Early Warning System in the Improvement of Software Quality

Riddhi Mehta[1] Bhadania Meera[2] Faldu Asha[3]
[1,3]Department of Computer Science
[1, 3]Nirma University, Ahmedabad (India)

Abstract—Basic reason behind failure in software development projects is delay in correction of problems even if they are detected within reasonable time. This raises need of a concrete system which can warn against such potential risks. This paper highlights such early warning system based on fuzzy logic using integrated set of software matric. Such system helps accessing risks associated with being schedule. This handles incomplete, inaccurate and non-précised information, and resolves conflicts in an uncertain environment. 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.

I. Introduction

According to recent analysis, almost $275 billion are spent on software projects in a year and out of those, only 26% software projects are completed success. And 7% projects are never completed. One of the basic reasons behind this failure during 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 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 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.

II. An Approach

I am presenting an intelligent early warning system which uses 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 the process of software development can be measured quantitatively using software metrics. All the metrics represent measurements from many different perspectives.

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.

III. 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

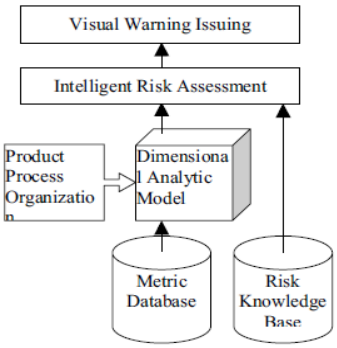

Fig. 1: An Early warning system for software

A. *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.

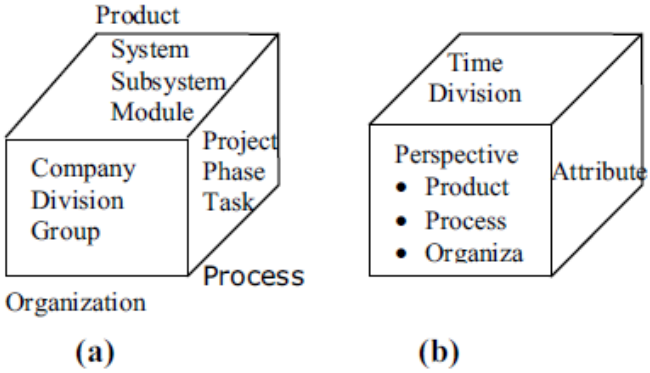

Fig .2: Dimensional analytic model

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.

B. *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.

1) 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 ongoing 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.

C. *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.

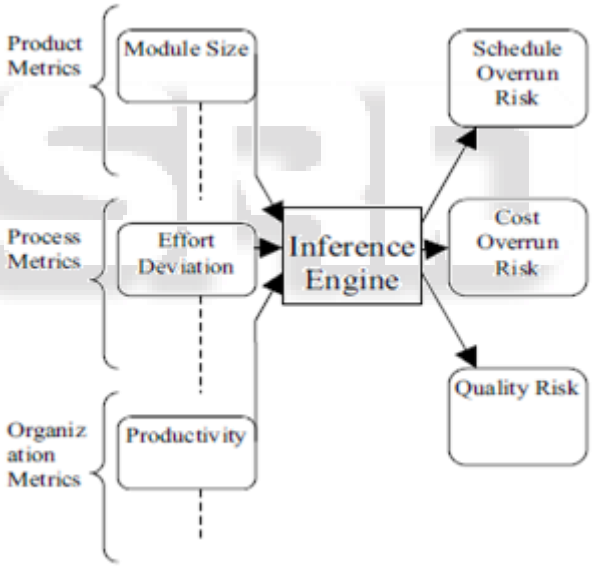

Fig. 3: Fuzzy inference engine

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.

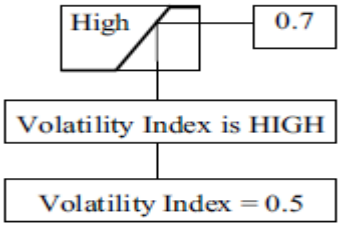

Fig. 4: Input Fuzzification

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'

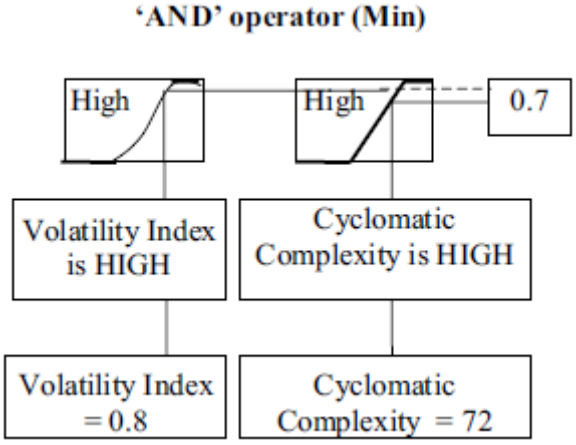

Fig. 5: Application of fuzzy rule

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.

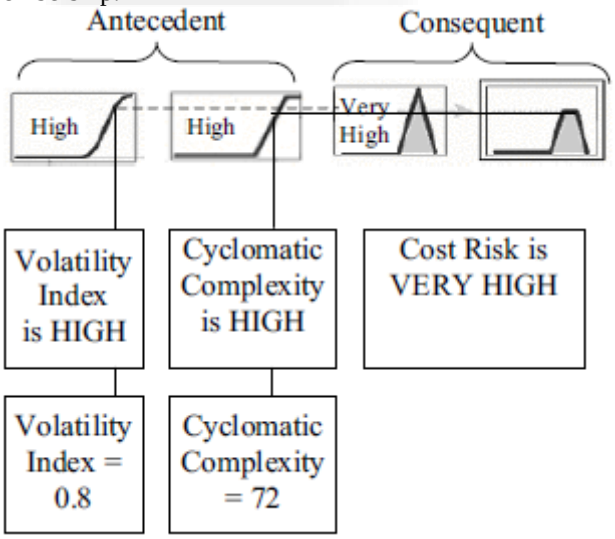

Fig. 6: Implication method

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. 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. The t-norms operators are a class of fuzzy conjunction operators. Examples of t-norms operators include MIN and algebraic product. The tconorms 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,

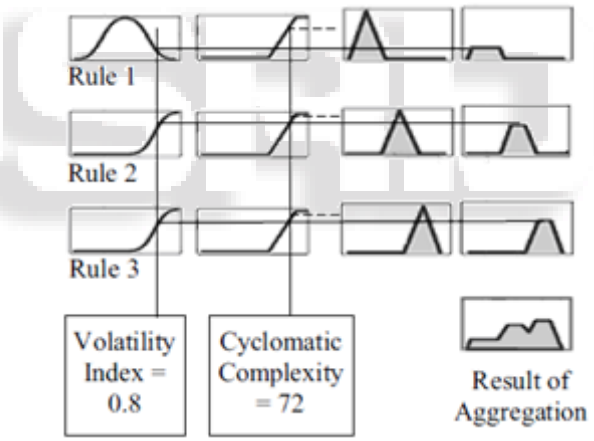

Fig. 7: Aggregation

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.

IV. 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.

REFERENCES

[1] Barry Boehm, Software Engineering Economics, Prentice Hall, 1981.
[2] Barry Boehm, Risk Management, IEEE Computer Society Press, 1989.
[3] Barry Boehm, et al., Software Cost Estimation in COCOMO II, Prentice Hall, 2000.
[4] Linda H. Rosenberg, Lawrence E. Hyatt, Software Metrics Program for Risk Assessment, software Assurance Technolo gy Center publications, http://satc.gsfc.nasa.gov/support/IAC_OCT96/iaf.html
[5] NASA Software IV & V Facility, The WWW Integrated Software Metrics Environment, 1995 http://research. ivv.nasa.gov/projects/WISE/wise.html
[6] 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
[7] 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.
[8] 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
[9] M. Lyu, Software Reliability Engineering, IEEE Computer Society Press. 1995.

