ISSN: 2348 - 2117

International Journal of Engineering Technology and Computer Research (IJETCR)
Available Online at www.ijetcr.org
Volume 3; Issue 5; September-October-2015; Page No. 72-77

A Hybrid Intelligence System for Risk Estimation in Software Project

[1]Shreya Rani, [2]Dr. Savita Shivani

[1]Department of Computer Science & Information Technology, Gyan Vihar School of Engineering and Technology, Jaipur, India

Shreyarani2010@gmail.com

[2]Department of Computer Science & Information Technology, Gyan Vihar School of Engineering and Technology, Jaipur, India

Savitashivani@gmail.com

Abstract

The aim of this thesis is to investigate fuzzy parameters that may cause risk in software project and further leads to different issues in the product. In previous research the major factors for arising risk in software projects was taken into consideration. But risk which may create flaws does not remains constant to every software projects. This thesis first examines the fuzzy risk parameters in mainly five categories of risk common to all software projects by using fuzzy logic. In second stage the dataset of five categories of risk are provided as input parameters to neural network for the prediction of overall risk probability in software projects with the software project risk of NASA'93 dataset. Finally the result so obtained is the probability of overall risk in software projects lying between low and high or 0 and 1.

Keywords: *Software project, risk contingency, Project planning, Fuzzy logic, Neural network, Risk probability*

1. INTRODUCTION

One of the major concerns of software industries is to develop invulnerable and secure software. A software project is successful if it is delivered within provided time constraints, budget and with the required quality. Any software can be estimated accurately by cost estimation, schedule estimation, effort estimation, size estimation as well as risk estimation. However risk analysis is one of the major issues in software project management. Risk is a situation which involves an exposure to danger, harm, loss and vulnerability. Risk is the future happenings which after the development become flaws. In order to mitigate risk developers should integrate security at all stage of software development life cycle. Risk management in general means reducing uncertainty at every stage of software development life cycle such as uncertainty in customer requirement, indescribable nature of product, complexity of the process and obscure result of the project.

The primary objective of software team is to manage risk and avoid it at any cost. But eventually not all risk can be avoided so risk contingency plan is needed to control issues. To analyze risk software team member must quantify the level of uncertainty and degree of loss. On the basis of level of uncertainty and degree of loss different categories of risks are classified:

i. Project risk
ii. Technical risk
iii. Business risk

Sometimes risk becomes issue when project schedule crosses its deadline, budget and the cost increases. Thus, in this paper only project risk is considered, as complexity, size and level of uncertainty of the project is responsible for project risk.

It determines fuzzy values of above five risk parameters using vague properties with the help of fuzzy logic which also causes some risk percentage in the software project. The overall risk in software project in terms of linguistic variables that is low, moderate and high. Training is provided with the NASA'93 dataset of different risk type using neural network. The intelligent system is able to detect the overall probability of risk.

2. LITERATURE SURVEY

Every software projects have unique scope and challenges therefore same technology and tool can't be used, as a result of which risk aroused are also different [1], which are software projects are complex and abstract in nature, unclear initial and final requirements, user interface design of software, previous concept is applied again and again, changes are required to be done easily and inevitable.

Bohem's risk management introduced that the risk can be minimized in software project planning phase in software development life cycle phase [2]. However it was very difficult for the project planner to convince the customer to use this model because it deals with high risk project and the success of the project is highly based on expertise [3].

Corresponding author: Shreya Rani

To identify the risk in the project Jones's risk management found most influencing factors [4] such as time pressure, inaccurate measurement & cost estimation, delay in final assessment of the product, low quality software developed, low production per project & change in the tools and technologies. He gave method for prediction of risk by pattern recognition as well as provided the list of typical risks. But he didn't have any explanation of how to overcome from risk in the software project.

Karolak's risk management introduced software engineering risk model (SERIM) as just in time strategy for managing risk [5]. Risk identification, Risk strategy and planning, Risk assessment, Risk mitigation and avoidance, Risk reporting, Risk prediction.

Risk identification is key factor which is totally dependent on the human efficiency. So, it is rarely used by the developer as it is very tough to implement in the entire project during SDLC.

In 1989 Robert Charette gave another method of risk analysis and management in software engineering [6]. In 1997, Charette in a case study proposed that the risk arise during the development phase are easier to solve as compared to the risk arise during maintenance phase [7]. Risk identification, Risk estimation, Risk evaluation, Risk planning, Risk control, Risk monitoring. In implementation phase the success of the project rely totally on expertise and the prior experience of the programmer.

According to Hall's risk management the software risk was placed into three classes which are

- Software project risk
- Software process risk
- Software product risk

It is difficult to implement along with tools which analyses, identify, control and track the risk as well as it is highly dependent on the expertise and the skills of the developer [8]. The SEI support SE community and provides SE principle and practices in the area of software development, security, system design, process management and risk management [9].

In software risk management according to the basic assumptions, the SEI compute the risks and with the help of which a good quality software development is obtained with the good quality software product [10]. Ray Madachy proposed expert-COCOMO model which analyses and detect the input anomalies for effort estimation in software project and it is extension of COCOMO [11].

3. PROPOSED MODEL

Overall risk in software project is directly proportional to the overall cost of the project. The non linear risk level and cost multiplier is totally dependent on the size of the project. Thus the overall project risk is based on the following equation

$$\sum_{j=0}^{\#categories} \cdot \sum_{i=0}^{\#risk\ category} risk\ level(ij) * project\ size(ij)$$ 12]

The above equation helps the neural network in training to provide the overall target risk.

The intelligent system predict the overall risk in the software projects as a linguistic variable that is low, moderate and high as well as project size is measured in kilo lines of code (KLOC).

The proposed model uses two technologies which are as follows:

i. Fuzzy Logic
ii. Neural Network

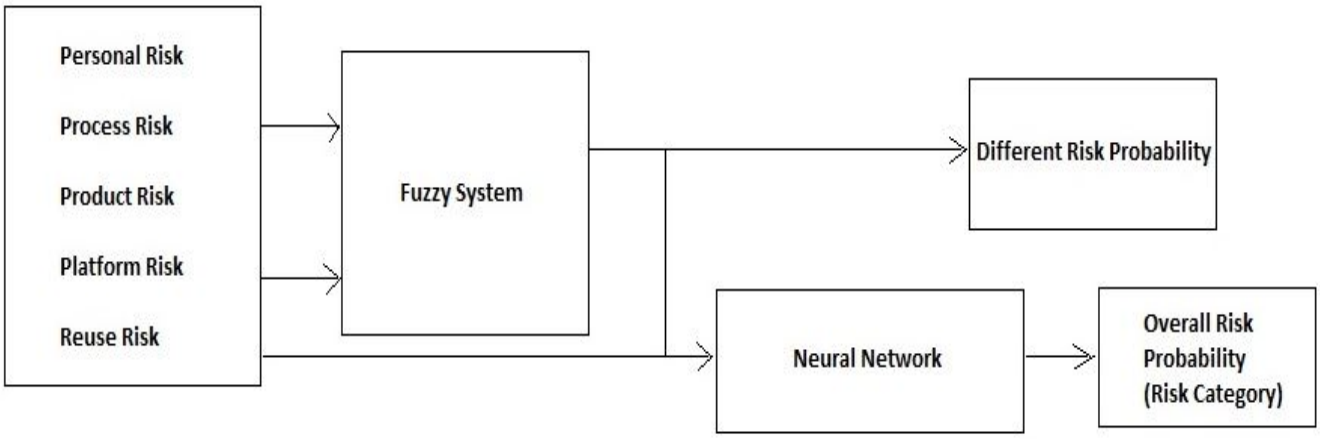


Fig 1: Flow Diagram of probability of overall risk estimation in software projects

Fuzzy Logic firstly examines the exact degree of different categories of risk in the software projects. It is the responsibility of project manager to consider each and every vague parameter also for the successful completion of the software project. The risk contingency for those vague parameters are measured exactly with the help of fuzzy system.

Secondly neural network is trained five different risk parameters. Training includes NASA'93 dataset to judge the overall risk contingency in the project due some vague parameters.

Fuzzy Logic Implementation

It is logic which deals with accuracy and the result so obtained is precise in Boolean logic between 0 and 1 or true and false. The membership function decides the level of accuracy and exactness in the result. The input to fuzzy system is given in the form of linguistic variables that is low, moderate, high and very high. With the help of those linguistic variables the membership function can estimate the level of risk contingency. This paper proposed the system for risk estimation using the following risk generating factors

1. Personal risk

It depends totally on the developer how much efficient he is in the work. Vague personal parameters which create risk in some percent criteria are as follows:

i. Programmer's experience

ii. Language efficiency based on programmer's qualification

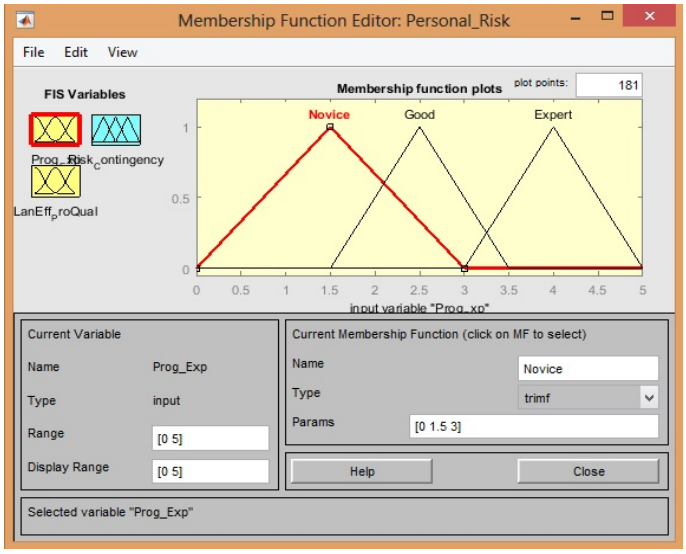


Fig 2: Membership function plot of personal risk contingency

2. Process risk

There are lots of factors on which process risk depends such as multisite development, precedentedness, development architecture risk resolution, and flexibility in development, process maturity and increment in development. But there are some vague parameters which must be taken into consideration. Those are tool flexibility as well as team cohesion.

i. Tool flexibility

ii. Team cohesion

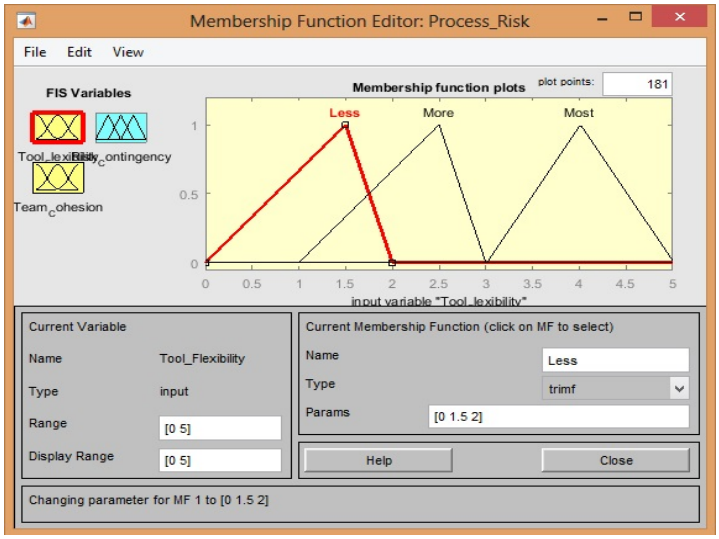


Fig 3: Membership function plot of process risk contingency

3. Product risk

The software product being developed also consists of many complexity of different type. Depending on the software different risk can arise in the product such as software size, product complexity, database size for back end processing, documentation of the software in project planning phase, reliability of software product. Two vague properties are taken as follows

i. Software complexity

ii. Reliability on software

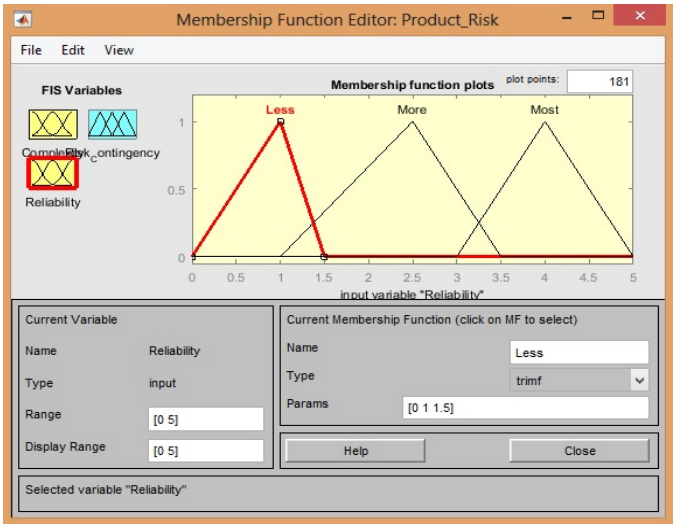


Fig 4: Membership function plot of product risk contingency

4. Platform risk

The architecture on which software is being developed can create platform risk. Infrastructure in the software company is the foundation factor for establishing risk awareness in the organization. There are four infrastructural elements which necessary to be considered in performing risk management which is organization, man power, resource and results. It also depends on execution time of software and main storage constraints

i. Execution time
ii. Database size

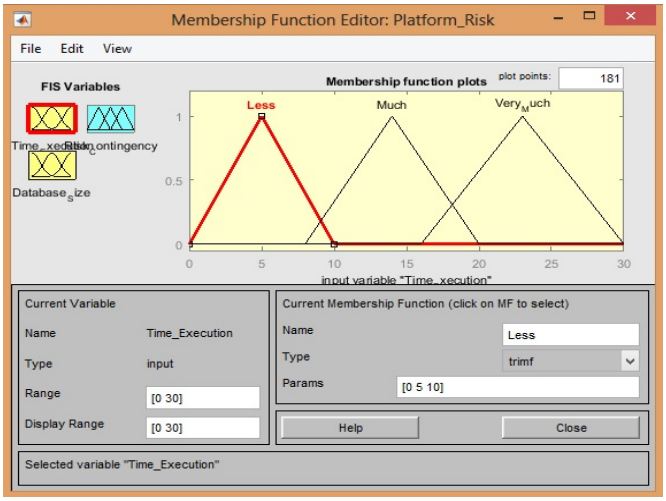


Fig 5: Membership function plot of Platform risk contingency

5. Reuse risk

There is much software which cannot be used again and again. If total lines of code is copied to new project and modified, then it is main cause of reuse risk. But not all software functionality can be inherited in further projects.

i. Reusability risk
ii. Software quality

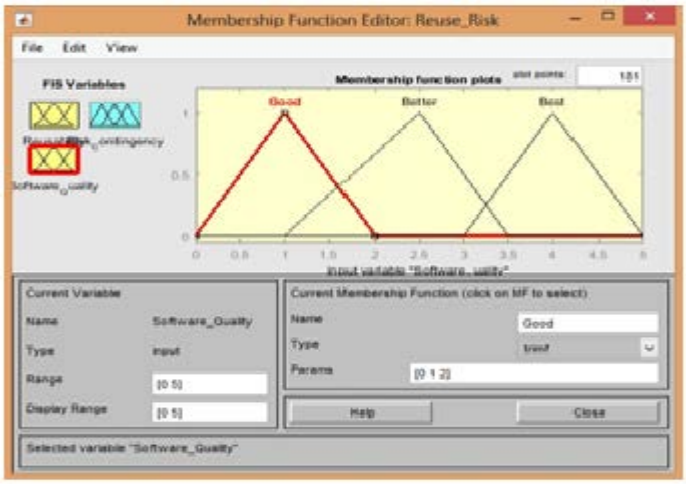
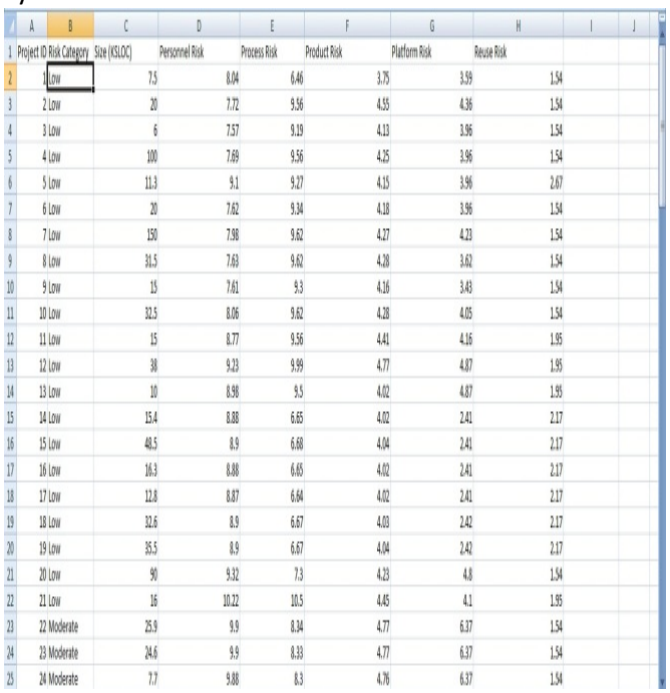


Fig 6: Membership function plot of Reuse risk contingency

Neural Network Implementation

Neural network is an intelligent system having the capability to train the neurons and make it learnt with the help of different risk datasets. The dataset so provided to train the neurons is taken from NASA website. The NASA dataset of different risk type is taken to estimate risk in software project. With the help of estimated risk measures neural network is used to predict overall risk in any software.



	Project ID	Risk Category	Size (KSLOC)	Personnel Risk	Process Risk	Product Risk	Platform Risk	Reuse Risk
2	1	Low	7.5	8.04	6.46	3.75	3.59	1.54
3	2	Low	20	7.72	9.56	4.55	4.36	1.54
4	3	Low	6	7.57	9.19	4.13	3.96	1.54
5	4	Low	100	7.69	9.56	4.25	3.96	1.54
6	5	Low	11.3	9.1	9.27	4.15	3.96	2.67
7	6	Low	20	7.62	9.34	4.18	3.96	1.54
8	7	Low	150	7.98	9.62	4.27	4.23	1.54
9	8	Low	31.5	7.63	9.62	4.28	3.62	1.54
10	9	Low	15	7.61	9.3	4.16	3.43	1.54
11	10	Low	32.5	8.06	9.62	4.28	4.05	1.54
12	11	Low	15	8.77	9.56	4.41	4.16	1.95
13	12	Low	38	9.23	9.99	4.77	4.87	1.95
14	13	Low	10	8.98	9.5	4.02	4.87	1.95
15	14	Low	15.4	8.88	6.65	4.02	2.41	2.17
16	15	Low	48.5	8.9	6.68	4.04	2.41	2.17
17	16	Low	16.3	8.88	6.65	4.02	2.41	2.17
18	17	Low	12.8	8.87	6.64	4.02	2.41	2.17
19	18	Low	32.6	8.9	6.67	4.03	2.42	2.17
20	19	Low	35.5	8.9	6.67	4.04	2.42	2.17
21	20	Low	90	9.32	7.3	4.23	4.8	1.54
22	21	Low	16	10.22	10.5	4.45	4.1	1.95
23	22	Moderate	25.9	9.9	8.34	4.77	6.37	1.54
24	23	Moderate	24.6	9.9	8.33	4.77	6.37	1.54
25	24	Moderate	7.7	9.88	8.3	4.76	6.37	1.54

Fig 7: NASA Dataset of different risk type categorized in three risk category

Training model of neural network can classify the probability of different risk category.

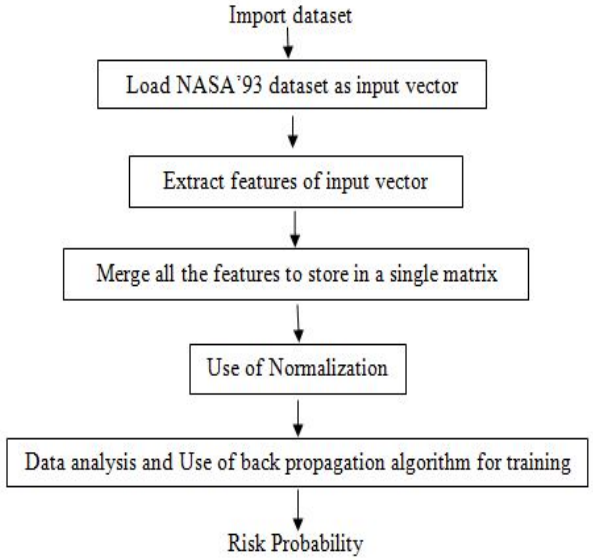


Fig 8: Risk evaluation model using neural network Back propagation algorithm

This algorithm works basically in similar fashion as perceptrons. The learning principle of back-propagation algorithm is applied to the hidden layer using gradient decent rule. Weight updating rule in back propagation algorithm is different from other updating rule as the activation of hidden unit is used instead of activation of input unit.

In feed forward fashion there are methods available for learning of weight. Learning process doesn't involve perceptron learning rule as it doesn't provide learning to the hidden neurons. Gradient descent rule is used to minimize the error. Backward propagation is done to adjust errors from output layer via hidden layer to input layer. Stepwise process used in back propagation algorithm

- Initialize the neural network with random weight
- For each number of training
- Provide training to input matrix and compute the output
- For all layers starting from input layer to the output layer
- Compare each of network output with the desired output until error is reduced
- Weight is adapted in the current layer.

4. RESULTS

The proposed software risk estimation model is able to detect risk of different type in software project using fuzzy parameters. The different risk parameters are used to as a whole to estimate the overall risk approximately using neural network. The training is done to provide the accuracy level of 94.92% and the observer error in the dataset is of 5.08%. The target or overall risk probability is estimated which is resulted in the form of linguistic variables in the software project.

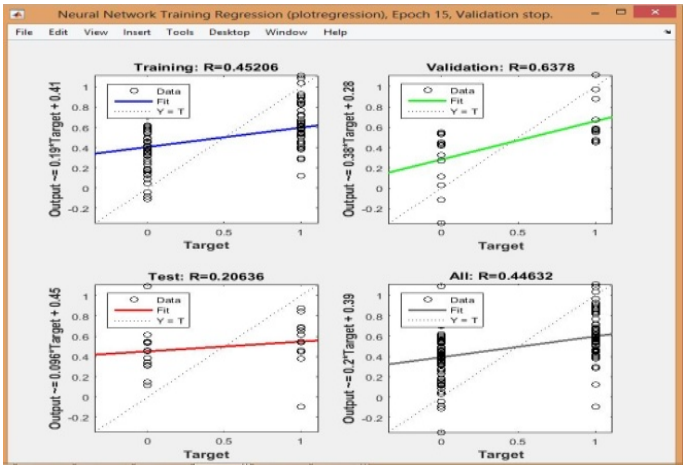


Fig 9: Regression graph after training of neural network

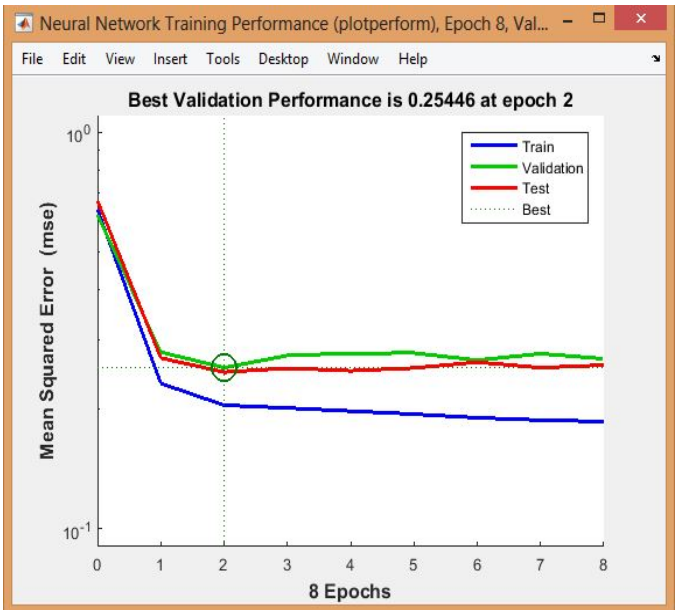


Fig 10: Performance plot after training of neural network

5. CONCLUSION AND FUTURE SCOPE

The aim of this research is to enhance the project planning phase by risk assessment at the start of software development life cycle.. This methodology is different from the other risk assessment process as it uses the vague factors of different type of risk parameters in software projects.

This model is being made using neuro-fuzzy approach which is beneficial to the project manager in the first phase of SDLC.

Accurate result is obtained using fuzzy techniques which takes the input in the form of linguistic variables. Risk contingency of overall project is measured by trained dataset of neural network which is totally dependent on risk category and size of the project in KLOC. Learning abilities in the model is provided to neural network with the help of NASA'98 dataset.

Limitation

The system estimate the risk in software project caused due to particular vague parameters. In this model not each and every type of risk can be estimated because it varies to different types of software project.

Future work

Software risk estimation is exciting area of research. Learning ability using soft computing can be enhanced in the system to detect almost every type of risk in software projects. Some additional parameters can be taken further for different project type such as development mode and some other parameters to improve risk contingency allowance in the model.

REFERENCES

1. Pressman R. S., Software Engineering – Practitioner's Approach, 6th Ed, McGraw Hill, NY, 2005, pp.644.
2. Boehm B. W., "Verifying and Validating Software Requirements and Design Specifications", IEEE Software, Jan 1984, pp.75-88.
3. Boehm B. W.,"Software Risk Management: Principles and Practices", IEEE Software, Jan 1991, pp.32-41.
4. Jones C., "Assessment and Control of Software Risks", Yourdon Press Prentice Hall, 1994, pp.47-60.
5. Karolak D. W., Software Engineering Risk Management IEEE Computer Society Press, 1996.
6. Charette R. N., "Software Engineering Risk Analysis and Management", McGraw Hill, 1989.
7. Charette R. N., "Managing Risk in Software Maintenance" IEEE Software, 1997, pp.43-50.
8. Hall E. M., "Managing Risk: Methods for Software Systems Development", Addison Wesley, 1998, pp.7.
9. Software Engineering Institute, http://www.sei.cmu.edu/solutions/solutions.cfm, access on March 2012.
10. Hall E. M., "Managing Risk: Methods for Software Systems Development", Addison Wesley, 1998, pp.52.
11. Madachy R., "Heuristic Risk Assessment Using Cost Factors", IEEE Software, May/June, Vol.14 No.3, 1997, pp.51-59.
12. Ekananta Manalif (April 2013), Fuzzy Expert-Cocomo Risk Assessment And Effort Contingency Model In Software Project Management, MS Thesis, The University of Western Ontario, London, Ontario, Canada