Comparison of conventional approaches and Soft-Computing approaches for Software Quality Prediction

Ekkehard Baisch
Alcatel Telecom, Quality Strategies,
Lorenzstrasse 10, D-70435 Stuttgart, Germany.
Fax: (+49)-711-821-43200,
e-mail: E.Baisch@stgl.sel.alcatel.de

Thomas Liedtke
Alcatel Telecom, Project Manager SSD,
Lorenzstrasse 10, D-70435 Stuttgart, Germany.
Fax: (+49)-711-821-42283,
e-mail: TLiedtke@stgl.sel.alcatel.de

Keywords: Software Development, Quality Prediction Models, Fuzzy Expert Systems, Genetic Algorithms, Multilinear Discriminant Analysis.

ABSTRACT :
Managing software development and maintenance projects requires early knowledge about quality and effort needed for achieving a necessary quality level. Quality prediction models can identify outlying software components that might cause potential quality problems.
Quality prediction is based on experience with similar predecessor projects constructing a relationship between the output - usually the number of errors - and some kind of input - here we use complexity metrics - to the quality of a software development project.
Two approaches are presented to build quality prediction models : Multilinear discriminant analysis as one example for conventional approaches and Fuzzy Expert-Systems generated by Genetic Algorithms.
Using the capability of Genetic Algorithms, the fuzzy rules can be automatically generated from example data to reduce the cost and improve the accuracy. The generated quality model - with respect to changes - provides both quality of fit (according to past data) and predictive accuracy (according to ongoing projects).
The comparison of the appoaches gives an answer on the effectiveness and the efficiency of a Soft-Computing approach.

1. Introduction

Although striving to high quality standards, only a few organisations apply true quality control. Quality control consists of comparing observed quality with expected quality, hence minimizing the effort expended on correcting the sources of defect. The approach to improve software development productivity is therefore based on the improvement of the quality, especially the correctness and the maintainability. In order to achieve an indication of software quality, the software must be subjected to measurement. This is accomplished through the use of metrics. The evaluation is done, based on statistical techniques that relate specific quantified product requirements to some attributes of quality. This presentation introduces fuzzy expert system techniques as a basis for constructing quality based productivity prediction models that can identify outlying software components that might cause potential quality problems, thus requesting additional effort. This effort extremely shows up in maintenance projects done years later by new developers.
The presentation is organized as follows. The second section presents a brief overview of the background and the problems associated with metric-based decision models (e.g. vagueness in reasoning). The following section discusses the fuzzy sets, their application in fuzzy classification and the construction of a fuzzy classification system for software quality control. The next section describes the extraction of a Fuzzy Expert System out of project data using Genetic Algorithms. Finally, experimental results are provided to demonstrate the effectiveness of the approach in the area of error- and change-prediction.

2. Fuzzy Data Analysis

The nature of software data presents a number of problems [1]. The data is often heaviliy positively skewed with a large number of outliers. Daily life in software quality control shows that it is often inappropriate to expect hard boundaries between

0-7803-4053-1/97/$10.00 © 1997 IEEE

what is a good component and what is not. It is hence somehow difficult to explain to the design staff that a specific module with 15 decisions should be redesigned while another one with only 14 decisions just passed the check. It is practically not realistic to deal with with decisions or rule sets based on crisp thresholds. However, this is basically the approach of common decision support systems in software quality control.

Fuzzy logic provides a natural conceptual framework for representation of knowledge and inference processes based on knowledge that is imprecise, incomplete or inconsistent. In a fuzzy expert system both the premises and the associated conclusions are fuzzy sets that are characterized by their possibility distribution. Fuzzy rules are used to express the heuristic knowledge about metrics and quality factors like maintainability. Rule predicates are based on the fuzzified metric values. The inference process of all active rules generates a fuzzy set for the defined quality factor.

3. Construction of a Fuzzy Expert System

Any software development project starts with the definition of project specific quality goals that depend on customer requirements, technical requirements and the company's business goals. These desired properties or quality factors, e.g. maintainability, reliability, reusability,... are the roots of a quality model [3]. In order to make them quantifiable, they are further refined into metrics according to a goal-oriented approach (e.g. *Goal Question Metric Paradigm*)

The next step is the definition of a rule base for the classification. Every rule consists of a combination of metrics as premises and a conclusion for a quality factor. The definition of rules representing heuristic knowledge is a difficult task. It is not clear whether one should use combinations of metrics as premises or only one metric per rule. It should be taken into account that according to the selected goals and their refinement to quality criteria and metrics, some metrics are more important than others and metrics belong together in certain cases that describe distinct quality criteria.

Addionally a lot of other parameters remain in a fuzzy rule base which need to be determined. It is necessary to define a scale for metrics and to place the fuzzy terms over the universe of discourse and to determine which value corresponds best to a certain fuzzy set. The amount of open parameters is illustrated in figure 1.

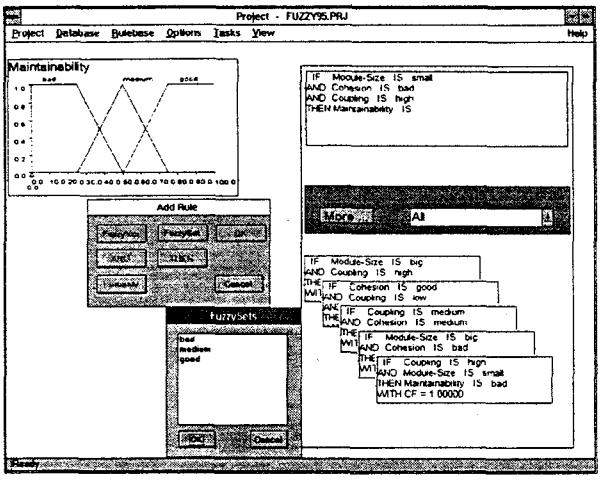

Fig.1: Construction of a Fuzzy Expert System

We propose to use Genetic Algorithms to assist human experts doing a data analysis for tailoring a fuzzy-rule-base to a specific software development environment. Genetic Algorithms can extract an evironment-specific fuzzy-rule-base if they are provided with the quality results (errors and changes) and the complexity metrics from finished projects in the same development environment. These data is called training data.

The extraction of fuzzy rules has several advantages compared to common classification approaches :

The extracted rules can be verified by experts and especially the overfitting phenomenon is identifiable now. Overfitting usually occurs after long training when the extraction model memorizes the patterns and peculiarities of the training data but looses its generality to data applied later.

4. Principle and Implemantation of a Genetic Algorithm

The advantage of Genetic Algorithms (GA), emulating biological evolutionary theory on a computer, is their ability to cope with non-linear problems where gradient descent methods reach their limit.

In computing terms a Genetic Algorithm maps a problem onto a set of (typically binary) strings, each string representing a potential solution. The GA then manipulates the most promising strings by

applying evolution inspired operators like cross-over, mutation and selection of the best.
The principle is shown in figure2.

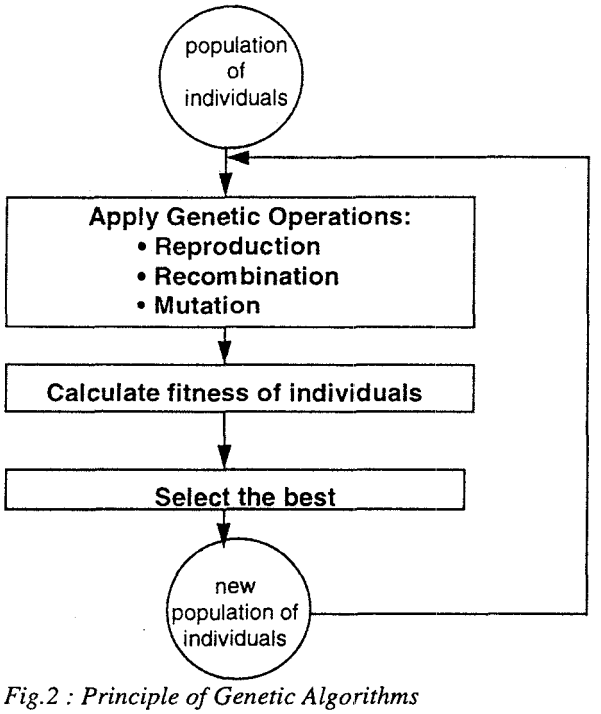

Fig.2 : Principle of Genetic Algorithms

In our case every individual represents a complete fuzzy expert-system. The main task is the encoding of a fuzzy rule-base into binary strings.
The computer-based tool realizing a Genetic Algorithm is called a Genetic Optimizer. the main features are shown in figure 3.

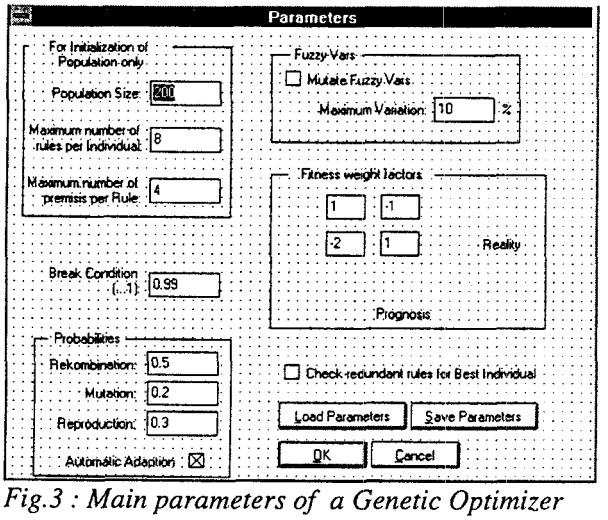

Fig.3 : Main parameters of a Genetic Optimizer

It is necessary to define the probabilities of the different Genetic operations. During the learning process it is necessary to adapt these probabilites in order to reduce the learning time. The positions of the fuzzy sets can be determined statistically or determined via mutation during the learning process.
One important feature of a Genetic Optimizer is the possibility to define a project specific fitness depending on the project goals. This is done using a contingency-table like weighting according to false and correct classification of the training data. Errors of type I are errors where critical modules are classified as non-critical. Within safety critical projects these errors need the highest negative weighting. Errors of type II are errors where uncritical elements are classified as critical. They cause only additional effort for inspections and reviews.

5. Application on Industrial Quality Data

We applied our prediction techniques to the software of the Alcatel 1000 S12 telecommunication system. Training data was taken from several real-time telecommunication projects that had been developed according to a similar design approach. We investigated a selection of 710 modules. The overall size of these modules is in the area of 2 Mil lines of executable code. The specific project had been in field use for over a year thus showing stability in terms of features and failures. Corrective software changes are given for each module together with several complexity metrics based on the *COSMOS* (*ESPRIT*) project.
The application of the change-prediction in our software-development environment is shown in figure 4.

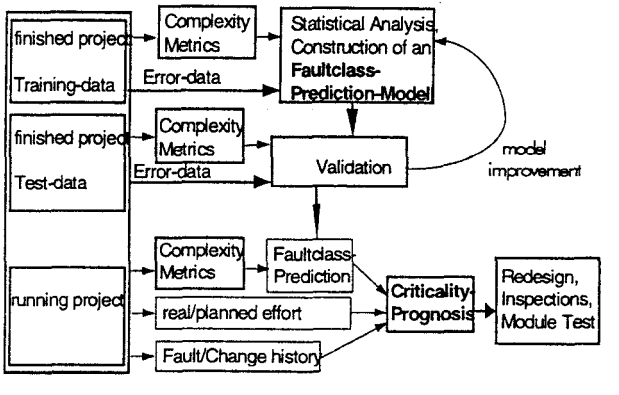

Fig.4 : application of change-prediction

Complexity metrics used in this project include number of (executable) statements, statement complexity, expression complexity, data complexity, depth of nesting (control flow), and data base access (number and complexity of data base accesses). Spearman rank correlations among different complexity metrics were considerably high. For all selected metrics they were above 0.8. Complexity metrics also correlated with number of changes (average over 0.4).

We classified the modules in 2 different classes. Set1-modules had no changes up to 20 changes, set2-modules more than 20 changes.
The data was splitted into training data and test data in order to validate the change-prediction model.

First we used Multilinear Discriminant Analysis(MLDA) to predict the change class of 409 modules in the training data. MLDA is a regression-based approach combining the complexity metrics linearly. It has a black-box behaviour because you do not understand why a certain module is classified as critical (change-prone) or not.
The result is shown in figure5.

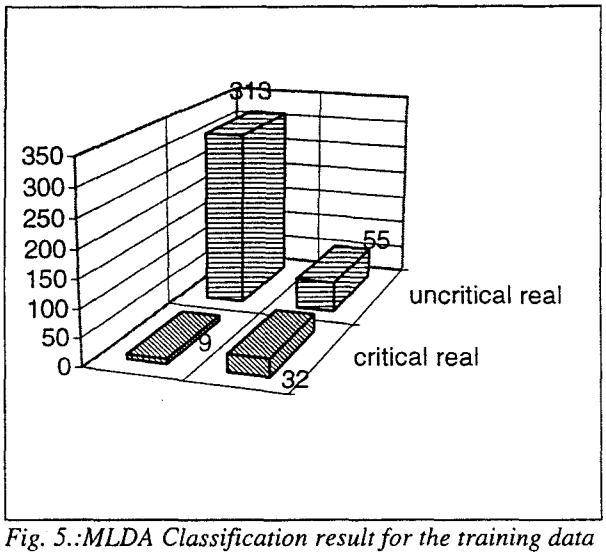

Fig. 5.:MLDA Classification result for the training data

The result of our classification is shown in the form of a contingency table. The rows show the real situation. The columns show the prognosis. The prognosis is correct on the first diagonale. These were 345 modules, 84,3% of all modules. 32 = 78% of the critical modules were identified correctly.

The validation was done using the test data of 301 modules. The result is illustrated in figure 6.
243 modules = 80,7% were classified correctly. 27 =75% of the critical modules were identified correctly.

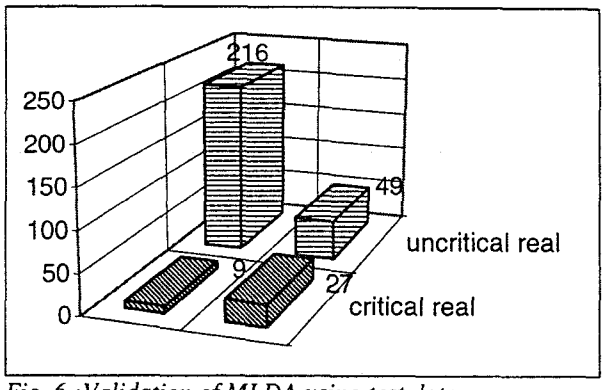

Fig. 6 :Validation of MLDA using test data

Next we applied the Genetic optimizer to extract a fuzzy expert system for change prediction. We weighted the fitness to reduce type I errors.
The following fuzzy-rule-base was extracted with the training data :

```
IF SCMET IS Set3
AND DACMET IS Set2
AND MCMMET IS Set3
THEN chan IS set2

IF DACMET IS Set3
THEN chan IS set2

IF MCCMET IS Set1
THEN chan IS set1

IF NRPMET IS Set3
THEN chan IS set2

IF DACMET IS Set1
AND NRPMET IS Set2
THEN chan IS set2

IF MCCMET IS Set2
AND MCMMET IS Set2
THEN chan IS set1

IF MCCMET IS Set2
AND NRPMET IS Set2
THEN chan IS set1

IF MCMMET IS Set2
AND NRPMET IS Set2
THEN chan IS set1

IF DACMET IS Set2
AND MCMMET IS Set1
THEN chan IS set1
```

Table 1 : Fuzzy-Rule base for change-class-prediction

The classification result for the training data is shown in figure7

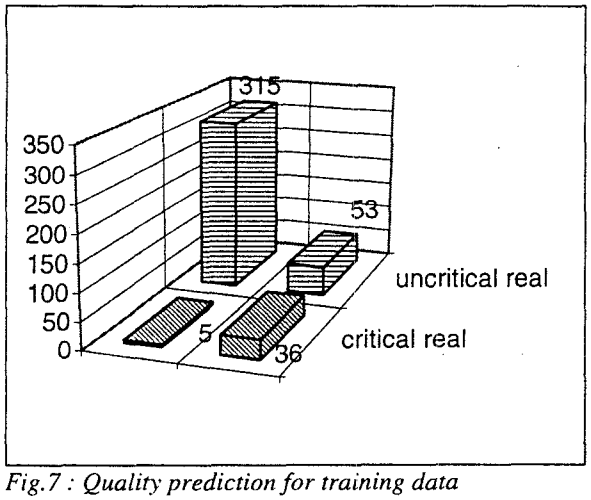

Fig.7 : Quality prediction for training data

351 modules were classified correctly, 85,8%.
36 = 87,8% of the critical modules were identified. Only 5 of the critical modules were not found. This was possible with the special weigthing of the fitness. This is an important improvement to the MLDA approach. The next step was the validation of the prediction model with test data, to see if there is a general improvement. The result is shown in figure8.

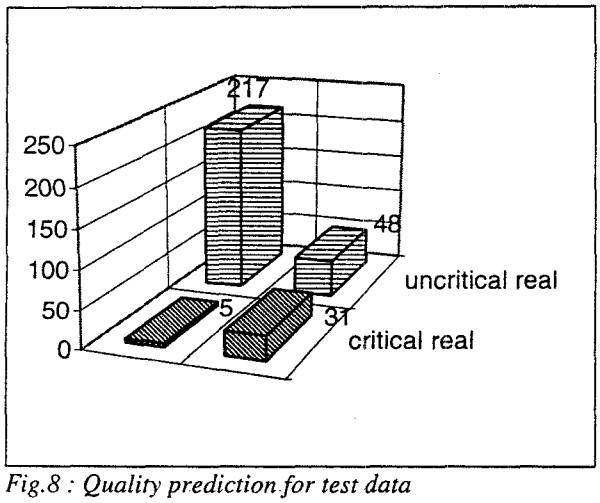

Fig.8 : Quality prediction for test data

248 = 82,4% of all modules were classified correctly. More important, 31 =86% of the critical modules were identified.

6. Conclusions

We demonstrated that it is possible to predict software module changes using complexity metrics as input for the prediction model. Two approaches were investigated. The conventional technique, Multilinear Discriminant Analysis, provides us with satisfactory results in a short time. The advantage is the simplicity of the approach understandable by every engineer with a minimum knowledge in statistics. The disadvantage is the black-box-behaviour optimizing only the overall sum of correctly classified elements.
The fuzzy expert system can be built giving a higher priority to the avoidance of certain types of errors. This is especially interesting for safety critical software applications. The approach requires expert skill with Genetic Algorithms and requires 10 times more calculation time in a PC environment.
Even with the priority to avoid certain error types the overall result is as good as the MDLA approach. Without this priority the result of all the correctly classified modules of the test data would be 10% better than the MDLA result. This shows the advantage of using a fuzzy approach avoiding overfitting due to crisp boundaries.
If a module is classified as critical the reason can be given identifying the rules which were active for this module in the fuzzy rule-base. An immediate feedback to the responsible designer is possible now.
Our goal is an evaluation of software quality early in the life-cycle to improve life-cycle productivity. This is especially done by correcting errors as soon as possible and by guaranteeing a certain maintainability of the resulting software product. If one had the maintenance effort data per module our approach could lead to tremendous improvements in software life-cycle productivity.

References

[1] Fenton, N.: Software Metrics - A Rigorous Approach. Chapman & Hall, London, 1991.
[2] Ebert, C.: Rule-Based Fuzzy Classification for Software Quality Control. *Fuzzy Sets and Systems*, Vol. 63, pp. 349 - 358, 1994.
[3] Kitchenham, B. : Towards a Constructive Quality Model. *Software Engineering Journal*, S. 114-126, July 1987.
[4] Basili, V.R. and H.D. Rombach: The TAME Project: Towards Improvement-Oriented Software Environments. IEEE Trans. Software Eng., Vol. 14, No. 6, pp. 758-773, Jun. 1988.