PREDICTION OF SOFTWARE REQUIREMENTS STABILITY BASED ON COMPLEXITY POINT MEASUREMENT USING MULTI-CRITERIA FUZZY APPROACH

D. Francis Xavier Christopher[1] and E.Chandra[2]

[1]Director, School of Computer Studies, RVS College of Arts and Science
Coimbatore, Tamil Nadu 641402, India
christopher@rvsgroup.com
[2]Director, School of Computer Studies, SNS Rajalakshmi College of Arts and Science
Coimbatore, Tamil Nadu 641402, India
crcspeech@gmail.com

ABSTRACT

Many software projects fail due to instable requirements and lack of managing the requirements changes efficiently. Software Requirements Stability Index Metric (RSI) helps to evaluate the overall stability of requirements and also keep track of the project status. Higher the stability, less changes tends to propagate. The existing system use Function Point modeling for measuring the Requirements Stability. However, the main drawback of the existing modeling is that the complexity of non-functional requirements has not been measured for Requirements Stability. The Non-Functional Factors plays a vital role in assessing the Requirements Stability. Numerous Measurement methods have been proposed for measuring the software complexity. This paper proposes Multi-criteria Fuzzy Based approach for finding out the complexity weight based on Requirement Complexity Attributes such as Functional Requirement Complexity, Non-Functional Requirement Complexity, Input Output Complexity, Interface and File Complexity. Based on the complexity weight, this paper computes the software complexity point. And then predict the Software Requirements Stability based on Software Complexity Point changes. The advantage of this model is that it is able to estimate the software complexity early which in turn predicts the Software Requirement Stability during the software development life cycle.

KEYWORDS:

Multi-Criteria Fuzzy Based Approach, Functional Requirement Complexity, Non-Functional Requirement Complexity, Input Output Complexity, Interface Complexity, File Complexity, Software Complexity Point Measurement, Requirements Stability Index.

1.0 INTRODUCTION

Requirements Elicitation is the most important stage in the Software Development Life Cycle Process. If the requirements have not been captured correctly during the Elicitation process, then the whole development process will fail which results in time and monetary costs [18]. Software developers often start with unclear, ambiguous, and incomplete requirements with inaccurate understanding of the user needs or insufficient requirements. Therefore, requirements development and management are the start point of the software development process. Software system needs to evolve. In particular, changes in the requirements may be related to the addition of new functionalities, modification to the existing ones, deleting the functionalities which are obsolete or to the improvement in the quality of service offered [10]. Requirements changes not only cause software defects but also cause in delay of delivery of the software project. Requirements changes at the later stage can cause uncertainty in the software development. Sometimes, these requirement changes will affect the quality of the software. For requirements engineering, the challenging issue is not the requirements change. It is how to deal with the change and how to measure them. For measuring the change, measure the software complexity. IEEE defines software complexity as "the degree to which a system or component has a design or implementation that is difficult to understand or verify" [1].

Software Complexity Measurement can be classified into three categories: Size, Structure and Quality Measurement [17]. In present, there are many algorithmic models and non-algorithmic models have been developed to measure the complexity of the software [20]. Some of the famous algorithmic models are Function Point Modeling, Constructive Cost Model (COCOMO), Software Life Cycle Management model (SLIM). Non-algorithmic techniques include Price-to-Win, Expert Judgment and Machine learning approaches. Machine Learning is used to group together a set of techniques that embody some of the facets of human mind. For example, fuzzy systems, analogy, regression tress, rule induction and neural networks. Among the machine learning approaches, fuzzy systems and neural networks are considered to belong to the soft computing groups [7].

This paper attempts to find the complexity weight for Requirement Complexity Attributes using Multi-Criteria Fuzzy Based Approach and compute the Complexity point of developing process. Finally predicting the Requirements Stability Index based on complexity point changes during the software development life cycle. Maximizing the requirements stability will obviously reduce the change impact. This paper is organized as follows: Section II discusses on brief introduction of fuzzy sets, algebraic operations, linguistic variables, triangular membership function and Multi-Criteria Fuzzy Based Approach. Section III discusses on developing the Complexity Point Measurement Model based on Multi-Criteria Fuzzy Based Approach. Section IV discusses on predicting the Requirements Stability based on Complexity Point Changes.

2.0 BACKGROUND ON MULTI-CRITERIA FUZZY BASED APPROACH

Multi-criteria Decision Method deals with the process of making decisions in the presence of multiple criteria or objectives [8] [14] [19]. A decision maker (DM) is required to make decisions among multiple criteria which can be qualitative or quantitative. The DM's evaluations on qualitative criteria are often subjective and imprecise. The weights of the criteria are usually expressed in linguistic terms. Instead of single crisp value for linguistic terms, the multi-criteria fuzzy will use a range of values to incorporate decision maker's uncertainty. In order to deal with uncertainty, the Multi-Criteria fuzzy based approach is used in this research paper.

Fuzzy Logic is a powerful problem-solving methodology to deal with imprecision and information granularity [12] [16]. A fuzzy model is used when the system is not suitable for analysis by conventional approach or when the available data is uncertain, inaccurate or vague. Fuzzy Logic brings us close to human decision making, enabling one to analyze approximate data to precise solutions [11]. The concept of Fuzzy Logic was first developed by Lofti Zadeh in 1965. Fuzzy Logic starts with the concept of fuzzy set theory. It is a theory of classes with un-sharp boundaries and considered as an extension of the classical set theory. Classical theory requires high understanding of the system, whereas Fuzzy logic is completely empirical and relies on experience and knowledge rather than the technical understanding of the subject for modeling the complex system [5] [11].Fuzzy Logic incorporates a simple, rule-based approach for solving the problem rather than solving it mathematically. The popular fuzzy logic systems can be categorized into three types: pure fuzzy logic systems, Takagi and Sugeno's fuzzy system and fuzzy logic system with fuzzifier and defuzzifier [13]. Most widely used fuzzy logic system with fuzzifier and defuzzifier was Mamdani Fuzzy system. It has been successfully applied to a variety of industrial processes and consumer products. Fuzzy reasoning consists of three main components: Fuzzification process, Fuzzy Inference and Defuzzification process [3] [15].

Step#1: Fuzzification Process

Fuzzification process is where the objective term is transformed into a fuzzy concept i.e., it converts a crisp input to a fuzzy set.

Step#2: Inference from Fuzzy rules

Fuzzy logic system use fuzzy IF-THEN rules. A sample fuzzy rule can be written as:

IF X1 is good AND X2 is very good THEN output is good.

Once all crisp input values are fuzzified into their respective linguistic values, the inference engine accesses the fuzzy rule base to derive the linguistic values for the intermediate and the output linguistic variables. From the sample fuzzy rule stated above, good, very good are the linguistic values.

Step#3: Defuzzification Process

Defuzzification process refers to the translation of fuzzy output into objective terms i.e., converting fuzzy output into crisp output.

A system based on fuzzy logic has a direct relationship with fuzzy concepts (such as fuzzy sets, linguistic variables etc.) [6]. A Fuzzy set is a class of objects with continuous membership grades, where the membership grade can be taken as an intermediate value between 0 and 1. A fuzzy subset A of a universal set X is defined by a membership function $\mu_A(X)$ which maps each element x in X to a real number [0, 1]. When the grade of membership for an element is 1, it means that the element is absolutely in that set. When the grade of membership is 0, it means that the element is absolutely not in that set. Ambiguous cases are assigned values between 0 and 1. This can be represented as below:

$$\mu_A(x) = \begin{cases} 1 & \text{if } x \in A \\ 0 & \text{if } x \notin A \end{cases} \tag{1}$$

The degree or extent to which the elements are the members of the interval is known as membership function [13]. A membership function is a curve that defines how each point in the input space is mapped to a membership value between 0 and 1. The input space is also called as the universe of discourse. In this paper, triangular fuzzy numbers are used as membership function. The idea of using fuzzy triangular number is to give the decision maker an opportunity to decide in better way if there is little uncertainty in deciding the dominance of one alternative over the other. Triangular fuzzy number [13] is a three point function defined by minimum (l), maximum (u) and modal (m) values which can be represented as

$$A_{ij} = (l_{ij}, m_{ij}, u_{ij}), \tag{2}$$

Where l_{ij} represents the lower limit
m_{ij} represents the median limit
u_{ij} represents the upper limit

A fuzzy number is a triangular fuzzy number if its membership function can be denoted as follows:

$$\mu_A(x) = \begin{cases} \frac{x-l}{m-l} & \text{for } l \le x \le m \\ \frac{u-x}{u-m} & \text{for } m \le x \le u \\ 0 & \text{otherwise} \end{cases} \qquad (3)$$

The following fuzzy membership function represents the triangular fuzzy set (0.3 0.5 0.7) graphically as in Fig 1.

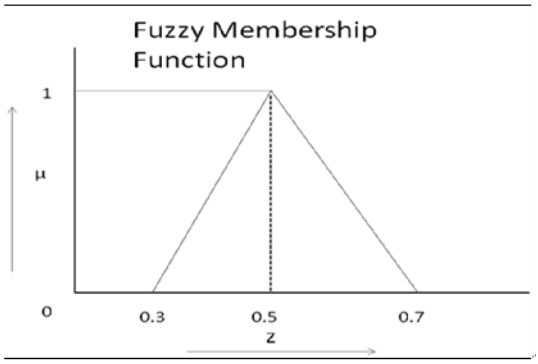

Fig. 1. Triangular Membership function

The following are the fuzzy arithmetic operations [5]:

Defining two triangular fuzzy sets A and B as A = (a1, a2, a3) and B= (b1, b2, b3) then

- Generalized Fuzzy Number Addition:

  $A \oplus B = (a1 + b1, a2 + b2, a3 + b3)$

- Generalized Fuzzy Number Multiplication:

  $A \otimes B = (a1 * b1, a2 * b2, a3 * b3)$

- Generalized Fuzzy Number Subtraction:

  $A \Theta B = (a1 - b1, a2 - b2, a3 - b3)$

- Generalized Fuzzy Number Division:

  $A \phi B = (a1 / b1, a2 / b2, a3 / b3)$

In this paper, fuzzy number addition and fuzzy number multiplication operations are used.

3.0 PROPOSED MEASUREMENT MODEL

This paper proposes Complexity Point Measurement model for measuring the software complexity based on the Software Requirements Complexity Attributes which is derived on the basis of software requirements written as per the recommendations of IEEE: 830: 1998 for Software Requirements Specification (SRS) document [1][2][4]. The following are the Software Requirement complexity attributes:

- Functional Requirement Complexity
- Non-Functional Requirement Complexity
- Input/output Attribute Complexity
- File Complexity
- Interface Complexity

This can be depicted in Fig. 2 as follows:

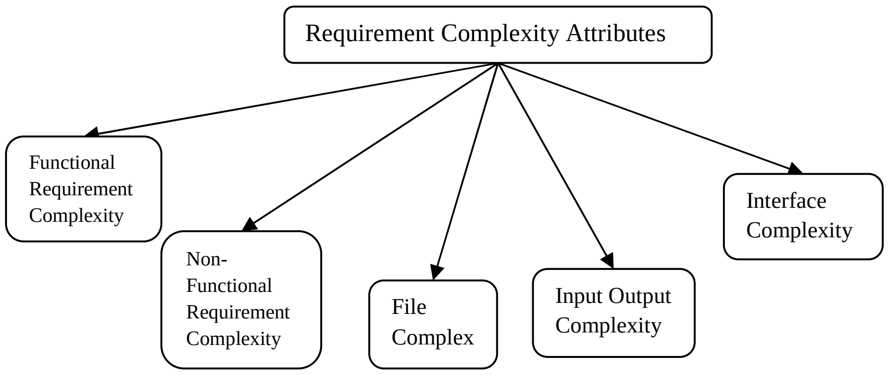
Requirement Complexity Attributes
Functional Requirement Complexity
Non-Functional Requirement Complexity
File Complex
Input Output Complexity
Interface Complexity

Fig. 2. Requirement Complexity Attributes

Functional Requirement Complexity

Functional Requirement defines the functionalities/services that need to be delivered to stakeholders. Functionality refers to what the system supposed to do. Every stated functional requirement can be partitioned into sub-functions or sub-processes. Also the sub-functions may compose of other sub-functions too [9] [21].

Non-Functional Requirement Complexity

Non-Functional Requirements are the constraints upon the behavior of the system and also refers to the system qualitative requirements. Non-Functional requirements are indirectly related to the functionality of the system. Non-functional requirements are associated with the factors like Security, Performance, Flexibility, Usability, Reliability, Scalability and Efficiency and so on also known as quality factors [9].

Input Output Complexity Attribute

This complexity refers to the input and output of the software system. A software system input is an elementary process that processes data or controls the information. Input complexity refers to number of input entering into the system. Output complexity refers to number of output leaving the system [1] [4].

Interface Complexity Attribute

This complexity attribute is used to define the number of external interfaces integrated to the proposed system. The external interface can be user, hardware, software, and communication external interfaces [21].

File Complexity Attribute

This complexity attribute is a user identifiable group of logically related data or control information maintained within the application boundary. File complexity refers to the number of files required for the data storage required during transformation [1] [4].

The steps to be performed for calculating the complexity point measurement:

Step# A: Apply Multi-Criteria Fuzzy Based Approach for finding the Complexity Weight for the Requirement Complexity Attributes

Step# B: Map each requirement to each complexity attribute based on weighting scale factors such as Very Low, Low, Average, High and Very High. Multiply the complexity weight and the total number of requirements in each scale which results in unadjusted Complexity Point.

Step# C: Compute the Adjusted Complexity based on the adjustment factors. Each factor is weighted based on a scale from 0-5 where 0- Not present, 1-Minor Influence to 5-Major Influence.

Step# D: Compute the Complexity Point by summing up Unadjusted Complexity computed in step#2 and Adjusted Complexity computed in step#3.

The following sub-section explains briefly about each step mentioned above.

3.1 Apply Multi-Criteria Fuzzy based Approach

3.1.1 Fuzzification Process

Fuzzification is the process of converting crisp input into fuzzy sets. The linguistic terms can be represented based on approximate reasoning of fuzzy sets. A linguistic term can be defined as a variable and the importance weight can be evaluated by linguistic terms such as very low, low, average, high and very high. For every complexity attribute, there is a corresponding importance weight and rating [5] [8]. The linguistic terms can be expressed as a triangular fuzzy numbers for ratings and weight as in Table1 and Table2.

Table 1: Linguistic Terms for Fuzzy Ratings

Linguistic Term	*Fuzzy Ratings*
Very Low (VL)	(0.0, 0.1, 0.3)
Low (L)	(0.1, 0.3, 0.5)
Average (M)	(0.3, 0.5, 0.7)
High (H)	(0.5, 0.7, 0.9)
Very High (VH)	(0.7, 0.9, 1)

The following Fig. 3 depicts the triangular membership function for the importance weight of criteria based on Table1. Linguistic Terms for Fuzzy Ratings

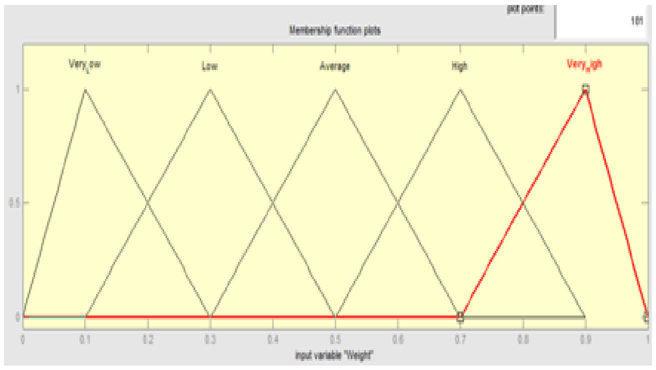

Fig. 3. Membership function for Fuzzy Ratings

Table2: Linguistic terms for Fuzzy Weight

Linguistic Term	*Fuzzy Weight*
Very Low	(0.0, 0.12, 0.25)
Low	(0.12, 0.25, 0.5)
Average	(0.25, 0.5, 0.75)
High	(0.5, 0.75, 1)
Very High	(0.75, 1, 1)

The following Fig. 4 depicts the triangular membership function for fuzzy ratings based on Table2.

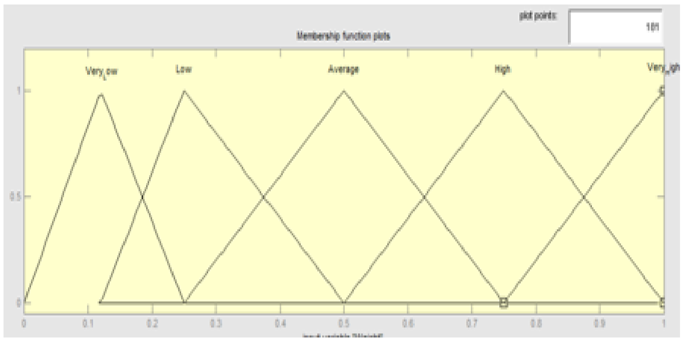

Fig. 4: Membership Function for Fuzzy Weight

3.1.2 Defuzzification Process:

Defuzzification process converts fuzzy output into crisp output using centroid method [8]. The centroid formula is as follows:

$$\textit{Centroid Formula} \quad z^* = \frac{\int \mu(z).z\,.dz}{\int \mu(z).dz} \tag{4}$$

Where z^* - denotes defuzzified crisp output,
z denotes the value on the x-axis and
µ (z) denotes the membership function.

Procedure

The following is the procedure for computing the complexity weight for each weighting scale factors such as Very Low, Low, Average, High, and Very High for the requirement complexity attributes.

Step#1: Assign fuzzy weight and fuzzy ratings for each requirement complexity attributes and for each criteria.

Step#2: Take the average weight and average ratings based on multi-criteria decisions under each requirement complexity attribute

Step#3: Take the average weight as fuzzy Input and average ratings as fuzzy output

Step#4: Create the Fuzzy rules based on IF-THEN rules.

Step#5: Apply Defuzzification process using centroid formula given in equation 3 in order to get the crisp output or crisp weighting factor for different weighting scales.

Functional Requirement Complexity Weight

Functional Requirement Complexity weight is based on the number of sub processes and its size [1].

If Size increases then software complexity increases. The sub process size in terms of KLOC can be fuzzified ratings in the range of Very Low (VL) to Very High (VH) as [< 0.3 (VL); 0.3 to 0.5 (L); 0.5 to 1 (M); 1 to 3 (H); >3 (VH)]. Apply the procedure steps from 1 to 3mentioned above for finding the fuzzy weighted average for each weighting scale factors.

Table 3: Weighted average for Functional Requirement Complexity

Size Criteria	D1	D2	D3	Avg. Weight	Avg. Rating
Size <0.3	VL	VL	L	(0.04,0.16,0.33)	(0,0.1,0.3)
0.3 – 0.5	L	L	L	(0.12,0.25,0.5)	(0.1,0.3,0.5)
0.5 -1	M	L	M	(0.21,0.42,0.67)	(0.33,0.55,0.75)
1-3	H	M	H	(0.42,0.67,0.92)	(0.5,0.7,0.9)
>3	VH	VH	VH	(0.75,1,1)	(0.7,0.9,1)

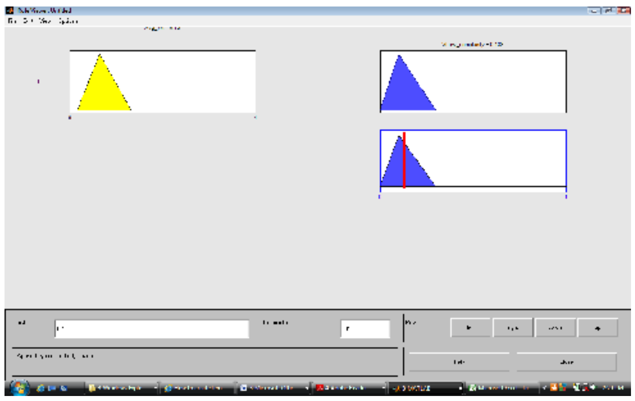

Fig. 5. Defuzzification Process for Very Low Complexity using MATLAB Tool.

Now apply the step3 to 5 in procedure using MATLAB Tool:

The fuzzy input value as Very Low average weight (0.04,0.16, 0.33) and fuzzy output value as Very Low (0, 0.1,0.3) which gives the defuzzified value or crisp output as 0.133.
The fuzzy rule is,

IF Sub Process size is very Low, THEN Output Complexity is Very Low.
Hence the Very Low Complexity weight for Functional Requirement Complexity Attribute is 0.13 which is depicted in Fig. 5.Similarly by applying the same procedure from steps 3 to 5 for other weighting factors, the result will be following crisp output.

Low Complexity Weight	– 0.3
Average Complexity Weight	– 0.53
High Complexity Weight	– 0.72
Very High Complexity Weight	– 0.87

Non-Functional Requirement Complexity

Non-Functional Requirement Complexity weight is based on the importance of quality characteristics [8].

If Quality Characteristics importance increases then software complexity increases. The fuzzified ratings will be in the range of Very Low (VL) to Very High (VH) as [< 0.3 (VL); 0.3 to 0.5 (L); 0.5 to 0.7 (M); 0.7 to 0.85 (H); >0.85 (VH)]. Apply the procedure steps from 1 to 5 mentioned above for finding the Complexity weight for each weighting scale factors.

Very Low Complexity Weight	– 0.21
Low Complexity Weight	– 0.3
Average Complexity Weight	– 0.5
High Complexity Weight	– 0.75
Very High Complexity Weight	– 0.9

Input Output Complexity

Input Output complexity weight depends on the number of Input and output for sub process functionality [1]. If number of input/output increases then software complexity increases. The fuzzified ratings will be in the range of Very Low (VL) to Very High (VH) as [<5 (VL); 5 to 10 (L), 10 to 20 (M); 20-50(H); >50(VH)]. Applying the procedure steps, the below result will be obtained.

Very Low Complexity Weight	– 0.21
Low Complexity Weight	– 0.3
Average Complexity Weight	– 0.57
High Complexity Weight	– 0.8
Very High Complexity Weight	– 1

File Complexity

File Complexity Attribute depends on the number of data storage files used in sub process functionality [1]. If number of files increases then software complexity increases. The fuzzified ratings will be in the range of Very Low (VL) to Very High (VH) as [< 10 (VL); 11 to 20 (L); 20 to 50 (M); 50 to 90 (H); >90 (VH)]. Apply the procedure steps from 1 to 5 mentioned above for finding the Complexity weight for each weighting scale factors.

Very Low Complexity Weight	– 0.27
Low Complexity Weight	– 0.4
Average Complexity Weight	– 0.67
High Complexity Weight	– 0.8
Very High Complexity Weight	– 1

Interface Complexity

Interface Complexity Attribute depends on the number of external interfaces namely user, hardware, software and communication used [21]. If number of interfaces increases then the software complexity increases. The fuzzified ratings will be in the range of Very Low (VL) to Very High (VH) as [1(VL); 1 to 2 (L); 2 to 5 (M); 5 to 8 (H); >8 (VH)]. Apply the procedure steps from 1 to 5 mentioned above for finding the Complexity weight for each weighting scale factors.

Very Low Complexity Weight	- 0.1
Low Complexity Weight	– 0.3
Average Complexity Weight	– 0.6
High Complexity Weight	– 0.9
Very High Complexity Weight	– 1

Summarizing the complexity weight factors for each requirement complexity attribute as in below Table 4:

Table 4: Complexity Weight Table

Attribute	Very Low	Low	Average	High	Very High
Functional	0.13	0.3	0.53	0.72	0.87
Non-Functional	0.21	0.3	0.5	0.75	0.9
Input Output	0.21	0.3	0.57	0.8	1
File	0.27	0.4	0.67	0.8	1
Interface	0.1	0.3	0.6	0.9	1

3.2 Unadjusted Complexity Point (UCP)

Once the requirements are baseline, map each requirement and its sub processes under each requirement complexity attribute based on weighting scale factors such as Very Low, Low, Average, High and Very High. Multiply the complexity weight and the total number of requirements in each weighting scale which results in unadjusted Complexity Point. Unadjusted Complexity Point (UCP) can be computed as follows:

$$UCP = \sum_{i=1}^{5} \sum_{j=1}^{5} W_{ij} C_{ij} \qquad (5)$$

Where, Wij - Weight of each complexity attribute for row j and column j.
C_{ij} - Number of each function feature with complexity weight.

3.3 Adjusted Complexity Point (ACP)

The Adjusted Complexity Factors needs to be evaluated for finding the Adjusted Complexity Point. The adjustment factors are 1.Communication and Clarification, 2.Understanding

Requirements Document Review 3.Test Case Document Review. The values of each factor are weighted on a scale of 0-5 where 0 – Not Present, 1 – Minor Influence to 5 – Strong Influence. The Adjustment Complexity Point can be computed as below:

$$ACP = \sum (F_i) \qquad (6)$$

Where, F_i - denotes the weighted scale of the adjustment factor.

3.4 Complexity Point (CP)

From Equation 4 and 5, the total complexity point can be computed as follows:

$$\text{Total CP} = UCP + ACP \qquad (7)$$

4.0PREDICTING SOFTWARE REQUIREMENTS STABILITY BASED ON COMPLEXITY POINT MEASUREMENT

This paper proposes the Complexity Point Measurement Model to measure the software requirement changes during the software development life cycle (SDLC). Based on this measurement, Software Requirement Stability Index can be predicted. This metric gives the stability factor of the requirements over a period of time, after the requirements have been mutually agreed and baselined.

The requirement change can be classified as

1. Add a new Functionality
2. Modify the Existing Functionality
3. Delete the obsolete Functionality

The Complexity Point Change can be represented as follows:

$$\Delta CP_{CHANGE} = \int_{i}^{i+1} CPdx \rightarrow CP(i+1) - CP(i) \qquad (8)$$

The related complexity point changes for add, update, delete functionality can be ΔCP_{ADD}, ΔCP_{UPDATE} and ΔCP_{DELETE} respectively. Compute the Complexity point change (Add, Update and Delete) as follows:

ΔCP_{ADD} = **Number of Complexity Point Added / Total Number of Initial Complexity Points)** *at that timestamp* (9)

ΔCP_{UPDATE} **= (Number of Complexity Point Modified/Total Number of Complexity Point)** *at that timestamp* (10)

ΔCP_{DELETE} **= (Number of Complexity Point Deleted/ Total Number of Initial Complexity Point)** *at that timestamp* (11)

The Total Number of complexity point changes during the particular timestamp is given by

$$\Delta CP_{CHANGE} = \Delta CP_{ADD} + \Delta CP_{UPDATE} + \Delta CP_{DELETE} \tag{12}$$

The Cumulative Number of Complexity Point Changes during the SDLC is given by

$$Cumulative\ \Delta CP_{CHANGE} = \sum_{i=1}^{i=n} \Delta CP_{CHANGE}\, i \tag{13}$$

Hence the Requirements Stability Index (RSI) based on Cumulative Complexity Point Changes is given by

$$RSI\ based\ on\ Cumulative\ \Delta CP_{CHANGE} = (CP + Cumulative \Delta CP_{CHANGE}) / CP \tag{14}$$

Where CP - denotes the total number of Initial Complexity Points
Cumulative ΔCP_{CHANGE} denotes the Cumulative number of Complexity Point changes during SDLC.

5.0 EXPERIMENTAL RESULTS AND DISCUSSION

Software Requirements Stability Tracker (RST) Tool was developed to track the stability of requirements and to find the overall stability of the project. The requirements gathered during the elicitation phase will be the basic input to the RST Tool. The RSI gives developers a means of continuing to document the requirements as they change throughout the development process and to monitor deviations from those originally specified.

The sample set of baseline requirements and requirements modification are gathered from sample customer module and given as input to RST tool. The results are depicted in Fig. 6 and Fig. 7.The following Fig. 6 depicts the Total Number of Requirement changes during the SDLC. Here, the X-axis values represent the timestamp (month) and Y-Axis value represents the Total number of Requirements and Requirements Add, update, delete in that particular timestamp (month).

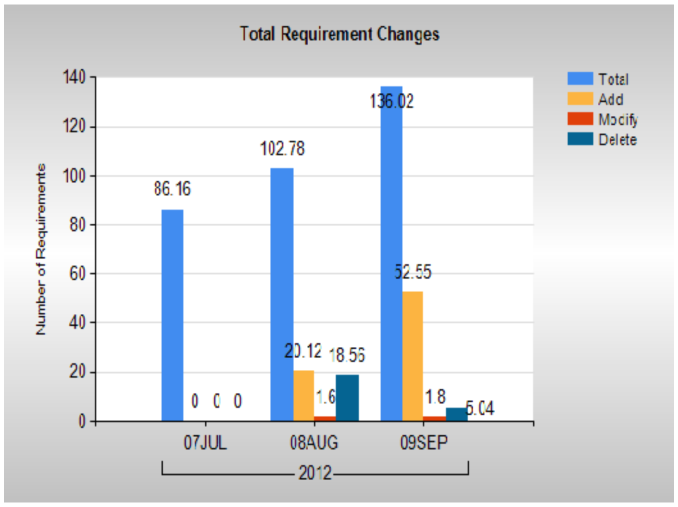

Fig. 6: Total Number of Requirements and Changes in Customer Module

The following Fig. 7 depicts the Requirements Stability Index based on the Software Complexity Point Measurement. Initially the Requirements Stability Index value will be one as the value of cumulative change will be zero initially. This is represented as "RSI" in Fig. 7. When changes come into the project, the Requirements Stability Index value will be changed. This is represented as "Change" in Fig 7.

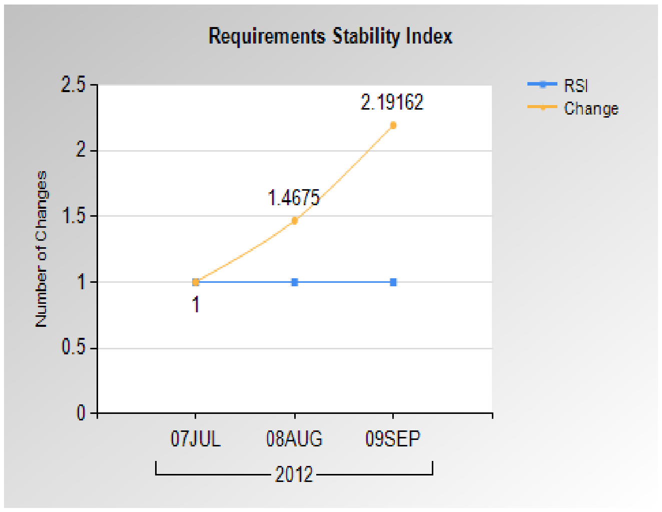

Fig 7. Requirements Stability Index Graph

From the sample data of software project "Customer Module", the following inference has been obtained during the period Jul'2012 to September'2012 as in Table 5 for customer module.

Table 5: Percentage of Requirements Stability for customer Module

Month	RSI	% of Requirement Stability= (100/RSI)%
July	1.000	100%
August	1.146	87.2%
September	2.191	45.6%

6.0 CONCLUSION

This research paper discusses the importance of measuring the requirements changes for the lack of instability in the Requirements. The prediction model for Requirements Stability approach provides the solution for measuring the requirements changes based on the Complexity Point Measurement Model. The scope of this research paper concentrates only on ongoing developing software projects. fIn future, the Measurement model can be developed for maintenance and transition projects based on different complexity attributes and different adjustment factors.

REFERENCES

[1] Ashish Sharma, D.S. Kushwaha, "A Complexity Measure Based on Requirements Engineering Document", Journal of Computer Science and Engineering, Volume 1, Issue 1, May 2010.

[2] Ashish Sharma and Dharmender Singh Kushwaha, "Applying Requirement Based Complexity for the Estimation of Software Development and Testing Effort", ACM SIGSOFT Software Engineering Notes, Volume 37, Issue 1, January 2012.

[3] Er. Kailash Aseri, "A Mathematical Study of Fuzzy Logic Techniques in Software Engineering Measurements", International Journal of Computer Science & Engineering Technology (IJCSET), Volume 3, Number 4, April 2012.

[4] Ghazal Keshavarz, Dr. Nasser Modiri, Dr. Mirmohsen Pedram, "Metric for Early Measurement of Software Complexity", International Journal on Computer Science and Engineering (IJCSE), Volume 3, Number 6, June 2011.

[5] Hua-Yang Lin, Ping-Yu Hsu Gwo-Ji Sheen, "A fuzzy-based decision-making procedure for data warehouse system selection", Expert Systems with Applications Journal, Volume 32, pp. 939-953, 2007.

[6] Iman Attarzadeh, Siew Hock Ow, "Improving the Accuracy of Software Cost Estimation Model based on a New Fuzzy Logic Model", World Applied Sciences Journal, Volume 8, Number 2, pp. 177-184, 2010.

[7] Iman Attarzadeh, Siew Hock Ow, "A Novel Algorithmic Cost Estimation Model Based on Soft Computing Technique", Journal of Computer Science, Volume 6, Number 2, pp. 117-125, 2010

[8] Jagat Sesh Challa, Arindam Paul, Yogesh Dada, Venkatesh Nerella,Praveen Ranjan Srivastava and Ajit Pratap Singh, "Integrated Software Quality Evaluation: A Fuzzy Multi-Criteria Approach", Journal of Information Processing Systems, Volume 7, Number 3, September 2011.

[9] Malik Qasaimeh, Alain Abran, "Extending Extreme Programming User Stories to Meet ISO 9001 Formality Requirements", Journal of Software Engineering and Applications, Volume 4, pp. 626-638, Nov. 2011.

[10] Martin Monperrus, Benoit Baudry, Joël Champeau,Brigitte Hoeltzener, Jean-Marc Jézéquel, "Automated Measurement of Models of Requirements", Software Quality Journal, Springer, Online Edition, 2011.

[11] Mohammad Azzeh, Daniel Neagu, Peter I. Cowling, "Analogy-Based Software Effort Estimation using Fuzzy Numbers", Journal of Systems and Software, Volume 84, Issue 2, Pages 270-284, Feb 2011.

[12] Navdeep Kaur, Maninderpal Singh, "A Fuzzy Logic Approach to Measure the Precise Testability ndex of Software", International Journal of Engineering Science and Technology (IJEST), Volume 3, Number 2, February 2011.

[13] Prasad Reddy P.V.G.D, Sudha K.R , Rama Sree P, Ramesh S.N.S.V.S.C "Fuzzy Based Approach for Predicting Software Development Effort", International Journal of Software Engineering (IJSE), Volume 1, Issue 1, 2010.

[14] Praveen Ranjan Srivastava, "Optimal Software Release Using Time and Cost Benefits via Fuzzy Multi-Criteria and Fault Tolerance", Journal of Information Processing Systems, Volume 8, Number 1, March 2012.

[15] Praveen Ranjan Srivastava, Sirish Kumar, A.P. Singh, G. Raghurama, "Software Testing Effort: An Assessment Through Fuzzy Criteria Approach", Journal of Uncertain Systems, Volume 5, Number 3, pp. 183-201, 2011.

[16] Rajesh Kumar, P.S.Grover, Avadhesh Kumar, "A Fuzzy Logic Approach to Measure Complexity of Generic Aspect-Oriented Systems", Journal of Object Technology, Volume 9, Number 3, May-June 2010.

[17] Sarah Maadqwy, Akram Salah, "Measuring Change Complexity from Requirements: A Proposed Methodology", IMACST: Volume 3, Number 1, Feb 2012.

[18] Dr. Sohail Asghar, Mahrukh Umar, "Requirement Engineering Challenges in Development of Software Applications and selection of Customer- off- the- Shelf(COTS) Components", International Journal of Software Engineering(IJSE), Volume 1, Issue 2, 2010.

[19] Sumeet Kaur Sehra, Yadwinder Singh Brar, Navdeep Kaur, "Multi-Criteria Decision Making Approach for Selecting Effort Estimation Model", International Journal of Computer Applications, Volume 39, Number 1, January 2012.

[20] Vahid Khatibi, Dayang N. A. Jawawi, “Software Cost Estimation Methods: A Review”, Journal of Emerging Trends in Computing and Information Sciences, Volume 2, Number 1, January 2011.

[21] Yogesh Singh, Sangeeta Sabharwal, “A Systematic Approach to measure the problem Complexity of Software Requirements Specifications of an Information Systems”, Information and Management Sciences Journal, Volume 15, Number 1, pp. 69-90, 2004.

Authors

D.Francis Xavier Christopher received his B.Sc., in 1996, M.Sc., in 1998 from Bharathiar University, Coimbatore .He obtained his M.Phil, in the area of Networking from Bharathiar University, Coimbatore in 2002. At present he is working as a Director, School of Computer Studies in RVS College of Arts and Science, Coimbatore. His research interest lies in the area of Software Engineering.

Dr.E.Chandra received her B.Sc., from Bharathiar University, Coimbatore in 1992 and received M.Sc., from Avinashilingam University, Coimbatore in 1994. She obtained her M.Phil., in the area of Neural Networks from Bharathiar University, in 1999. She obtained her PhD degree in the area of Speech recognition system from Alagappa University Karikudi in 2007. At present she is working as a Director at School of Computer Studies in SNS Rajalakshmi College of Arts & Science, Coimbatore. She has publ ished more than 20 research papers in National, International journals and conferences. She has guided for more than 30 M.Phil., research scholars. At present guiding 5 Ph.D research scholars. Her research interest lies in the area of Data Mining, Artificial intelligence, neural networks, speech recognition systems and fuzzy logic. She is an active member of CSI, Currently management committee member of CSI, Life member of Society of Statistics and Computer Applications.