Prediction of Software Development Faults in PL/SQL Files using Genetic Nets

TONG-SENG QUAH, MIE MIE THET THWIN
School of Electrical & Electronic Engineering
Nanyang Technological University
S2-B2c-84, EEE, NTU, 639798
SINGAPORE

Abstract: - Database applications constitute one of the largest and most important software domains in the world. Some classes or modules in such applications are responsible for database operations. Structured Query Language (SQL) is used to communicate with database middleware in these classes or modules. It can be issued interactively or embedded in a host language. This paper aims to predict the software development faults in PL/SQL files using SQL metrics. Based on actual project defect data, the SQL metrics are empirically validated by analyzing their relationship with the probability of fault detection across PL/SQL files. SQL metrics were extracted from Oracle PL/SQL code of a warehouse management database application system. The faults were collected from the journal files that contain the documentation of all changes in source files. The result demonstrates that these measures may be useful in predicting the fault concerning with database accesses. In our study, genetic net is used to evaluate the capability of this set of SQL metrics in predicting the number of faults in database applications. This genetic net combines a genetic algorithm with a statistical estimator to produce a model which also shows the usefulness of respective inputs.

Key-Words: - SQL Metrics, Software Prediction, Genetic Net, Software Metrics

1 Introduction

Software metrics have been used as a quantitative means of assessing software development process as well as the quality of software products. Many researchers have studied the correlation between software design metrics and the likelihood of occurrence of software faults. They classified the software modules (or classes) as fault-prone or non-fault-prone modules (or classes) and predicted the number of faults in modules (or classes) using various software metrics [1..14].

Although database applications are essential to every organization, studies on product measures for static database operation statements are rarely found in literature. SQL is the standard language for relational database management systems. The relationship between the fault occurrence for database applications and SQL metrics is studied in this paper. We defined SQL metrics that have strong relationship with faults and then performed empirical validation for these metrics.

In our study, we analyzed the fault reports of development projects involving database applications using PL/SQL code and found that faults are related to the number of SQL statements and the complexity of SQL statements. SQL commands are mainly composed in the PL/SQL files to perform database operations. SQL complexity can be measured using product metrics such as the number of sub-queries: the number of group-by clause.

A variety of statistical techniques are used in software quality modeling. Models are often based on statistical relationships between measures of quality and measures of software metrics. However, relationships between static software metrics and quality factors are often complex and nonlinear, limiting the accuracy of conventional approaches. Artificial neural networks and genetic training strategy are adept at modeling nonlinear functional relationships that are difficult to model with other techniques, and thus, are attractive for software quality modeling. We introduce using neural network model with genetic training strategy to improve prediction results for estimating faults in PL/SQL codes using SQL metrics in this study.

2 SQL Metrics

Database applications constitute one of the largest and most important software domains in the world. Some classes or modules in those applications are responsible for handling database accesses. We analyzed the fault reports of these classes or modules and found that faults are related to the number of SQL statements, which are invoked from a class or module; and the complexity of SQL statements. For example, retrieving wrong database records. In such cases developers need to check and modify the corresponding SQL statements to correct the error. To be able to predict the number of such faults for these classes or modules are very important in developing database applications.

The following SQL metrics are defined and used in this study. Metrics having weak relationships with fault occurrences, such as Data Definition Language (DDL) commands and Data Control Language (DCL) commands are omitted in this study.

Total number of select commands (TNSC)
It measures the total number of select commands used in a class or a module. Select command is an executable command that can be issued interactively or embedded in a host language.

Total number of insert/update operations (TNIUO)
It measures the total number of invocation of insert/update operations in a class or module.

Total number of delete operations (TNDO)
It measures the total number of invocation of delete operations in a class or module.

Average number of search condition criteria in where-clause (ANSC)
Most relational database queries retrieve only a portion of the records contained in a table. The where-clause qualifies the query command statement to limit the data retrieved to specific rows in the table. It is generally used with a select statement to specify search criteria for retrieving rows of data from a table or group of tables. The where-clause can also appear in delete and update command statements. A predicate states selection criteria, which is applied to each row of data values in the table being queried by the select statement. The number of selection criteria is directly co-related to the complexity of SQL statement.

Total number of sub-queries (TNSQ)
A sub-query consists of two or more ordinary queries nested in such a way that the results of each inner query are used in the comparison test for the selection clause of the next outer query (or another command statement). We have found this metric is very important to measure the complexity of SQL statements.

Total number of group-by-clause (TNGB)
When a query statement includes a group-by clause, the select-clause for that query may list one or more aggregate functions (SUM, COUNT, AVG, etc.) operating on groups of data values in other columns. The group-by clause contains a column-list argument that must include all simple column names that appear in the select-clause. Simple column names are called grouping columns, since they group together records with identical data in that column. This record grouping allows mathematical operations to be performed on the other columns specified as arguments in the aggregate functions. Records can also be grouped by any valid expression appearing in the select-clause argument. In these cases, the expression must also be referenced in the group-by-clause by a number representing its relative column position in the select-clause (or output result). This metric counts the total number of select statements containing group-by-clause invoked from a particular application class.

3 Research Design

We use genetic training strategy of NeuroShell Predictor in this study. The genetic net combines a genetic algorithm with a statistical estimator to produce a model which also shows the usefulness of respective input variables. Genetic algorithms (GAs) seek to solve optimization problems using the methods of evolution, specifically, survival of the fittest. The functioning of the genetic estimator is based upon General Regression Neural Net (GRNN) [17]. Genetic learning stores every set of inputs and related output in the training data. When the neural network is presented with a new set of inputs, the new inputs are compared to every set of stored inputs. Depending upon how close the match is, the output for each training row is weighted. Closer matches receive higher weights, and inputs that are farther away from the training inputs receive lower weights. The predicted output for the new set of inputs is a “weighted” average of all of the outputs with which the network was trained.

3.1 Data collection

We collected the experiment data from a set of warehouse management applications (WMA) that was developed using C, JAM and PL/SQL languages. This set of applications has more than a thousand source files of C, JAM and PL/SQL codes and uses Oracle database. The warehouse system has been customized and used by many companies. Faults were collected from the journal files that contain the documentation of all changes in source files such as status of module, start date, end date, developer, nature of changes, etc. Data on software metrics were extracted from 103 PL/SQL files of the above mentioned warehouse application using metric extraction tool that we developed using VC++ incorporating MKS LEX & YACC utilities as embedded language.

3.2 Experiment

We used data collected from warehouse application system for prediction of software development faults. We extracted six software metrics from 103 pl/SQL files. It contains TNSC, TNDO, TNIUO, TNGB, TNSQ and ANSC metrics as described in Section 2. The dependent variable was the number of faults and the independent variables were the six software metrics. First, each data pattern was examined for erroneous entries, outliers, blank entries and redundancy. We set a threshold value 1000 for maximum number of generations without improvement. After performing 1621 generations, we arrived at the optimized coefficient of multiple determination (R-square) value of 0.737046.

To measure the goodness of fit of the model, we use the coefficient of multiple determination (R-square), the coefficient of correlation(r), mean square error (MSE) and root mean square error (RMSE). These statistical measures are shown in Table 1. The correlation of the predicted change and the observed change is represented by the coefficient of correlation (r). An r value of 0.860836 represents high correlations for cross-validation. The number of observations is 103. The significance level of a cross-validation is indicated by a p value. A commonly accepted p value is 0.05. In our experiment, a two tailed probability p value is less than 0.0001. This shows a high degree of confidence for the successful validations. The results clearly indicate close relationship between SQL metrics (independent variables) and the predicted number of faults (dependent variable). A graphical representation comparing the actual versus predicted number of faults in the warehouse application system is shown in Appendix A.

Table 2 shows the relative importance of input variables, i.e. the selected SQL metrics. It displays a list of input metrics and a corresponding number which indicates the importance of the variable in predicting the output. The Relative Importance of Inputs values signify the importance of the variables. The weight value ranges from 0 to 1. Weights near 0 signify the least important variables, while weights near 1 signify the most important. The genetic method gives more precise "importance" factors than what a pure neural network method does. The Genetic Training Strategy uses a "genetic algorithm" or survival of the fittest technique to determine a weighting scheme for the inputs. The genetic algorithm tests many weighting schemes until it finds the one that gives the best predictions for the training data.

Table 1. Experimental result for WMA system

R-square	0.737046
r (correlation coefficient)	0.860836
Avg error	0.266629
MSE	0.190009
RMSE	0.4359
t values	17.00059
p values	<0.0001

Table 2. Relative importance of Data Access Tier metrics for WMA system

Metrics	the relative importance
TNSQ	0.369
ANSC	0.185
TNGB	0.159
TNIUO	0.155
TNSC	0.129
TNDO	0.004

Table 2 clearly indicates that the total number of sub-queries is the most important factor in increasing the complexity of SQL statements.

4. Conclusions

We studied the relationship between fault occurrence for database applications and SQL metrics. First we proposed SQL metrics that have strong relationship with faults and then performed empirical validation for these metrics. We analyzed the fault reports kept by project teams of database applications using PL/SQL code and found that faults are related to the number of SQL statements and the complexity of SQL statements. The relationship between fault occurrences for database applications and SQL metrics has been empirically validated in this study. From the results presented above, our proposed SQL metrics in this study appear to be useful in predicting faults in PL/SQL files.

This empirical study has presented the prediction of the number of faults in PL/SQL files using genetic training strategy. The Genetic Training Strategy uses a “genetic algorithm” or survival of the fittest technique to determine a weighting scheme for the inputs. The genetic algorithm tests many weighting schemes until it finds the one that gives the best predictions for the training data. We collected 103 data patterns from the warehouse management system. For that number of patterns, the genetic training method is much better because the genetic method uses a "one hold out" strategy both during training and afterwards when evaluating new data [19]. Moreover, the genetic method generally gives more reliable importance factors than the neural method does [18].

These findings paved the way for future research into using genetic network for predicting software maintainability. In addition, our research results also provide a new avenue for software project manger to determine the readiness of software under development.

5. Future Plan

We intend to extend this investigation with wide range of applications and various types of data access techniques. Our future research direction aims to estimate software readiness by using metrics for defect tracking. To estimate readiness, three factors will be considered in our future study: (1) how many faults are remaining in the programs (2) how many changes are required to correct the errors and (3) how much time is required in changing the programs. Software metrics concerning with polymorphism, inheritance, complexity, cohesion, coupling, dynamic memory allocation, SQL and size will be used.

6. Acknowledgement

We gratefully acknowledged CAIB GmbH, Murrhardt, Germany for providing us with the software development data used in our research.

References:

[1] A. Mounir Boukadoum, Houari A. Sahraoui and Hakim Lounis, Machine Learning Approach to Predict Software Evolvabilit using fuzzy binary trees, *International Conference on Artificial Intelligence*, 2001

[2] L.C. Briand, W.L Melo, J. Wust, Assessing the applicability of fault-proneness models across object-oriented software projects, *IEEE Transactions on Software Engineering*, Vol. 28, 2002, pp. 706 –720.

[3] El Emam, A primer on object-oriented measurement, *Proceedings of the Seventh International Software Metrics Symposium*, 2001, pp. 185 –187.

[4] El Emam, W. Melo, C.M. Javam, The Prediction of Faulty Classes Using Object-Oriented Design Metrics, *Journal of Systems and Software*, Vol. 56, issue 1, 2001, pp. 63-75.

[5] N.E. Fenton and N. Ohlsson, Quantitative analysis of faults and failures in a complex software system, *IEEE Transactions on Software Engineering*, Vol. 26, 2000, pp. 797-814.

[6] F. Fioravanti, P. Nesi, A study on fault-proneness detection of object-oriented systems, *Fifth European Conference on Software Maintenance and Reengineering*, 2001, pp. 121 –130, 2001

[7] John E. Frenund, Frank J. Williams., Benjamin M. Perles, *The Elementary Business Statistics-The Modern Approach*, Prentice-Hill, 1993

[8] Todd L. Graves, Alan F. Karr, J.S. Marron, and Harvey Siy, Predicting Fault Incidence Using Software Change History, *IEEE Transactions on Software Engineering*, Vol. 26, No. 7, 2000, pp. 653-661.

[9] T. M. Khoshgoftaar, E.B. Allen, Z. Xu, Predicting testability of program modules using a neural network, *Proceedings of the 3rd IEEE Symposium on Application-Specific Systems and Software Engineering Technology*, 2000, pp. 57-62.

[10] Thet Thwin Mie Mie, Jon T.S. Quah, Application of Neural Network for Predicting Software Development Faults Using Object-Oriented Design Metrics, *Proceedings of the 9th International Conference on Neural Information Processing*, 2002, Singapore

[11] Thet Thwin Mie Mie, Tong-Seng Quah, Application of Neural Networks for Estimating Software Maintainability Using Object-Oriented Metrics, *Proceedings of the 15th International Conference on Software Engineering and Knowledge Engineering*, 2003, pp. 69-73, San Francisco, U.S.A

[12] Jon T.S. Quah, Mie Mie Thet Thwin, Prediction of Software Readiness Using Neural Network, *Proceedings of the IEEE International Conference on Information Technology and Applications*, 2002, Sydney, Australia

[13] Tong-Seng Quah, Mie Mie Thet Thwin, Application of Neural Networks for Software Quality Prediction Using Object-Oriented Metrics, *Proceedings of the International Conference on Software Maintenance*, 2003, Amsterdam, The Netherlands

[14] Tong-Seng Quah, Mie Mie Thet Thwin, Prediction of Software Development Faults in PL/SQL Files Using Neural Network Models, *Information and Software Technology*, Vol. 46 No. 8, 2004, pp 519-523.

[15] Ramarkrishnan, Gehrke, *Database Management Systems*, McGRAW-Hill, 3rd Edition, 2003.

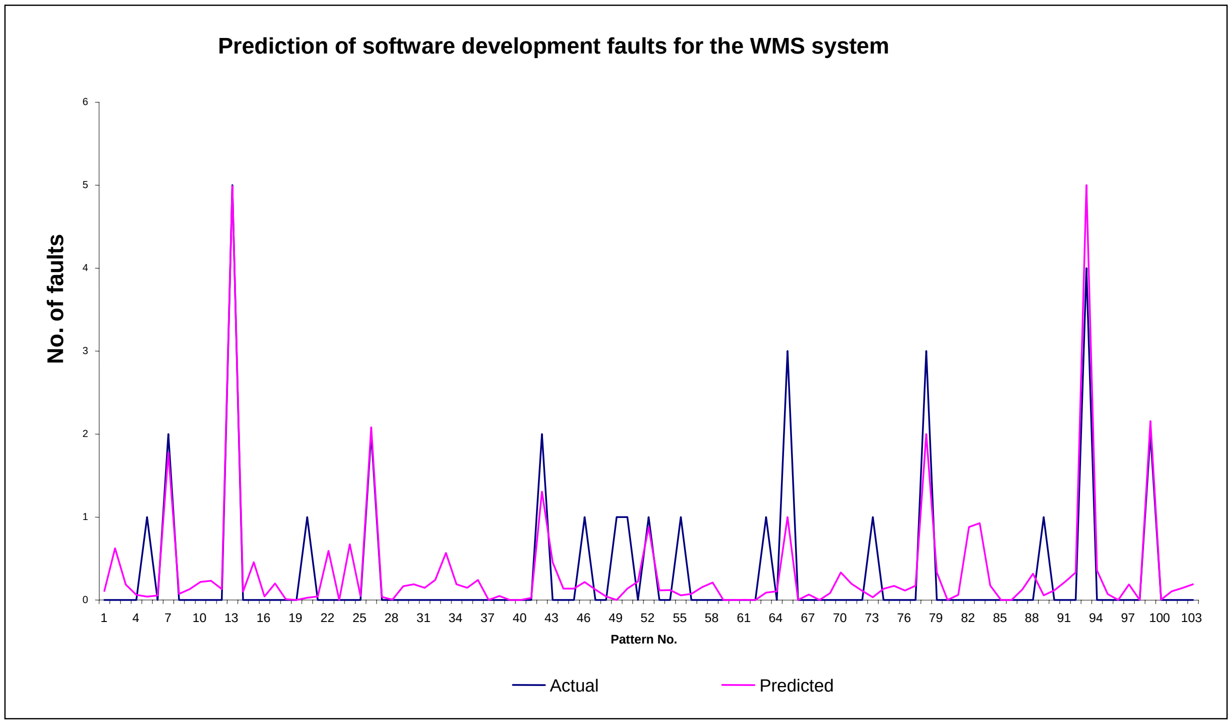
[16] Frenund, John E., Williams, Frank J., Perles Benjamin M., *The Elementary Business Statistics- The Modern Approach*, Prince-Hill, 1993

[17] D.F, Specht, A general regression neural network, *IEEE Transactions on Neural Networks*, Vol. 2, Issue: 6, 1991, pp. 568-576.

[18] C.H. Chen, *Fuzzy Logic and Neural Network Handbook*, New York, N.Y.: McGraw-Hill, Inc., 1996

[19] NeuroShell Predictor Help, Ward Systms Group, Inc. http://www.wardsystems.com

Appendix A



Prediction of software development faults for the WMS system
No. of faults
0
1
2
3
4
5
6
1
4
7
10
13
16
19
22
25
28
31
34
37
40
43
46
49
52
55
58
61
64
67
70
73
76
79
82
85
88
91
94
97
100
103
Pattern No.
Actual
Predicted