Building fault prediction models from abstract cognitive complexity metrics - analyzing and interpreting fault related influences

Roland Neumann[1], *Stamatia Bibi*[2]

[1]Hasso-Plattner-Institute for Software Systems Engineering GmbH at the University Potsdam,

[2]Aristotle University of Thessaloniki, Greece

neumann@hpi.uni-potsdam.de[1], sbibi@csd.auth.gr[2]

Abstract:

Finding software defects as early as possible is a critical task that saves values. Fault prediction models identify faults by spotting error prone components, exploring design guidelines and thus directing test effort. These models gain knowledge from past mistakes to prevent future ones. This paper describes, applies, evaluates and compares modelling techniques of fault related code structures based on abstract complexity metrics. These complexity metrics are calculated from base metrics for all complexity aspects assuring statistical independence. Modelling techniques applied are MARS, Classification And Regression Trees, Association Rules and Bayesian Belief Networks. Their ability to interpret fault reasons and predict future possible faults is compared in this paper.

Keywords

Software metrics, fault prediction, complexity, MARS, CART, Bayesian Networks

1 Introducing complexity aspects

Embedded software is substituting specialized hardware in terms of functionality in technical systems. Even in safety critical environment, this process is increasing due to a better flexibility, the rising need for more functionality and cost effectiveness. Nowadays real time software is embedded in mission critical systems, like in avionics with the replacement of mechanical flap transmission due to the size and weight increase. Starting with a hydraulic power system we have now flight control systems giving direction based on sensor information. Many advantages are indicated by this change, although new quality assurance techniques had to be found where the old ones are not sufficient anymore. Quality aspects like failure causes and models have been improved for years, but the new systems brought additional failures with different causes. These failures are based on functionality which is controlled from software algorithms. These source code defects have to be found.

Source code defects in dependable systems induce a high risk of human or property damage in case of a failure. A failure with its source related to the software part of the system is caused by a program error or fault [4]. First a distinction between faults and failures is necessary. Failures in software are deviations of the delivered service from specific conditions [3]. Faults are code properties deviating from intended component behaviour. While faults can emerge from different possible reasons like disturbed concentration, the cognitive complexity is a main one. The cognitive complexity describes the difficulty to understand a source code part with all its relating communications. This study bases on structure for analysing cognitive complexity of object oriented software [5] to define metrics.

In the rest of this paper a possibility to independently measure complexity aspects is presented, followed by a short description of the modelling techniques used. After that, the prospects of the models for inspecting fault influences are presented on a praxis example. Finally, the modelling and prediction quality is compared and the paper is concluded.

2 Measuring complexity

A fault may be induced in the architecture or implementation step by a human. Reasons for this insertion include a lack of understanding of the complex structure leading to an overlooked important interaction effect. These faults can be found investigating the complexity of the architecture or the complexity of code. Analysing complexity needs decomposition into different aspects. To weight these aspects they first have to become measurable.

To quantify structural properties the usage of software metrics is necessary. Software metrics assign numbers to specific code properties according to a counting rule. There are prerequisites for using software metrics for predicting faults coming from measurement theory like scale types [6]. An introduction in software measurement and software metrics with their interpretation can be found in [7] or [8]. Though, there is a problem with interrelation when modeling fault influences from these metrics [13]. These interrelations can be removed using the principal components analysis technique. This leads to abstract metrics describing complexity aspects independently in this data set. The basic metrics have to comply to the ratio scale [6]. The meaning of these complexity aspect metrics is then interpreted using their factor loading values and Eigenvalues [5].

With these independent and abstract complexity metrics, their fault relations can be modelled. These models gain knowledge from empirical data of software systems to prevent errors. This is one important goal in software engineering making a quality assessment of large systems easier. Especially, these large systems improve the empirical model quality with their large data sets.

3 Empirical data basis

For the empirical validation of the various modelling techniques, a basis data set with known number of faults has to be selected. The data set comes from a component of a large railway operation interlocking system in C++. Its properties are measured using a set of 11 metrics. These metrics are then transformed to independent complexity aspects. The number of aspects used is restricted to five since they describe a sufficient amount of data (86%) though being few enough to give good examples. The aspect's meaning can be interpreted according to [5] as:

KE1:	Abstract size measuring count of Methods, foreign attributes, lines and nesting depth
KE2:	Inherited functionality measuring inheritance depth, inherited methods and parents
KE3:	Further inheritance measuring inheritance to and direct children
KE4:	Unused attributes measuring ancestor type attributes and foreign type attributes minus attribute usage count
KE5:	Data Storage measuring ancestor typed attributes and hidden attributes access

4 Modeling fault proneness

For building regression models, the right modelling technique has to be selected. Linear models tend to only fit locally while nonlinear models require preliminary knowledge of interconnections. Neural nets (NN) [17] said to fit various relations are hard to interpret. To overcome these problems, a piecewise linear regression technique (MARS) with interpretable results and performing equal to NN is suggested in [17]. Its usage and prospects are presented and compared with concurrent techniques in the following.

MARS (Multivariate Adaptive Regression Splines) is a data mining technique fitting piecewise linear functions to the input data. The parameters of these linear functions (the knots) are placed with a tree based partitioning algorithm. For maximum robustness, less important knots are removed. When applying the technique to interrelated metrics, multidimensional output functions are generated. With independent input variables, the modelling technique still generates dependent output functions for a better fit. Though this may produce a smaller modelling error, it decreases the prediction quality and has to prevented.

Association Rules (AR), Classification And Regression Trees (CART), and Bayesian Belief Networks (BBN) provide probabilistic models of fault prediction based on historical data. Association rules [16] describe the underlying data relationships with a set of rules that jointly define the target variables. This technique finds frequent attribute combinations. An association rule is a simple probabilistic statement about the co-occurrence of certain data ranges at target values.

CART is a widely used statistical procedure in predictive modelling for producing classification and regression models with a tree-based structure [16]. The CART model consists of an hierarchy of univariate binary decisions. The algorithm used operates by choosing the best variable for splitting data into two groups at the root node. This splitting procedure is then repeated to the data in each of the child nodes recursively, for class labels being as homogeneous as possible.

Bayesian Belief Networks are Directed Acyclic Graphs (DAGs), being causal networks that consist of a set of nodes and a set of directed noncyclical links between them [16]. Each node represents a random variable that can take finite, mutually exclusive values according to a probability distribution. Each link expresses probabilistic cause-effect relations among the linked variables and is depicted by an arc starting from the influencing variable and terminating on the influenced variable. The strength of the dependencies is measured with the help of conditional probabilities.

Due to the discrete nature of these methods complexity aspects have to be separated into categories. One problem is the choice of the interval number of and width. Initially the interval number was selected from the number of records n according to Sturge's rule [19] $k = 1 + 3.3\log(n)$.

The interval width was chosen for equal class number in each interval. While applying the methods the great number of complexity aspect intervals in some cases prevented the extraction of useful models. In these cases neighbour categories were merged.

From these models, tentative conclusions can be drawn regarding the origins of faults by analyzing if the model indicates the dependency of fault number on certain complexity metrics. The models accuracy and robustness is then assessed and compared while sufficient experience from the use of these techniques has been acquired, indicating certain advantages and drawbacks of the methods.

5 Interpreting the models

To interpret the models, first the meaning of the input variables has to be known. These are a prerequisite to draw conclusions from the established models. For our data set, the complexity aspects are introduced previously. The fault prediction models consist of their base functions defining the relations between the complexity aspects and previous faults in this specific software. The MARS model, generated from the modelling data set, consists of the following base functions (Fig. 1):

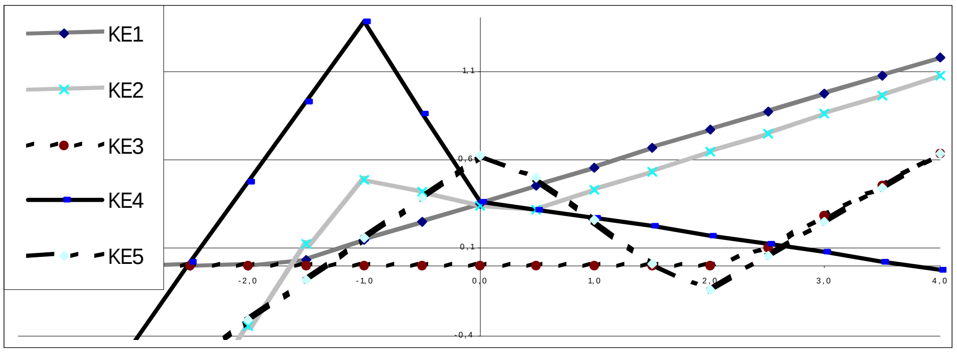

Figure 1: MARS fault influences from complexity aspects

The influence of the complexity aspects to the fault number can be drawn from these equations. For example, the complexity aspect KE1 accounts for a linear increasing fault number greater than a value of -1.7. Since this aspect is related to size, it depicts a fault influence from medium classes increasing with the class size, small to medium sized classes generate no faults. With this model, the complexity aspects of each class can be assessed to spot possible fault influences.

The next modelling techniques do a different, probabilistic approach. Since most classes contain no faults the fault probability of every training data set will be not high though indicating possible problems. Usage of the Association Rules (AR) technique extracts a set of rules from the data set. For this set, some rules with sufficient support could only be generated for concatenated fault classes (1 – 4 faults). The AR technique involves the following steps with its rule set is shown in Table 1:

1. Discretization of the target variable into four possible fault values
2. Creation of the prediction model
3. Transformation of fault category number into a continuous number when needed taking the centre of gravity of the class (the median of the class).

Support	Confidence	Rule	
1.1	100.0	-0.191<KE3= -0.008 & KE4 = -0.408 & KE5 >-0.309	faults _2_3_4
53.9	87.2	KE4 > -0.155	faults_0
5.3	80.0	-0.914<KE3= -0.754 & KE4 = -0.155	faults_1
11.1	70.0	KE4 = -0.155 & KE1> - 0.53 & KE2 > -1.104	faults_1_2_3_4
5.5	63.6	-0.408<KE4= -0.155 & KE3= -0.614 & KE1 > -0.53	faults_1
7.1	52.9	-0.191<KE3= 0.785 & KE4 = -0.155 & KE2>-1.104	faults_1_2_3_4

Table 1: AR rule set

The first column of the rule set represents support values and the second column represents confidence value.

For example the fourth rule can be interpreted as following: If the fourth complexity aspect value falls between the first four classes and the values of the first and the second complexity aspects fall into the last 6 classes then there is 70% probability of the class being faulty with one fault most probable. This rules was generated from 11.1% of the modelling data set (42 classes of the modelling data set follow that rule). This rule set shows the usual drawback of association rules which is its inability to provide estimates for all possible cases. In the current data set, only two classes could not be estimated. In that case the answer is provided by the distribution of the projects given their fault values which points out that the most possible situation is that no faults will be found. Therefore the classes that could not be directly estimated by the model were considered having no faults.

The next applied method was CART with its results shown in Figure 2. The circular nodes represent the splitting nodes and the rectangular nodes represent the leaf nodes that provide the estimation. Each leaf node is accompanied by a probability that is indicative of the validity of the estimation.

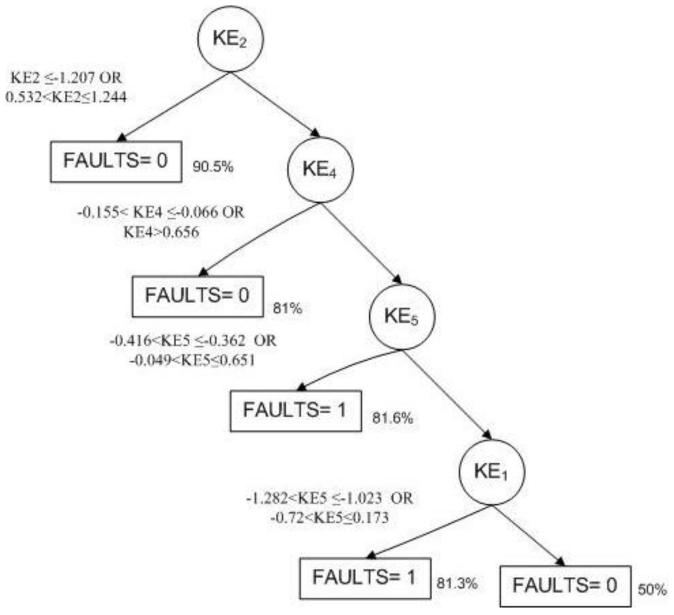
KE2
KE2 ≤-1.207 OR 0.532<KE2≤1.244
FAULTS= 0
90.5%
KE4
-0.155< KE4 ≤-0.066 OR KE4>0.656
FAULTS= 0
81%
KE5
-0.416<KE5 ≤-0.362 OR -0.049<KE5≤0.651
FAULTS= 1
81.6%
KE1
-1.282<KE5 ≤-1.023 OR -0.72<KE5≤0.173
FAULTS= 1
81.3%
FAULTS= 0
50%

Figure 2: CART estimation model

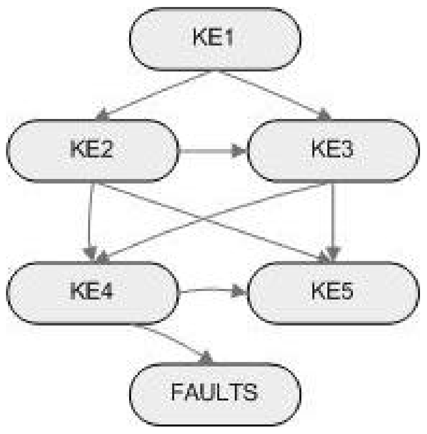

Figure 3: BBN estimation model

For example the node that gives an estimation of 0 faults and is accompanied by the probability 81% can be interpreted as following: If the value KE2 **is not** less than -1.207 **and** not between 0.532 and 1.244 **and** the value of KE4 **is** between -0.416 and -0.362 **or** between -0.049 and 0.651 there is 81% possibility that no faults will be found in a class. The model provides fault estimation for all possible classes even for those whose attributes have not been met in the modeling data set. CART results are limited since classes having 2 to 4 faults do not appear at all at the estimation tree.

This means that in the predictive data set all classes with 2, 3 or 4 faults are considered to have either 0 or 1 fault. All the complexity aspects are included in the model apart from KE3. CART method tries to split the classes into sets as homogeneous as possible according to their fault. Not enough data points were provided to create sets of classes with 2 to 4 errors in preventing rules for these fault numbers.

At construction of BBN, initially ten classes were considered for the values of each complexity aspect. With application of the k2 tool [19] no dependency between the complexity aspects and the fault number was found rendering the model inappropriate for fault estimation. The next step was to merge neighbor classes of the complexity aspects. In that case the BBN structure was better as it involved the fault number as well (Fig. 3). There the fault number depends only in the value of the fourth complexity aspect. Table 2 depicts the node probability table of fault node.

Faults/ KE4	= -0.408	= -0.155	= -0.008	= 0.23	> 0.23
0	0.385	0.519	0.902	0.8	0.797
1	0.385	0.364	0.062	0.141	0.131
2	0.141	0.091	0.012	0.035	0.024
3	0.076	0.013	0.012	0.012	0.024
4	0.013	0.013	0.012	0.012	0.024

Table 2: Node Probability Table for fault node

As an interpretation example, when -0.155< KE4 = -0.08, there is 0.902 probability that no faults will be found in the predicted class, 0.062 probability that one fault will be found in the class, 0.012 that two faults will be found in the class and so on. For KE4 being in the [-0.155; -0.08] range the BBN estimates no faults. In the second column where KE4 = -0.408 the possibility of finding no faults in a class is the same with the possibility of finding one fault in the class. Since we aim at the fault proneness, we assume that there is 0.616 (0.385+0.141+0.076+0.013) possibility that the class will present 1 to 4 faults. This outweighs the possibility of finding no faults. As the no fault probability is quite high due to many fault free classes, only KE4 values less than -0.408 provide strong probabilities for faultiness.

In general all methods indicate the fourth complexity aspect as the main fault related factor. The unused attributes aspect seems important for this data set and its values should be kept greater than -0.155 to avoid faults. Another complexity aspect whose value affects the fault existence is KE3 that measures the further inheritance and direct children of a class.

6 Evaluating the models

A comparison of model quality is possible through different methods. First the quality assessment has to be divided into modeling and prognosis part. Modeling describes how good the model fits the values of the training data set. The prognosis quality residuum RES, representing the difference between the real and the prognosted fault value, can only be assessed with an evaluation data set with known faults. Usually this is conducted with separation of the raw data into an training and an evaluation part.

Evaluation methods generate a single value for the modeling and the prognosis quality. These values can be compared and interpreted. A standard though not uncritizised [18] method is MMRE (Mean Magnitude of Relative Error). In fault prognosis, the calculation of relative errors leads to a division by zero problem for error-free classes. Besides, the aim in fault prognosis is to prevent faults in classes leading to an zero aim value. This decreases the importance of a relative view on differences. The modeling difference at a class with many faults is as important as at a fault free class. The basis of all evaluation methods is the residuum RES, the unsigned difference between real and modeled / prognosted value. This can also be squared to amplify high single values. The sum of these unsigned and squared differences can also be weighted with the number of data points to generate an average difference-per-point value.

An important method assessing discrete data is the hit rate describing the number of correct model values to total number of values. For the machine learning methods, hitrate1 is also presented which shows the ability of the methods to predict correctly the possible faults that may appear. Hitrate1 is used for estimation assessment of a fault interval (e.g. 2-4 faults). At calculation of hitrate1 the model is considered successful if the predicted class presents either 2, 3 or 4 faults. The modeling results for the training data set are shown in Table 3. The modeling accuracy of AR and BBN could not provided due to the lack of an automated method for providing such data. Future work involves the resolution of this problem.

Methods	Hitrate	|RES|	(RES)^2	Rel. |RES|	Rel.(RES)^2
MARS	77.4 %	107,7	72,0	0,347	0,232
CART	82.9 %	76,0	138,0	0,245	0,445

Table 3: Modelling evaluation criteria

Though seemingly fitting better, the CART technique only generates discrete values for faults. This leads to a bigger difference for some data points generating higher squared deviances. MARS values tend to be not as good as CART for most data row but having not as extreme outliers.

For our prognosis study, we separated 30 classes from the raw data selected to represent the whole. These prognosis results are shown in Table 4, their values are compared in Figure 4.

Methods	Hitrate	Hitrate1	|RES|	(RES)^2	Rel. |RES|	Rel.(RES)^2
MARS	46,7 %		23,3	29,2	0,775	0,972
CART	40,0 %	40%	29,0	53,0	0,967	1,767
AR	70,0 %	83.3%	13,0	21,0	0,433	0,700
BBN	40,0 %	66.7%	25,0	41,0	0,833	1,367

Table 4: Prognosis evaluation criteria

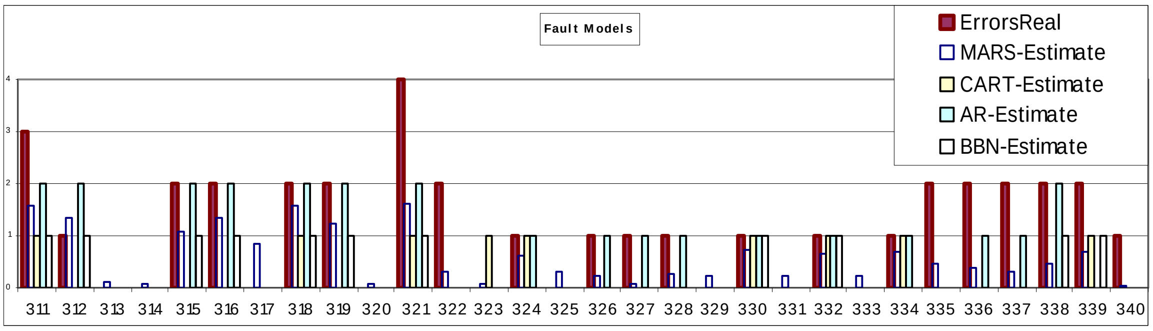

Figure 4: Prognosis comparison

The prognosis assessment shows the same effects as the modeling assessment. The values for the squared residuals are higher for CART, AR and BBN though seemingly fitting better due to discretisation. No model could fit extreme outliers since the few data rows. The MARS technique was also applied for 100 separated data rows showing comparable results to the other techniques.

AR performed well for prognosis data as they are able to correctly estimate the fault number of 70% of the classes. Only one rule is included which actually estimates the majority of fault free classes correctly. Since most (67,7%) classes in the data set have no faults, a class is considered to have no faults if it cannot be estimated by the rule set. The other five rules provide either a particular fault number or a possible range of faults that may appear in a class. In the last case, when a particular fault number is needed the median of the fault category is used as explained previously. When AR method is evaluated according to its ability to predict a range of possible fault values its performance even greater estimating 83.3% of fault classes correctly .

CART has a poor estimation accuracy due to their inability to estimate classes with 2, 3 or 4 faults or even a range of fault values involving these fault numbers. Although it is able to estimate the majority of classes with no faults correctly and have a high estimation accuracy on the modelling data.

BBN has a low estimation accuracy when a particular fault number has to be predicted, though their estimation accuracy is improved for prediction of possible faults. Its estimation accuracy then can be explained with the only usage of the fourth complexity aspect.

7 Conclusions

This paper presents current concepts for building fault prediction models and evaluates their suitability. As these models use independent complexity metrics as input variables, the model functions can be interpreted. With these models, fault reasons can be traced back to complexity structures in classes. This gains reusable knowledge from past errors.

In conclusion, MARS, AR, CART, BBN are all data mining methods that offer a convenient way to solve problems that are not explained purely logically but rather probabilistically. Software fault estimation is one of these problems: we are not sure of the factors that affect the existence of faults directly and we expect a support from statistical methods to point out the underlying relationships that appear in fault data. The presented techniques have a principal difference, MARS predicts a continuous fault value while the other methods give a discrete one with a certain probability.

Though there are more sophisticated evaluation techniques (e.g. 'leave one out', correctness / completeness graph [4], [11]) for further work, our study shows some good first results. All estimation methods suffer from possible overfitting the model data decreasing the prediction accuracy. This has to be prevented as with the tree pruning in MARS [2]. AR, CART and BBN seem promising as they are able to predict fault values of most classes correctly, though data set dependent only the fault proneness. One of the advantages of these methods is their ability to include uncertainty in the estimation models. Uncertainty is depicted by the existence of probabilities and also by the estimation of a range of possible fault values. The fault proneness prediction with a high confidence value is better (1, 2, 3 or 4) than the fault number prediction with a high possibility of wrong estimation.

A combination of MARS method with AR would take advantage of the rules with high support and confidence values and would take advantage of the MARS equation when the rules will not be able to provide an estimate with high confidence. This could be the target of future work. In our survey, AR might perform best for the assessed data. Though not being able to generate estimations for every data row, MARS should be used when each class in a system has to be estimated. This might be necessary for comparison issues as inspections. Using the presented techniques, conclusions and actions from past faults can be drawn assessing measurable software properties to prevent faults in future classes.

References

1. R.D. De Veaux, D.C. Psichogios, L.H. Ungar, A Comparison of two Nonparametric Estimation Schemes: MARS and Neural Networks, Computers Chemical Engineering, CACE Int. Journal, Vol 17, N. 8, pp. 819-837, 1993.
2. G. H. Dunteman. Principal Components Analysis. Sage , Newbury Park, CA, 1989.
3. J.C. Laprie, Dependability: concepts and terminology, LAAS-CNRS, Toulose, PDCS Meeting Durham, U.K., Oct. 11-13, 1989.
4. A. Bondavalli, L. Simoncini, Failure Classification with Respect to Detection, in Predictably Dependable Computing Systems -- First year Report, Task B, Volume 2, May 1990.
5. R. Neumann, D. Klemann, Metrication of object oriented complexity, SOQUA04, to appear, 2004.
6. H. Zuse, A Framework of Software Measurement, de Gruyter, Dec. 1997.
7. M. Lorenz, J. Kidd. Object-Oriented Software Metrics. Prentice Hall, Englewood Cliffs, N.J., 1994.
8. S.A. Whitmire, Object-Oriented Design Measurement, New York: Wiley, 1997.
9. L. Briand, J. Daly, J. Wüst, A Unified Framework for Coupling Measurement in Object-Oriented Systems, Technical Report ISERN-96-14, 1996.
10. S.R. Chidamber, C.F. Kemerer, A Metrics Suite for Object Oriented Design, IEEE Trans. on Software Eng., 20 no. 6, June 1994, pp. 476-493.
11. L.C. Briand, W.L. Melo, J. Wüst, Assessing the applicability of fault-proneness models across object-oriented software projects, IEEE trans. Software Engineering 28 no. 7, pp. 706-720, 2002.
12. R. Neumann, A Categorization for Object Oriented Software Metrics in Fault Prediction, proc. SMEF04, Rome, Italy, Jan. 2004
13. R.E. Courtney, D.A. Gustafson, Shotgun correlations in software measurement, Software Engineering Journal, Jan. 1993, pp. 5-13
14. S.N. Cant, B. Henderson-Sellers, D.R. Jeffery, Application of cognitive complexity metrics to object-oriented programs, Journal of Object-Oriented Programming, pp. 52-63, July-August 1994.
15. K. El-Emam, S. Benlarbi, N. Goel, S. Rai, The Confounding Effect of Class Size on the Validity of Object-Oriented Metrics, IEEE Transactions on Software Engineering 27 no.7, pp. 630-650, 2001.
16. D. Hand, H. Mannila, P. Smyth, Principles of Data Mining, MIT Press, 2001.
17. L. Francis: Martian chronicles: Is MARS better than Neural Networks?, proc. CAS casualty actuarial society forum Winter, 2003.
18. T. Foss, E. Stensrud, B. Kitchenham, I. Myrtveit, A Simulation Study of the Model Evaluation Criterion MMRE, IEEE trans Software Eng 29 no. 11, 2003.
19. G. Cooper, E. Herskovits, A Bayesian method for the induction of probabilistic networks from data, Machine Learning 9, pp. 309-347, 1992.