

01 Jan 2003

An Intelligent Early Warning System for Software Quality Improvement and Project Management

Xiaoqing Frank Liu

Missouri University of Science and Technology, fliu@mst.edu

Gautam Kane

Monu Bambroo

Follow this and additional works at: https://scholarsmine.mst.edu/comsci_facwork



Part of the [Computer Sciences Commons](#)

Recommended Citation

X. F. Liu et al., "An Intelligent Early Warning System for Software Quality Improvement and Project Management," *Proceedings of the 15th IEEE International Conference on Tools with Artificial Intelligence, 2003*, Institute of Electrical and Electronics Engineers (IEEE), Jan 2003.

The definitive version is available at <https://doi.org/10.1109/TAI.2003.1250167>

This Article - Conference proceedings is brought to you for free and open access by Scholars' Mine. It has been accepted for inclusion in Computer Science Faculty Research & Creative Works by an authorized administrator of Scholars' Mine. This work is protected by U. S. Copyright Law. Unauthorized use including reproduction for redistribution requires the permission of the copyright holder. For more information, please contact scholarsmine@mst.edu.

An Intelligent Early Warning System for Software Quality Improvement and Project Management

Xiaoqing (Frank) Liu

Gautam Kane

Monu Bambroo

Dept. of Comp. Sci.
Univ. of Missouri-Rolla
fliu@umr.edu

Dept. of Comp. Sci.
Univ. of Missouri-Rolla
gskz67@umr.edu

Dept. of Comp. Sci.
Univ. of Missouri-Rolla
mr bx97@umr.edu

Abstract

One of the main reasons behind unfruitful software development projects is that it is often too late to correct the problems by the time they are detected. It clearly indicates the need for early warning about the potential risks. In this paper, we discuss an intelligent software early warning system based on fuzzy logic using an integrated set of software metrics. It helps to assess risks associated with being behind schedule, over budget, and poor quality in software development and maintenance from multiple perspectives. It handles incomplete, inaccurate, and imprecise information, and resolve conflicts in an uncertain environment in its software risk assessment using fuzzy linguistic variables, fuzzy sets, and fuzzy inference rules. Process, product, and organizational metrics are collected or computed based on solid software models. The intelligent risk assessment process consists of the following steps: fuzzification of software metrics, rule firing, derivation and aggregation of resulted risk fuzzy sets, and defuzzification of linguistic risk variables.

system to reduce the risks. A few research projects are in preliminary stages for assessing individual risk factors directly based on a few software metrics. Although they have demonstrated clearly their benefits, many challenging issues remain to be resolved. Conflicting risk assessment results may be obtained based on multiple groups of metrics from multiple perspectives, and it is very difficult to reconcile them since they are crisp. The overall risk associated with the entire process or system from multiple types of risk factors often is not obtained. In addition, most of the existing works are directly based on quantitative measurements represented by numbers, which are sometimes inaccurate, unreliable, and incomplete. The quantitative measurements have a lot of uncertainty and their risk assessment may not be reliable. Human common sense and knowledge, which form the basis of any risk management exercise, are ignored.

On the other hand, fuzzy logic has found a lot of successful applications in risk assessment from financial markets, environment control, project control, to health care. It can help to detect risks as early as possible in an uncertain environment. More importantly, it allows us to use common sense and knowledge for risk detection. It provides a rich set of mathematical tools for assessing risks associated with software project management and software quality control.

1. Introduction

1.1 Background

It is reported that \$275 billion a year is spent on software projects, only 26 percent of software projects are completed successfully, and only 72 percent of projects are completed at all, successfully or not. One of the main reasons behind this unfruitful development is that it is often too late to correct the problems by the time they are detected. In order to decrease cost and improve quality, it is necessary for project managers and developers to visualize potential risks in advance. It clearly indicates the need for early warning about the potential risks. Many early warning systems are available in our society and in many other fields of engineering and found to be very beneficial; and the software industry also needs such a

1.2 Relevant work

Barry Boehm's work in software metrics, such as the COCOMO model and the spiral model has originated research in software risk assessment [BOE81,BOE89,BOE00]. The quantitative effort and cost estimation model has significant impact in software engineering. Several other research projects have demonstrated interesting results. NASA's approach of risk assessment directly based on software metrics shows promising results [HYAT96]. NASA's WISE Project Management System [NASA95] uses the Internet to provide the framework for issue management in software development. Enhanced Measurement for Early Risk Assessment of Latent Defects (EMERALD) [NORTEL96, NOTEL98], developed in the Nortel, consists of decision support tools to help software designers and managers to assess risk to improve software quality and

reliability in the area of defect analysis. At various points in the development process EMERALD predicts which modules are likely to be fault-prone. In a research paper [Luqi2000], a formal model is introduced to assess the risk and the duration of software projects, based on a few objective indicators that can be measured early in the process. The approach supports (a) automation of risk assessment, and (b) early estimation methods for evolutionary software processes. Project risks related to schedule and budget is addressed with the help of risk assessment model based on software metrics. The indicators used are requirements volatility (RV), complexity (CX), and efficiency (EF). In all the above systems, individual risk factors are assessed directly based on software metrics. Overall risk associated with the entire process or system often is not obtained. In addition, conflicting results may be obtained based on multiple groups of metrics from multiple perspectives, and it is very difficult to reconcile them since they are crisp. In addition, all of the above works are directly based on quantitative measurements represented by numbers, which are sometimes not very accurate, reliable, and complete. Human common sense and knowledge, which form the basis of any risk management exercise, are ignored.

In search for best predictive technique for effort estimation, comparison of neural network models, fuzzy logic models and regression models are made in a research project [GRAY97]. It concludes that neuro-fuzzy hybrids may be used for better estimation. Researchers have tried to incorporate the fuzzy logic for estimation in software development. The use of case based reasoning for software project management is explored in another recent research project [CVAS94]. The case indices are expressed using fuzzy sets. Risk assessment is done by employing fuzzy aggregation to evaluate the cases. Application of fuzzy clustering for software quality prediction is proposed in a paper, where data set is modeled by fuzzy clusters and fuzzy inference technique is applied to predict fault-prone modules [TMK00]. The drawbacks of use of traditional methods for risk assessment like checklists, risk matrix are discussed and a fuzzy expert system for early operational risk assessment is developed in another related research project [TMK02].

1.3 Our Approach

We develop an intelligent early warning system using fuzzy logic based on an integrated set of software metrics from multiple perspectives to make sponsors, users, project managers and software developers aware of many potential risks as early as possible. It has the potential to improve software development and maintenance by a great margin.

Various factors for risk assessment in software development can be measured quantitatively using software metrics. All the metrics represent measurements from many different perspectives. For example, 'Size' is associated with software artifact, whereas, 'Effort' is associated with development process. Similarly 'Productivity' is associated with individuals as well as organization units. Hence we use a 'dimensional analytic model' to organize the metrics from three different dimensions: Product, Process and Organization [FLIU99]. Separation of perspectives of different metrics helps to establish relationship between metrics from the same dimension. In addition we need to capture relationships of metrics across dimensions, as the software development is characterized by three dimensions as a whole. A set of fuzzy rules are developed based on individual metrics and their combinations from these three dimensions to identify risks.

The system is customizable. The thresholds for many inference rules of risks based on metrics differ a lot for different organizations. Hence it is not wise to have the thresholds hard coded in the system. As mentioned earlier, only prediction of risk is not enough. The root cause of the risk should be found out. Our system has trace-down capability for the identified risk. The risk level is projected over all the three dimensions. The faulty module, phase or group can be traced-down with the help of the dimensional analytic model.

2. System Architecture

The system is designed in such a way that it provides warning across all phases in software engineering cycles. The system architecture is shown in Fig.1. It contains following primary components: 1) Metric Database 2) Risk Knowledge Base 3) Dimensional Analytic Model 4) Intelligent Risk Assessment Engine and 5) Visual Warning Issuing System

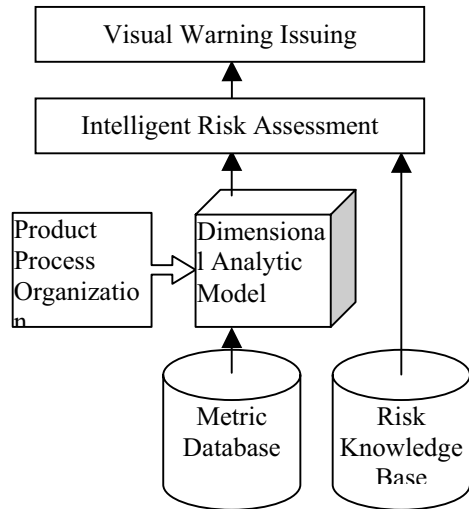


Fig 1: An Early warning system for software development and maintenance

2.1 Dimensional Analytic Model for Metrics Database

Risk should be identified based on objective data about software product, process and organization. Metrics database stores software metrics which are used for risk analysis and warning generation. The software metrics serve as base for intelligent risk assessment in software development and maintenance. The metrics database contains three types of metrics: 1) Product Metrics 2) Process Metrics and 3) Organization Metrics. Dimensional Analytic Model is used to visualize software development quantitatively as shown in the Fig.2.

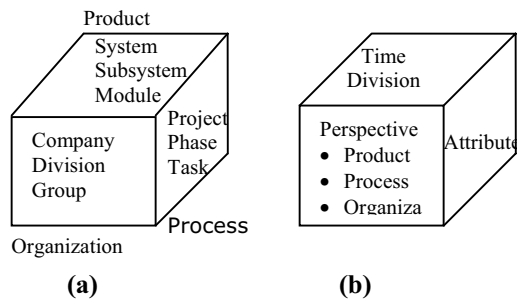


Fig. 2: Dimensional analytic model

As shown in fig. 2-(a), product dimension is divided into system, subsystem and module, process dimension is divided into project, process and task, and organization dimension is divided into company, division and group. Fig 2-(b) describes attributes that apply to all the

dimensions through their hierarchies. Using an attribute from each of the dimensions, it is possible to describe the state of a particular software artifact at a particular point in time. By relating time to each of the dimensions and their related attributes, it is possible to identify its trend for a particular attribute of an artifact from a particular dimension over a given period of time.

Example:

The ‘Lines of code’ is a module-level as well as system level metric from product dimension. The ‘Volatility index’ is a system-level metric. The ‘Schedule deviation’ is a task-level, phase-level as well as project-level metric from process dimension. Along organization dimension, the ‘Productivity’ is an individual-level, group-level and company-level metric.

2.2 Knowledge Base

Knowledge base contains a number of fuzzy linguistic variables (fuzzy sets), and a number of fuzzy inference rules. Semantics of fuzzy linguistic variables (fuzzy sets) are defined by their membership functions based on software metrics. It contains a list of fuzzy inference rules about risk detection across all phases in software life cycle. Fuzzy rules are basically of the IF-THEN structure. Fuzzy inference rules are represented in antecedent-consequence structure. Antecedents represent symptoms of software artifacts, processes or organizations in terms of risks based on fuzzy sets and software metrics. Since our system is based on the ‘Dimensional Analytical Model’ rules use individual metrics and their combinations based on their relationships from different dimensions.

Product Metric based Rule

IF Volatility index of subsystem is HIGH
AND Requirements quality is LOW
THEN Schedule Risk is VERY HIGH

Process Metric based Rule

IF Effort deviation is HIGH
AND Customer involvement is HIGH
THEN Risk of schedule overrun is VERY HIGH

Organization Metric based Rule

IF Number of Communication paths within a group are HIGH
AND Group Productivity is LOW
THEN Schedule Risk is VERY HIGH

Relation between product and process dimensions

IF Customer involvement is HIGH
AND Volatility index is HIGH
THEN Schedule risk is VERY HIGH

Individual Risk Factor Assessment

A list of rules is used to detect individual risk factors and to issue warnings about them ranging from highest to lowest. The intensity of the symptoms and associated risks are expressed in terms of fuzzy variables (fuzzy sets). In order to identify an overall risk associated with an on-going process or system in-built, risk factors associated with on-going basic units of a software process or system should be assessed. A number of fuzzy inference rules are developed to assess individual risk factors associated with basic units of process, product, or organization.

2.3 Intelligent Risk Assessment

Intelligent risk assessment is the result of fuzzy inference engine, which is the central processing part of the system. At a particular state of software development, all the metrics have certain values along all the three dimensions. These metrics form a set of input for the inference engine. Inference engine apply the knowledge base on this set of inputs to produce Quality risk, Schedule overrun and Cost overrun as output set. The input and output sets are stored over the time period for analysis purpose.

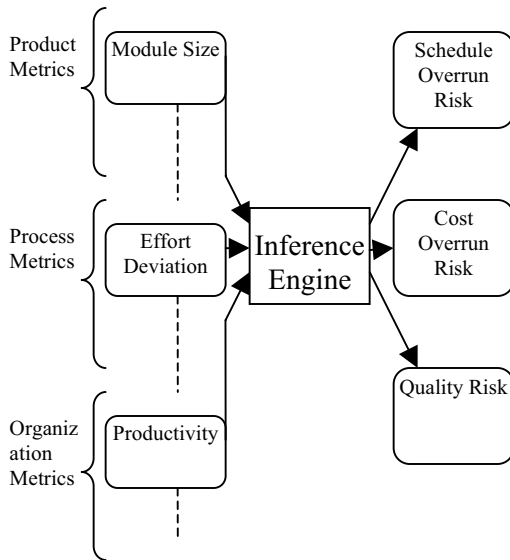


Fig. 3 Fuzzy inference engine

Various steps involved in fuzzy inference are as described below

a) Fuzzify Inputs

The first step is to take the inputs (metrics) and determine the degree to which they belong to each of the appropriate fuzzy sets via membership functions. Fig. 4 shows metrics 'Volatility Index' having value 0.5 is mapped on trapezoidal membership function for High region.

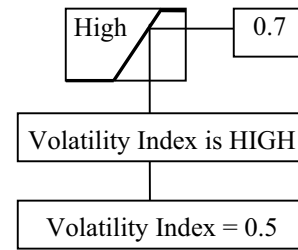


Fig. 4 Input fuzzification

b) Apply Fuzzy Rules

The fuzzy rules may have more than 2 antecedents. Each antecedent is fuzzified. OR operator is applied to the fuzzified values of antecedents across every rule. This results in maximum of the values of antecedents as output. Fig. 5 shows the application of fuzzy rule with two antecedents. The antecedent part of the rules is: 'if Volatility Index is High and Cyclomatic Complexity is High'

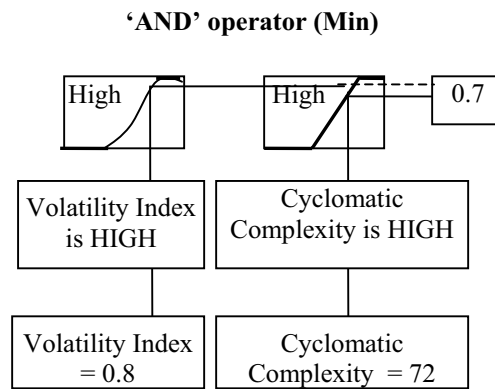


Fig. 5 Application of fuzzy rule

c) Implication Method

Before applying the implication method, we must take care of the rule's weight. Every rule is given a weight ranging from 0 to 1. Once proper weights are assigned to each rule, the implication method is applied. A consequent is a fuzzy set represented by a membership function. The resultant is an area of fuzzy set with degree of membership chopped. Fig. 6 shows the implication method. The rule in Fig. 5 has one consequent 'Cost risk'. The result of step (b) is projected on the consequent to obtain the resultant area of membership.

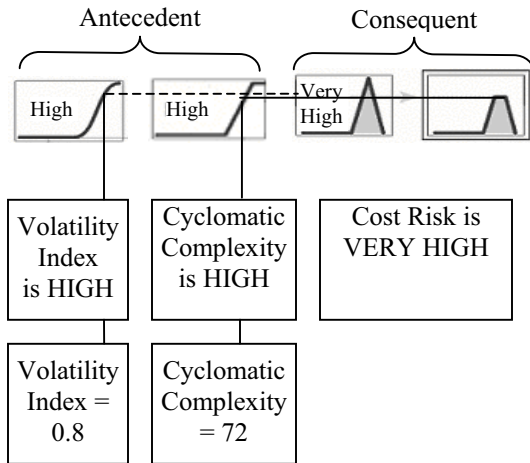


Fig. 6 Implication method

d) Aggregation

Many rules may have same consequent. Each rule produce the region of such consequent separately. Aggregation is the process by which the fuzzy sets that represent such consequents are combined into a single fuzzy set. The input of the aggregation process is the list of truncated output functions returned by the implication process for each rule. The output of the aggregation process is one fuzzy set for each output variable. There exist many ways to combine multiple decision criteria in fuzzy decision science. Many aggregation operators can be used to fuse multiple risk factors in fuzzy risk assessment and fuzzy logic [ZIMMERMANN91]. They can be combined with t-norms, t-conorms, and compromise operators. The t-norms and t-conorms operators have been discussed extensively in fuzzy logic [ZIMMERMANN91]. The t-norms operators are a class of fuzzy conjunction operators. Examples of t-norms operators include MIN and algebraic product. The t-conorms operators are a class of fuzzy disjunction operators. Examples of t-norms operators include MIN and algebraic sum. Averaging and compensatory operators are often used to combine multiple risk factors in fuzzy risk assessment if they are conflicting with each other. The resulting trade-offs of an average operator lie between the most optimistic lower bound and the most pessimistic upper bound. For a compensatory operator, a decrease in one operand can be compensated by an increase in another operand. Thus the "min", a t-norm operator, and the "max", a t-conorm operator, is not compensatory. Actually, many averaging operators are not compensatory and vice versa. An operator is said to be a compromise operator if and only if it is both averaging and compensate operator. An example of aggregation of quality risk using t-norm operator based on system structure is shown in Fig. 7. Rules considered are,

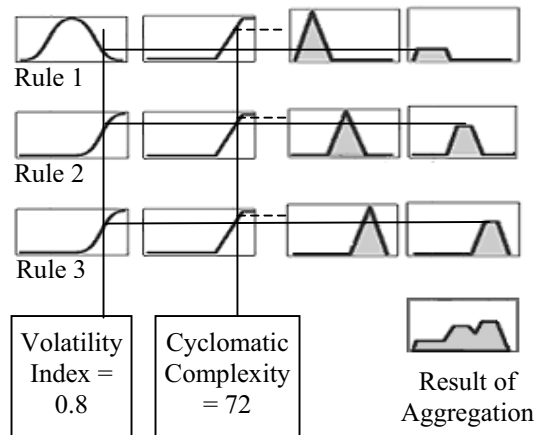


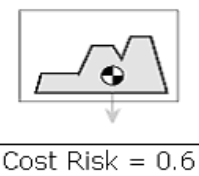
Fig. 7 Aggregation

Rules considered in the figure are,

- Rule 1: If Volatility Index is MEDIUM and Cyclomatic Complexity is HIGH then Cost Risk is MEDIUM
- Rule 2: If Volatility Index is HIGH and Cyclomatic Complexity is HIGH then Cost Risk is HIGH
- Rule 3: If Volatility Index is HIGH and Cyclomatic Complexity is HIGH then Cost Risk is VERY HIGH

e) Defuzzification

The input for the defuzzification process is a fuzzy set (the aggregate output fuzzy set) and the output is a single number. As much as fuzziness helps the rule evaluation during the intermediate steps, the final desired output for each variable is generally a single number. However, the aggregate of a fuzzy set encompasses a range of output values, and so must be defuzzified in order to resolve a single output value from the set. Perhaps the most popular defuzzification method is the centroid calculation, which returns the center of area under the curve.



Result of Defuzzification

Fig. 8 Defuzzification

2.4 Visual Warning Issuing

From the fuzzy inference engine, Quality, Schedule and Cost risks are predicted. The risk regions corresponding to their causes can be represented visually. Fig 9-a shows an example of visual warnings produced by the system.

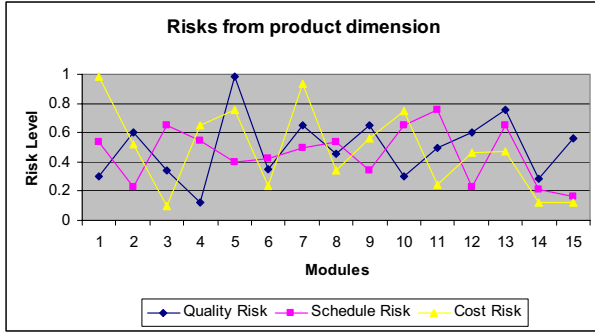


Fig. 9-a Risks predicted from product dimension

The graph in Fig. 9-a shows the quality, schedule and cost risks for all modules, predicted as result of the fuzzy inference system. The risks predicted are mapped on product dimension. From this graph manager can get overall picture for all the module risks. The trade-off between three types of risks becomes easier with such visual warning. For example, for module 15, Schedule and Cost risks are in safe region. So if Quality is not of main concern, then manager can decide to proceed in the development cycle. Else, if quality is of prime concern, then it is advisable to stop and reconsider the factors causing high quality risks. These factors can be analyzed from graph shown in fig. 9-b

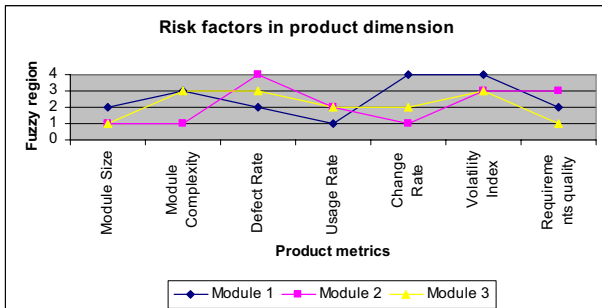


Fig. 9-b Risk factor analysis in product dimension

This graph helps in finding the cause factors for high risk modules. The y-axis shows the fuzzy regions for the metrics values (1-Low, 2-Medium, 3-High, 4-Very High). Similar type of graph can be generated for process and organization dimensions.

Tracing down to cause of risk

As, mentioned earlier, only identification of risk is not sufficient. The risks should be traced down to find out their root causes. A graph in Fig. 10-a shows the quality risk predicted over time period of 10 months for overall

system. Vertical axis shows the risk level (1-Low, 2-Medium, 3-High, 4-Very High)

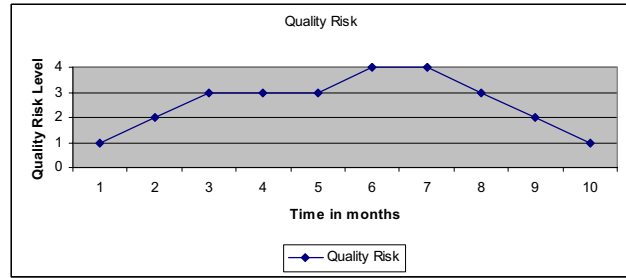


Fig. 10-a Quality risk of entire system over 10 months period

This risk needs to be mapped to all the three dimensions to search for the cause of risk for module/phase/group. Following graph (Fig. 10-b) shows the quality risk for all the modules on product dimension. This example includes four modules. From this graph, high risk modules are easily identified.

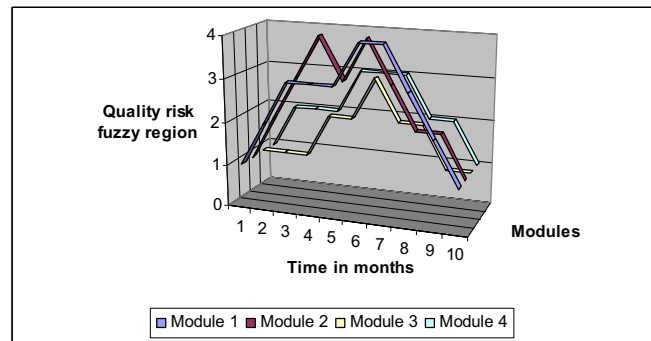


Fig. 10-b Module level tracing

Further in order to find factors for the risk, each module/phase/group is traced down for each factor considered in fuzzy inference. A graph in Fig 10-c shows metrics of module 1 and their risk levels over the time period of 10 months. This example includes the following module metrics: size, volatility index and cyclomatic complexity.

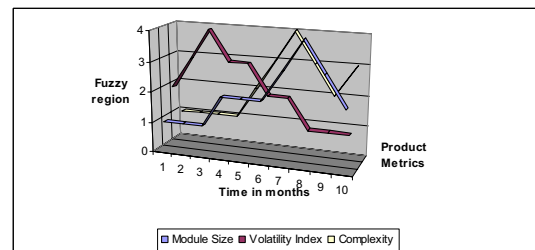


Fig 10-c Metrics level tracing

3. Conclusion

This paper discusses a customizable early warning system for software development. The system is able to detect the risks in very early phase of development using fuzzy logic. It differs from other traditional risk assessment methods, as it utilizes the quantifiable aspects of software development i.e. metrics from three different dimensions: product, process and organization. Direct utilization of metrics for risk assessment is very difficult task due to their imprecise nature. Intelligent risk detection rules utilize not only individual metrics in each dimension, but also their combinations from these three dimensions. This improves the accuracy risk prediction by fuzzy inference engine. The quality, schedule and cost risks assessed can be traced down to the module, development task or group, which is responsible for the expected failures. The individual factors of identified risks can be analyzed through visual warnings. Fuzzy logic and neural networks can be used in combination to further enhance fuzzy inference engine in our future research.

4. Acknowledgements

This system could have not been developed without the support and assistance of team members. We appreciate the assistance of our Software Engineering project team. The team comprised of Sanjeev, Toby, Austin, Steve, Kolcu, Lee Wang, Ghatti, Sriram, Pongupati, Praveen, Gaurav and Kaustubh. They contributed in development of metrics and visual warning part of the system. It's their solidarity; constant input and encouragement that helped us tackle most of the basic issues.

References

- [BOE81] Barry Boehm, Software Engineering Economics, Prentice Hall, 1981.
- [BOE89] Barry Boehm, Risk Management, IEEE Computer Society Press, 1989.
- [BOE00] Barry Boehm, et al., Software Cost Estimation in COCOMO II, Prentice Hall, 2000.
- [HYAT96] Linda H. Rosenberg, Lawrence E. Hyatt, Software Metrics Program for Risk Assessment, software Assurance Technology Center publications, http://satc.gsfc.nasa.gov/support/IAC_OCT96/iaf.html
- [NASA95] NASA Software IV & V Facility, The WWW Integrated Software Metrics Environment, 1995 <http://research.ivv.nasa.gov/projects/WISE/wise.html>
- [NORTEL96] Hudepohl, J.P. Aud, S.J. Khoshgoftaar, T.M. Allen, E.B. Mayrand, J., Integrating Metrics and Models for Software Risk Assessment, 1996. Proc. Int'l Symp. On Softw. Reliability Engg., 1996, 93-98
- [NORTEL98] Khoshgoftaar, T.M. Allen, E.B. Jones, W.D. Hudepohl, J.P., Return on Investment of Software Quality Predictions, Proc. IEEE Workshop on Application-Specific Softw. Engg. Tech., 1998, 145-150.
- [LUQI2000] Nogueira J. C., Luqi, Bhattacharya S., A risk assessment model for software prototyping projects, Proc. Int'l Workshop on Rapid System Prototyping, 2000, 28-33
- [MLYU95] M. Lyu, Software Reliability Engineering, IEEE Computer Society Press. 1995.
- [NSCH75] N. Schneidewind, 'Analysis of Error Processes in Computer Software', Proceedings of the International Conference on Reliable Software, IEEE Computer Society, 21-23 April 1975, pp. 337-346
- [MUSA98] J. Musa, Software Reliability Engineering: More Reliable Software, Faster Development and Testing, McGraw-hill, 1998.
- [GRAY97] S.G. MacDonell, A.R. Gray, 'A Comparison of Modeling Techniques for Software Development Effort Prediction', International Conference on Neural Information Processing and Intelligent Information Systems, Proceedings, pp. 869-872, 1997
- [CVAS94] Vasudevan, C., 'An experience-based approach to software project management', Tools with Artificial Intelligence, Proceedings, Sixth International Conference, pp. 624-630, 1994
- [TMK00] Yuan X., Khoshgoftaar T.M., Allen E.B., Ganesan K., 'An application of fuzzy clustering to software quality prediction', Application-Specific Systems and Software Engineering Technology, Proceedings. 3rd IEEE Symposium, pp. 85-90, 2000
- [TMK02] Zhiwei Xu, Khoshgoftaar T.M., Allen, E.B., 'Early operational risk assessment of software using fuzzy expert systems', World Automation Congress, Proceedings of the 5th Biannual, Vol. 13, pp. 435-442, 2002
- [BBB95] Baisch E., Bleile T., Belschner R., 'A neural fuzzy system to evaluate software development productivity', Intelligent Systems for the 21st Century, IEEE International Conference, vol. 5, pp. 4603-4608, 1995
- [FLIU99] Liu Frank, Ravi Viswanathan, "A WWW Based Software Metrics Environment for Software Process Management and Software Product Quality Improvement", Computer Software and Applications Conference, 1999. COMPSAC '99. Proceedings. The Twenty-Third Annual International, on page(s): 301-304, 27-29 Oct.1999, Phoenix, AZ, USA, ISBN: 0-7695-0368-3
- [Zimmermann91] H. J. Zimmermann, Fuzzy set theory and its applications, Kluwer Academic, 1991.