Software Metric and Back Propagation Techniques Applied in Fuzzy Logic

Zindhu S[1], Aswini G[2] and Anusha Prem I[3]

[1] Assistant Professor, Department of Computer Science,
St. Joseph's College of Arts and Science for Women, Hosur, Tamil Nadu
zindhuchandrandevi@gmail.com

[2] Assistant Professor, Department of Computer Science,
St. Joseph's College of Arts and Science for Women, Hosur, Tamil Nadu
aswini083@gmail.com

[3]Assistant Professor, Department of Computer Science,
St. Joseph's College of Arts and Science for Women, Hosur, Tamil Nadu
ianushaprem@gmail.com

ABSTRACT—*Software quality is one of the most important factors in the software development. It can be depends on many attributes. One of the best techniques in Fuzzy Logic is metrics and maintainability. This paper presents the application of fuzzy logic in software metrics. Software metric is the measurement of the software development process and product. It can be used as variables in the project management. The most common types of these models are predicting the development effort for a software system based on size, complexity, characteristics and metrics. There are many problems that have not been overcome using the traditional techniques of both formal and linear regression model. Once the problem faced by managers, who are using project metrics models is the elicitation of numerical inputs. These problems can be seen as collective failure of software measurement. The proposed techniques can help to overcome some of the difficulties by representing the imprecision in both input and output. This techniques especially fuzzy logic is investigated and some usable recommendation is made. Different levels of available information and desired precision can be used differently, mainly depends on current phase, although a single model can be used for consistency.*

Keywords— Fuzzy Logic, Neural Network, Backpropgation, Software Metrics

---

1. INTRODUCTION

Fuzzy Logic modeling technique has been shown to be a useful addition to the existing statistical and machine learning techniques used for software development. From theoretical reasons preferred to fuzzy logic in some circumstances, several papers have shown favorable empirical comparisons support by using metrics data sets to compare the predictive accuracy of this techniques. Fuzzy Logic modelling software has been developed for supporting project estimation process. The three choices of being able to use knowledge expert, linguistic inputs and producing linguistic output.

The level of commercial interest in using this techniques in anger grows it becomes necessary to provide well–document and replicable standard for its implementation. The most successful software metric model for effort estimation in Function Point Analysis [1] and a hallmark of this has been its carefully documented procedures, certificates and workshops. It would be premature to impose such restrictions on the use of fuzzy logic techniques. It does seem prudent to outline some general skeleton guidelines to encourage, inconsistent, goals of experiment and rigor. The selection of input and output precision levels based on the stage of the development life cycle.

2. FUZZY LOGIC IN SOFTWARE EFFORT ESTIMATION

2.1 A fuzzy model is construct two main properties

All Fuzzy logic offers a particular way to generate a keen mapping between input and output spaces. In software effort estimation techniques justify the decision of implementing this model. First one it is impossible to develop a precise mathematical model of domain [4]. Next one metrics only produce estimation of the real complexity. According to the previous assertion formulating a tiny set of naturals rules describes the interaction between the software development metrics and effort estimation could effortlessly reveal their intrinsic and wider correlations.

2.2 The Software Development Life-Cycle

Many alternative representations have been proposed for how software is, ought to be developed [5]. Different models exist for different types of system such as object-oriented system, where the idealized development process is considered by some to be fundamentally different to other types of development. Another uses of prototypes are customer reviews and external activities may differ from one organization to the next. Herewith consider the fundamental phases of development that are ubiquitous to almost all such models found in practice and the literature.

The models names are Analysis, Design, Coding, Maintenance Phases and Testing. Each phase is briefly below. So, these phases can also be shown graphically as in the Figure-1. It also indicates the common notion of iteration and feedback between consecutive phases. Table-1 shows the levels of precisions of development life-cycle.

2.2.1 Analysis

It is the first stage of the development process and starts with the problem being defined and initial user requirements being collected. Analysis phase include the construction of simple prototypes used to determine or fine-tune of the requirments. But, it should not involve any detailed design or real coding. This information beyond the high-level functions required of the system is available in the development life-cycle. Unfortunately, this is also the phase where planning is most crucial in terms of both time to delivery and financial cost estimation. Such information is necessary for contract negotiation and most strategic planning.

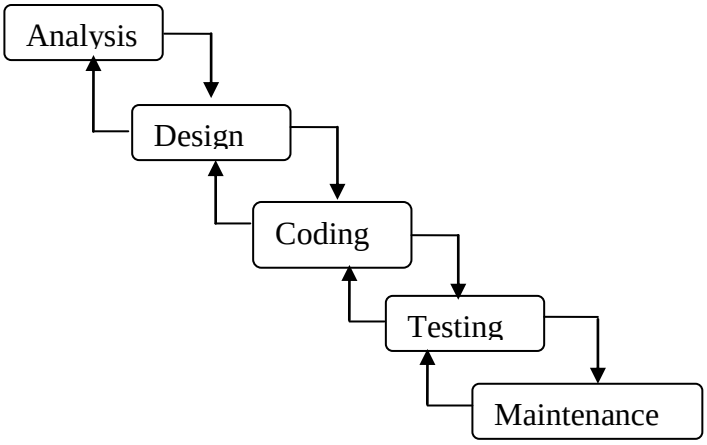


Figure 1: Stages in a generic life-cycle model

Design and Coding phases are the prediction of development effort and also estimates based on cost and duration. While other dependent variables are interest in the primary application of effort estimation. Other ideas can be easily generalized for other metric applications.

2.2.2 Design

The requirements must then be translated closer to the actual implementation. The system specifications are developed in this stage. For example – Entity Relationship Diagrams, Data Flow Diagrams, Structure Charts and Pseudo code routines. The system should be understood to a high degree at this stage with the coding stage involving the implementation of the necessary functions.

2.2.3Coding

Actual source code is writes in the coding phase. This phase includes generated code and automatic template code. It may also include more sophisticated prototypes that evolve into the final system.

2.2.4Testing

Testing can be performed concurrently in software development. It may occur mostly after the coding phase. When the system is tested metric models may be used for estimating testing efforts for predicting the number of defects remaining. It includes functionality and usability. This stage the system should be complete in terms of functionality so a large amount of information about it is available for modeling.

Phase	Inputs	Outputs
Analysis	Fuzzy label	Fuzzy label
Design	Fuzzy number	Fuzzy number
Coding	Crisp value (or) fuzzy number	Fuzzy number
Testing	Crisp value	Fuzzy number
Maintenance	Crisp value	Fuzzy number

Table 1: Levels of precision

2.2.5 Maintenance

When the system is modified to add new functionality or correct defects after it has been released to the customer some estimate of maintenance effort may be made. Unlike the effort estimates for the previous four phases. It can be combined all to give the total. But, the effort on maintenance is often treated separately. Estimates can be made for the entire maintenance process if this is sufficiently trivial. This phase can be treated as a new development process itself.

3. ADVANTAGE OF FUZZY LOGIC FOR SOFTWARE METRIC MODELS

Fuzzy logic modeling techniques offer several potential advantages over more traditional techniques for this metric model. This model already been discussed at considerable length in the literature.[2]

3.1 Data requirements

Fuzzy logic allows model development with little or even no data. It is consider as the given the problems with data gathering in software metrics. The collection of homogeneous data sets is complicated by rapidly changing technologies and reluctance for inter-organizational sharing of metrics data. Even within a single organisation there can be considerable pressure from programmers and managers against measurement collection.

3.2 Robustness

Software metric data sets are likely to contain unusual systems that result from a variety of causes and may reduce the generalisability of any empirically derived model [3]. Some problems includes different development practices, developer learning and non measured influences. By this developing models consider with expert involvement. This model can be interpreted and checked for reasonableness, some of the problems with non-representative data corrupting empirically tuned models can be reduced.

3.3 Organizational process learning and communication

The uses of fuzzy logic models provide an opportunity to learn from the resulting models that is less evident with regression and neural network model. A linguistically based model can also be a powerful communication tool.

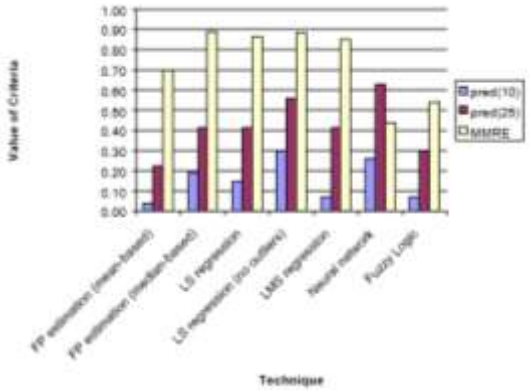


Figure : Comparision of Results

For example a programmer pointing out that complexity for a particular module is very high and it may be needing rework. This model is relatively easily understood by management there is a greater chance of management support, it is essential for the success of any metrics programs.

4 ESTIMATION METHODS

Software effort estimate can be described through a Prediction Interval (PI). This interval is based on a stated certainty level and contains a minimum and maximum effort value.

4.1 Selection of Estimation approach

The use of artificial neural network is a cross disciplinary field that integrates neuroscience, computer science and information science. Many neural network models, the back-propagation network display a strong learning ability using nonlinear models with a high fault tolerance. The evidence on differences in estimation accuracy of different estimation approaches and model suggest them. There is no more best approach. There are many ways to categorizing estimation approaches.

4.2 Expert estimation

This model is based on judgmental processes.

4.3 Formal estimation models

This model is based on mechanical processes. Formal estimation model is not tailored to a particular organization. It may be inaccurate. Uses of historical data are consequently crucial if one cannot be sure that the estimation models core relationship.

4.4 Combined-based estimation

This model is based on both judgmental and mechanical combination of estimates from different sources.

5 SOFTWARE METRIC MODELS FOR EFFORT PREDICTION

The size of a computer system could be measured in terms of the number of lines, screens, source codes and reports contained in the specification. The complexity may be measured in terms of the structure of system and object-oriented systems. Some other metrics are include in the quality of developers working on the system.

These descriptive measures of the system and development process are software metrics. The potential variables of interest here such as the required developer effort for programming and testing are also software metrics.

Such a model is shown in the below Figure.

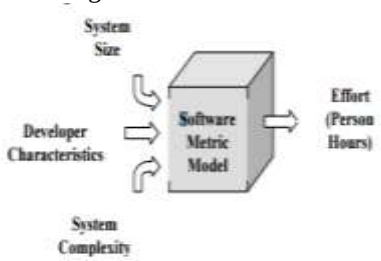


Figure 3: Software metric model for effort estimation

A software metric model may now be developed using data that contains some measures of system size and complexity. Many other variables could be added in this system. Such predictive models are usually developed using linear regression analysis is available on historical data. Predictions can then be made for new system development projects using the resulting model as the independent variables are estimated.

5.1 Software Measurement

Software metrics measure the following, such as design, source code, testing and productivity of an engineer. There are different points of views under which the software quality analysis can be performed and the main parameters are

i. Software Fault Analysis
ii. Software Quality Analysis
iii. Software Reliability Analysis

These parameters are involved in the prediction and process of software quality improvement and they also help us to analyze any software system under theoretical, conceptual and practical aspects.

6 METHODS APPLIED

6. 1 Error BackPropagation

The tuning of the network, so that we keep moving toward the point of minimum error as shown in figure.

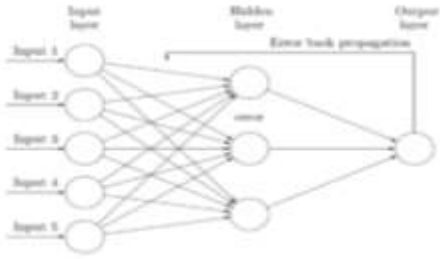
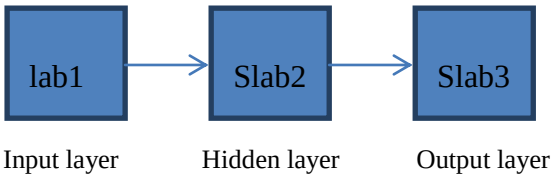
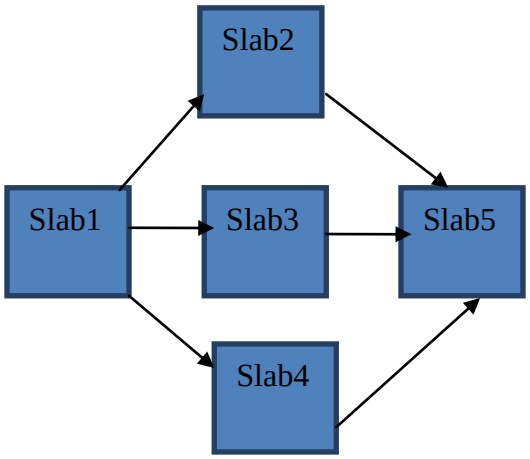


Figure 4 . Training of ANN based model using back propagation with supervised learning

ANN to predict software quality using (ii) General Regression Neural network (ii) Ward neural network. GRNN is a one-pass learning system. It's architecture and learning is parallel and very fast, accurate than the ward to predict the actual.



Slab- is a group of neurons with activation function. In this GRNN, input layer contains 71 neurons, hidden layer contains 3 neurons and output layer contain 1 neuron. Ward network is a back propagation technique with multilayer perception. Each neuron may have different activation function. There are three layer used here namely input layer, hidden layer and output layer.



6.2 Fault prediction in software metrics

Fault can be predicted by using Human Machine Interface. The effort predicted by changing number of lines. A change can be deletion or addition of a line.

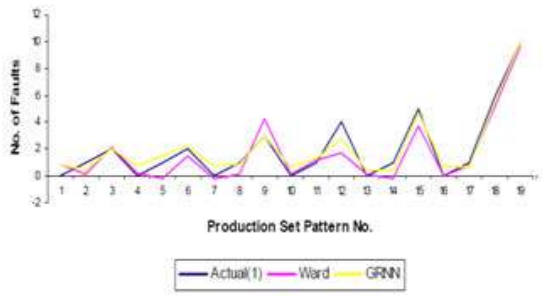


Figure 5:. No. of failure for HMI

Content change may be added or deleted in QES(quality evaluation system) used for prediction of line changes. Both prediction algorithm shows that GRNN is more.

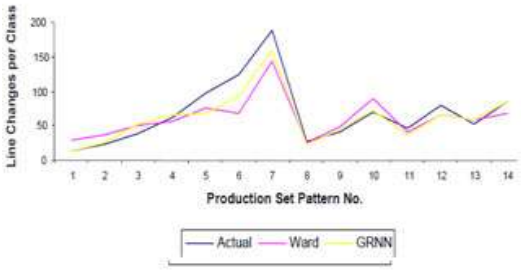
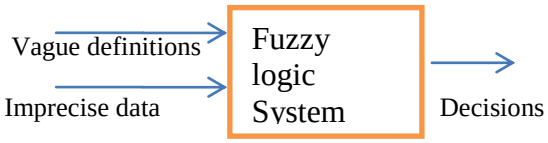


Figure 6: No. of lines changes per class for QES

6.*3 Fuzzy Backpropagation*

According to Mac Doneld and Gray, Fuzzy systems used for software metric analysis. It is only adaptive technique when the small data sets available. According to Singh et al, Fuzzy logic applied for development of testability measurement. Fuzzy logic makes wonders on imprecise inputs. The measurement definitions are vague. The data collected are also imprecise.

Diagram shows that system gives a precise output from imprecise input.



The precise output can be calculated by right threshold (approximately 0.2)

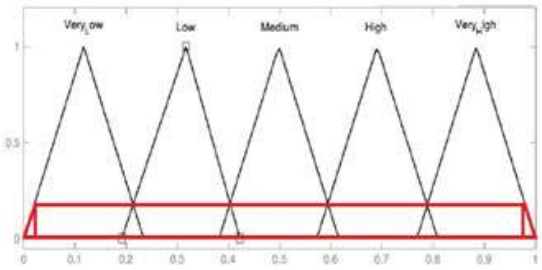


Figure 7: the Precise output

7 CONCLUSION

The tremendous technique is automation usage show that full automation is to happen. Metrics development and analysis is a slow process metric process always lags behind the core development process. In future, software development process will be fully automated. One of the biggest challenges is to keep rapid changes in software development environment. Fuzzy logic is well suited to a life-cycle approach to software metric modeling. It is a unique opportunity not previously available developed using measurements from primarily developed using measurements from a single phase in a project's life. The ensuing consistency. Economy and community make this an attractive modeling technique for most applications as effort estimation. The other advantages of data-free model building, more robotics and improved communication further enhance the opportunities from using fuzzy logic for software metrics.

8. REFERENCES

[1] N.E. Fenton and S. L. Pfleeger. Software Metrics: A Rigorous & Practical Approach PWS, 1997.
[2] A. Gray and S. MacDonell. Application of fuzzy logic to software metric models for development effort estimation.
[3] Y. Miyazaki, M. Terakado, K. Ozaki and N. Nozaki. Robust regression for developing software estimation models. Journals of System and Software.
[4] Grimstad, S., Jorgensen, M., Molokken-Ostvold, K., Software Effort Estimation Terminology. The Tower of Babel. Information and software Technology. Elsevier, s2005
[5] R.S. Pressman. Software Engineering: A.Practitioner's Approach. McGraw-Hill, fourth edition, 1997.
[6] S.G. MacDonell and A.R.Gray. Fulsome. A fuzzy logic modeling tool for software metricians.
[7] A.A. Moataz, O.S.Moshood, A.Jarallah, "Adaptive fuzzylogic-based framework for software developmenteffort prediction", Information and Software Technology.
[8] JaswinderKaur, Satwinder Singh, Dr. Karanjeet Singh Kahlon, PourushBassi,"Neural Network-A NovelTechnique for Software Effort Estimation", International Journal of Computer Theory and Engineering.
[9] Ali Idri and Taghi M. Khoshgoftaar& Alain Abran,"Can Neural Networks be easily Interpreted in Software CostEstimation", IEEE Transaction.
[10] Molokken K.; Jorgensen M., 2003. A review of software surveys on software effort estimation, Proceedings of IEEE International Symposium on Empirical Software Engineering