Constraints
- Main Storage Constraints
- Virtual Machine Volatility
- Computer Turnaround Time

Hardware Resources

- Analyst Capability
- Virtual Machine Experience
- Programmer Capability
- Programming Language Experience
- Application Experience

Software Project

- Use of Modern Programming Practices
- Use of Software Tools
- Required development Schedule

C. Complete COCOMO

Most programming frameworks, particularly the large ones, consist of distinctive parts or subsystems. The intricacy for each of these parts or subsystems is doubtlessly not quite the same as others. Considering this substance, the complete COCOMO recognizes the framework as heterogeneous in nature. As stated by the complete COCOMO, a product framework is made out of different segments or subsystems where each of these components or subsystems has diverse characteristics from one another.

III. COCOMO II

COCOMO faced a ton of issues to evaluating the costs of software that created new life cycle process for example, rapid development process model and object-oriented approaches. Therefore, COCOMO-II was provided in 1995 having three sub models; an application-composition model, an early design model and a post-architecture model. COCOMO-II has, a set of seventeen Effort Multipliers (EM) or cost drivers as an input, which are used to conform the nominal effort (PM) to reflect the software product being produced.

IV. Fuzzy Logic

The concept of fuzzy logic (FL) is not a bearing methodology, however could be a means of process knowledge by permitting partial set membership rather than crisp set membership or non-membership. It supports fuzzy set theory and introduced in 1965 by Prof. L.A Zadeh in the paper fuzzy sets [6]. If feedback controllers may well be programmed just to accept noisy and imprecise input, they may be far more effective and perhaps easier to implement. FL is a problem-solving control system methodology that lends itself to implementation within the system ranging from simple, small, embedded micro-controllers to large, networked, multi-channel PC or workstation-based knowledge acquisition and management systems. It can be enforced in hardware, software, or a mixture of both.

A. Fuzzy Systems

Fuzzy systems are knowledge based or rule based system. The heart of fuzzy systems is a knowledge base consisting of the so called Fuzzy "If Then rules" in which some words are characterized [6] by continuous member functions. The famous fuzzy logic systems can be distributed into three sorts: pure fuzzy logic system, Takagi and Sugeno's fuzzy system and fuzzy logic system with fuzzifier and defuzzifier. Since most of the engineering applications produce crisp data

as input and wants crisp data as output, the last sort is the most generally utilized fuzzy logic system with fuzzifier and defuzzifier**.** It was initially proposed by Mamdani [7]. It has been successfully applied to a variety of industrial processes and consumer products as show in Fig3.1 below:

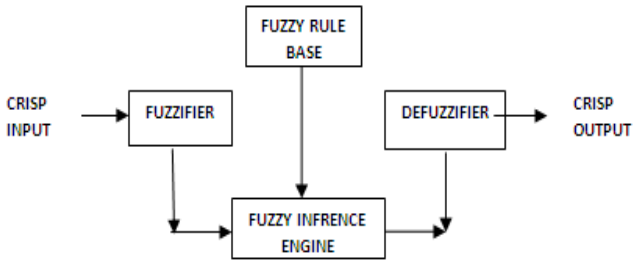

Fig 3.1 Fuzzy logic system

1) Fuzzifier - It converts the crisp input into a fuzzy set, and to describe situation graphically, membership functions are used [8].
2) Fuzzy Rule Base- It uses "if-then rules" formulae.
3) Fuzzy Inference Engine- A collection of if -then rules put away in fuzzy rule base will be referred as inference engine. It performs two functions i.e., aggregation and composition.
4) Defuzzification - It converts fuzzy output into crisp output.

V. Proposed Work

A. Flow of Work

The proposed Fuzzy based model will estimate software quality using Risk Parameter. Initially, collect the database from PROMISE repository [9]. This repository data set made openly accessible keeping in mind the end goal to support repeatable, verifiable, refutable and improvable predictive models of software engineering. It includes large number of risk attributes. Each attribute is described according to their category. Basically, it has four risk category, which are personal, platform, process and reuse. These categories are further split in to different sub categories as shown in Fig 4.1 Now apply the fuzzification on each risk attribute. Assign weight to each sub category. At last apply the rule based system on each software risk attribute. This store information set made openly accessible keeping in mind the end goal to support repeatable, certain

B. Fuzzy Rule Based System

Recognizing the data parameter x from the universe X, and the yield parameter y from the universe Y, the statement of a framework might be described with a rule base (RB) system in the following structure:

Rule1: IF x=A1 THEN y= B_1

Rule2: IF x=A2 THEN y= B_2

Rule n: IF x=An THEN y= B_n

This is indicated as a single input, single output (SISO) framework

If there is more than one rule proposition, i.e. the i^{th} rule has the following form

Rule i: IF x_1=A_1i AND x_2= A_2i…THEN y=bi

then this is indicated as a multi input, single output(MISO) framework**.**

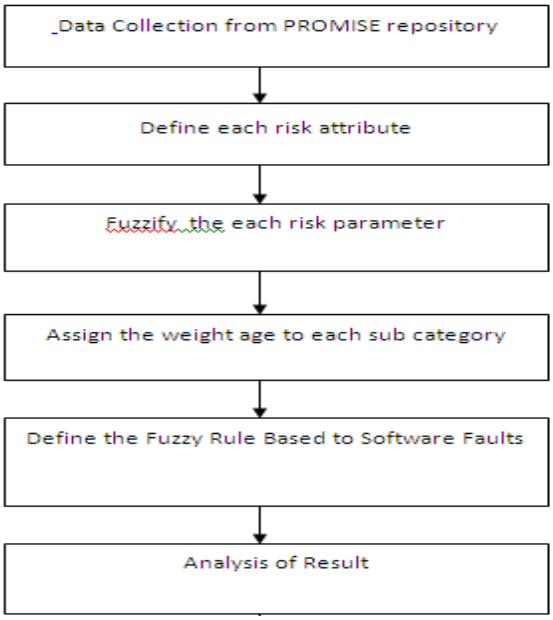

Fig 4.1 Flow of work

After applying the rule based system to each category, we perform analysis on the basis of results occurred. Proposed Model as shown in Fig 4.2 above explains number of risk attributes which are individually fuzzified with the help of fuzzification method. There are large numbers of risk attributes that generally occur in any software project. Here, we simply name them as A_1… A5. Every risk attribute belongs to particular risk category. Basically, there are 4-categories of risk which are personal, process, reuse and platform. These all categories make the complete system.

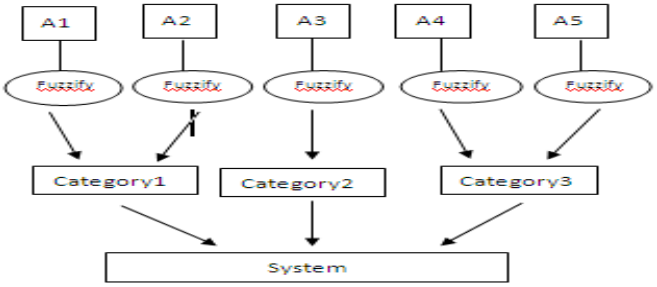

Fig 4.2 Proposed Model

C. Data Collection

The data is collected from PROMISE Software Engineering Repository [8]. Dataset contains various attributes. Here, we categorize the attributes according to the risk occurred. The attributes are here given in Table 4.2 as below:

Table 4.2

Data Collection

S.No	Risk Occurred	ID	Attribute Name	Purpose
1	Personal Risk	ACAP	Analyst Capability	It describes the capability of the analyst. Its rating should not consider the kevel of experience
2		PCAP	Programmer Capability	Evaluation depends upon the capability of the programmer as a team not individuals.
3		AEXP	Analyst Experience	Rating should be done on the level of application experience of project team
4		VEXP	Virtual Machine Experience	It describes how efficiently programmer can use the hardware.
5		LEXP	Language Experience	It measures the level of programming language and software tool experience of project team developing the software system
6		MODP	Modern Programming Practices	It describes the programming knowledge of the team member and rate them according to their levels.
7	Process Risk	TOOL	Use of Software Tools	Tool rating ranges from simple edit and code.
8		TIME	Time Constraint	percentage of available execution time expected to be used by the system or subsystem consuming the execution time resource.
9		TURN	Turnaround Time	This attribute defines how much time it will take to complete the project
10		DATA	Data base size	It captures the influence test data requirements have on program development
11		SCED	Schedule Constraint	It specifies a rating that represents the degree of schedule constraint imposed on a system software
12		STOR	Main Memory Constraint	It specifies a rating that represents thedegree of main storage constraint imposed on a software system or subsystem.
13	Reuse Risk	VIRT	Machine Volatility	describes the volatility of the machines ,how it works efficiently
14		CPLX	Process Complexity	It describes the rating of complex processes of system software.
15		RELY	Required software Reliability	Software must perform its intended function over a period of time.

Here, we describe the different risk attributes which are extracted from NASA repository [9]. The value of these risk attributes vary from "very low to extra high". These Attributes value are identified according to the effort calculated on the different effort multiplier. There are large numbers of effort multipliers in the existing COCOMO Model. The effort multiplier used in the model helps to maintain and manage any software project. The range of each multiplier depends on the capability of programmer or the technician. Fuzzy is best approach for risk assessment and identification. It gives the accurate risk assessment on the basis of different risk attribute. We apply a fuzzification approach on every risk parameter for risk identification and then categorize them. A complete software system contains various risk categories. Now every parameter is distributed according to their category. One risk attribute may belong to two different categories. Now we apply the fuzzy rule based system on each category. It gives accurate software quality estimation of different risk parameters. In this way a fuzzy based model has been developed for software quality estimation on the basis of risk parameter.

VI. **Conclusion**

In this present work, a fuzzy based model is presented for software quality assessment using risk attributes. In all these approaches fuzzy logic techniques are applied on risk parameters for quality estimation of any software. This paper has presented the conceptual model of the presented work with detailed exploration of each stage.

VII.ACKNOWLEDGEMENT

I take this chance to express my earnest much obliged and profound appreciation to those individuals who enlarged their wholehearted co-operation and have helped me in completing this paper.

My deepest thanks to **Dr. SUPRIYA PANDA**, for guiding and rectifying various documents of mine with consideration and care. She has taken the ache to go through the complete paper again and again and make important revisions as and when needed.

I also extended my heartfelt thanks to my family and well-wishers.

References

[1] Ekananta Manalif, Luiz Fernando Capretz and Danny Ho, *Development of Integrated Software Project Planning Model based on Cost Factors and Fuzzy Technique.*
[2] Galorath D. D. and Evans M. W., Software *Sizing, Estimation, and Risk Management*, Auerbach Publication, 2006.
[3] Heldman K. and Heldman W., CompTIA Project+: Study Guide, Wiley Publishing Inc., 2010, pp.34.
[4] B.W. Boehm, "*Software risk management: Principles and practice,"* IEEE Software, Vol.8, Issue.1, Jan 1991, pp. 32-41.
[5] A Survey on COCOMO, Onur Yaman, SWE578 2011F
[6] Zadeh. L. A., *Fuzzy Sets, Information and Control*, 1965, Vol. 8, pp. 338-353
[7] Mamdani, E., H., Assilian, S. (1975), An *experiment in linguistic synthesis with a fuzzy logic controller*, Intern.,. J. Man-Machine Stud. 7. 1-13
[8] Https://code.google.com/p/promisedata/source/checkout.
[9] www.promisedata.org

