Application of Neural Networks for Software Quality Prediction Using Object-Oriented Metrics

Tong-Seng Quah, Mie Mie Thet Thwin
School of Electrical & Electronic Engineering
Nanyang Technological University
ITSQuah@.ntu.edu.sg

Abstract

This paper presents the application of neural networks in software quality estimation using object-oriented metrics. Quality estimation includes estimating reliability as well as maintainability of a software. Reliability is typically measured as the number of defects. Maintenance effort can be measured as the number of lines changed per class. In this paper, two kinds of investigation are performed. The first on predicting the number of defects in a class and the second on predicting the number of lines change per class. Two neural network models are used, they are Ward neural network and General Regression neural network (GRNN). Object-oriented design metrics concerning inheritance related measures, complexity measures, cohesion measures, coupling measures and memory allocation measures are used as the independent variables. GRNN network model is found to predict more accurately than Ward network model.

1. Introduction

Many object-oriented metrics have been proposed over the last decade. Prediction models using object-oriented design metrics can be used for obtaining assurances about software quality. In practice, quality estimation means either estimating reliability or maintainability. Reliability is typically measured as the number of defects. These can be pre-release or post-release. The estimated number of defects can also be normalized by a size measure to obtain a defect density estimate. Maintainability is typically measured as change effort. Change effort can mean either the average effort to make a change to a class, or the total effort spent on changing a class.

Khoshgoftarr et al. introduced the use of the neural networks as a tool for predicting software quality. In [24], they presented a discriminant model and a neural network model of the large telecommunications system, classifying modules as not fault-prone or fault-prone. They compared the neural-network model with a nonparametric discriminant model, and found the neural network model had better predictive accuracy.

We conduct our study in the object-oriented paradigm. However since the object-oriented paradigm exhibits different characteristics from the procedural paradigm, different software metrics have to be defined and used.

Our neural network model aims to predict object oriented software quality by estimating the number of faults and the number of lines changed per class. We used software metrics including both object-oriented metrics and traditional complexity metrics. Object oriented metrics used include inheritance related measures, cohesion measures and coupling measures.

We also introduce using Ward neural network and General Regression neural network to improve prediction result for estimating software quality. Ward neural network is a backpropagation network with different activation functions. They are applied to hidden layer slabs to detect different features in a pattern processed through a network to lead to better prediction. We use a Gaussian function in one hidden slab to detect feature in the mid-range of the data and a Gaussian complement in another hidden slab to detect features for the upper and lower extremes of the data. Thus, the output layer will get different "views of the data". Combining the two feature sets in the output layer leads to a better prediction.

Another architecture that we have chosen is the General Regression Neural Network (GRNN). Specht [23] state that it is a memory-based network that provide estimates of continuous variables and converges to the underlying (linear or nonlinear) regression surface. This is a one-pass learning algorithm with a highly parallel structure. Even with sparse data is a multidimensional measurement space; the algorithm provides smooth transitions from one observed value to another.

2. Related work

There is great interest in the use of object-oriented approach in software engineering. With the increasing use of object-oriented methods in new software development there is a growing need to both document and improve

Proceedings of the International Conference on Software Maintenance (ICSM'03)
1063-6773/03 $17.00 © 2003 IEEE

current practices in object-oriented design and development.

Many measures have been proposed in the literature to capture the quality of object-oriented (OO) code and design and used for detecting fault-proneness of classes [3, 4, 6, 9, 17]. Many investigations using statistical methods had been made to predict software quality.

Emanm and Melo [4] have constructed a model to predict which classes in a future release of a commercial Java application will be faulty. The model was then validated on a subsequent release of the same application. Their results indicated that the prediction model had a high accuracy.

Fioravanti and Nesi have extracted over 200 different object-oriented metrics to identify a suitable model for detecting fault-proneness of classes [9]. They came to the conclusion that only few of them were relevant for identifying fault-prone classes.

A set of object-oriented metrics in terms of their usefulness in predicting fault-proneness, an important software quality indicator is empirically validated in [21]. Their validation is carried out using two data analysis techniques: regression analysis and discriminant analysis.

L. Briand et al., the relationships between existing object-oriented coupling, cohesion, and inheritance measures and the probability of fault detection in system classes during testing explored empirically. Their univariate analysis have shown that many coupling and inheritance measures are strongly related to the probability of fault detection in a class. Their multivariate analysis results showed that by using some of the coupling and inheritance measures, very accurate models could be derived to predict in which classes most of the faults actually lie [11].

Most of these prediction models are built using statistical models. Neural networks have seen an explosion of interest over the years, and are being successfully applied across a range of problem domains, in areas as diverse as finance, medicine, engineering, geology and physics. Indeed, anywhere that there are problems of prediction, classification or control, neural networks are being introduced. Neural network can be used as a predictive model because it is very sophisticated modeling techniques capable of modeling complex functions.

In [2], Khoshgoftaar et al presented a case study of real-time avionics software to predict the testability of each module from static measurements of source code. They found that neural network is a promising technique for building predictive models, because they are able to model nonlinear relationships.

Our neural network model aims to predict object oriented software quality by estimating the number of faults and the number of lines changed per class. We also introduce using Ward neural network and General Regression neural network to improve prediction result for estimating software quality.

3. Design of the study

3.1. Object-oriented metrics

As discussed in section I, we are introducing the research on software defects and maintenance efforts predictions into object oriented paradigm using neural networks. As such object oriented metrics have to be selected and used in our study. To detect the software defects and predict the maintenance effort, the following metrics are used:

Depth of Inheritance Tree (DIT) of a class is the length of the longest path from the class to the root in the inheritance hierarchy. This determines the complexity of a class based on its ancestors, since a class with many ancestors is likely to inherit much of the complexity of its ancestors. The deeper a class is in the hierarchy, the greater the number of methods it is likely to inherit making it more complex to predict its behavior. This has direct relationship to maintainability.

Number of Children (NOC) measures the number of immediate descendants of a particular class. This measures an amount of potential reuse of the class. The more reuse a class might have, the more complex it may be, and the more classes are directly affected by changes in it implementation. This increases the magnitude of ripple effects.

Coupling Between Objects (CBO) is defined as the number of other classes to which it is coupled. Coupling metrics measure the degree of inter dependence among the components of a software system. High coupling makes a system more complex; highly interrelated modules are harder to understand, change or correct. By minimize coupling, propagating errors across modules can be avoided.

Response For a Class (RFC) is the number of methods that can potentially be executed in response to a message received by an object of that class. The response set of a class consists of the set of M methods of the class, and the set of methods directly or indirectly invoked by methods in M.

Inheritance Coupling (IC) provides the number of parent classes to which a given class is coupled. A class is coupled to its parent class if one of its inherited methods is functionally dependent on the new or redefined methods in the parent class.

Coupling Between Methods (CBM) provides the total number of new/redefined methods in which all the inherited methods are coupled. CBM measures the total number of function dependency relationships between the inherited methods and new/redefined methods.

Weighted Methods per Class (WMC) is the summation of McCabe's cyclomatic complexity of each local method. The more control flows a class's methods have, the harder it is to understand them, thus, the harder it is to maintain them. A method with a low cyclomatic complexity is generally better.

Weighted Methods per Class (WMC1) is defined as the number of all member functions and operators in each class.

Number of Object/Memory Allocation (NOMA) metric measures the total number of statements that allocates new objects or memories in a class. A class with more object/memory allocating activities tends to introduce more the object management faults that are related to object management such as object copying, dangling reference, object memory usage faults and so on.

Message Passing Coupling (MPC) gives an indication of how many message are passed among objects of the classes. The number of messages sent out from a class indicates how dependent the implementation of the local methods is on the methods in other classes.

Lack of Cohesion in Methods (LCOM) is the number of pairs of methods in the class using no attributes in common, minus the number of pairs of methods that do. If this difference is negative, LCOM is set to zero.

Data Abstraction Coupling (DAC) is the number of attributes in a class that have as their type another class.

The number of local methods (NOM) defined in a class indicates the operation property of a class. The more methods a class has, the more complex will be the class's interface.

SIZE1 is calculated by counting the number of executable statements (measured by number of semicolons) in a class.

SIZE2 is the total number of attributes and methods of a class.

3.2. Neural network modeling

The first neural network architecture that we have chosen is the Ward Network[25]. It is a Backpropagation network that has three slabs (slab2, slab3 and slab4) in the hidden layer. Hidden layers in neural network are known as feature detectors. A slab is a group of neurons. When each slab in the hidden layer has a different activation function, it offers three ways of viewing the data. We use linear function to the output slab (slab5). We use hyperbolic tangent (tanh) function is used in one slab of hidden layer (slab3) because it is better for continuous valued outputs especially if the linear function is used on the output layer. Gaussian function is used in another slab of the hidden layer (slab2). This function is unique, because unlike the others, it is not an increasing function. It is the classic bell shaped curve, which maps high values

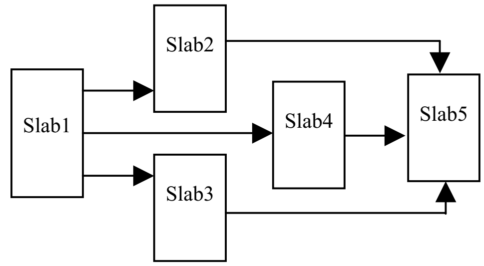

Figure 1. Ward Neural Network.

into low ones, and maps mid-range values into high ones. Gaussian Complement is used in the third slab of the hidden layer (slab4) to bring out meaningful characteristics in the extremes of the data. The learning rate and momentum are set to 0.1 and initial weight is set to 0.3 in this study.

Another neural network architecture that we have chosen is the General Regression Neural Network (GRNN). GRNN is based on a one-pass learning algorithm with a highly parallel structure. GRNN is a powerful memory based network that could estimates continuous variables and converges to the underlying regression surface. The strength of GRNN is that it is able to deal with sparse data effectively. Specht [23] claims that the algorithm in GRNN is able to provide a smooth transition from one observed value to another, even with sparse data in a multidimensional measurement space. GRNN applications are able to produce continuous valued outputs. For GRNN networks, the number of neurons in the hidden layer is usually the number of patterns in the training set because each pattern in the training set is represented by on neurons. The primary advantage to the GRNN is the speed at which the network can be trained. Training a GRNN is performed in one pass. The smoothing factor allows the GRNN to interpolate between the patterns or spectra in the training set.

3.3. Principal component analysis

If a group of variables in a data set are strongly correlated, these variables are likely to measure the same underlying dimension (i.e., class property) of the object to be measured. Many object-oriented metrics have high correlation with each other. For example, the number of local method (NOM) is strongly correlated with class size. The confounding effect of class size is studied in [7]. Principal component analysis (PCA) is a standard technique to identify the underlying, orthogonal dimensions that explain relations between the variables in a data set. Principal components (PCs) are linear combinations of the standardized independent variables. It is also a data reduction technique. The varimax rotation method was adopted in this study. It is an orthogonal rotation method that minimizes the number of variable that have high loadings on each factor. It simplifies the interpretation of the factors. We selected the PCs only PCs whose eigenvalue is larger than 1.0.

4. Prediction of maintenance effort

This investigation is to predict the maintenance effort. The commercial software product QUES(Quality Evaluation System) data is used in this investigation, which is presented in [20]. The maintenance effort is measured by using the number of lines changed per class. A line change could be an addition or a deletion. A change of the content of a line is counted as a deletion followed by an addition. This measurement is used in this study to estimate the maintainability of the object-oriented systems. In this study, DIT, MPC, RFC, LCOM, DAC, WMC, NOM, SIZE1 and SIZE2 are used as independent variable. Their correlation matrix is shown in Appendix C. QUES system was designed and developed with Class-Ada. First, each data pattern was examined for erroneous entries, outliers, blank entries and redundancy. After standardizing the metric data, we performed the principal component analysis. Table 1 presents the relationship between the original object-oriented metrics and the domain metrics for QUES system.

For QUES system, PCA identified three PCs, which capture 89% of the data set variance; Table 1 shows for each rotated component the coefficients of the measure, with coefficients larger than 0.6 set in boldface. The eigen value, the percentage of the data set variance each PC describes, and the cumulative variance percentage are also provided. Based on the analysis of the coefficients associated with each metrics within each of the three rotated components, the PCs are interpreted as follows:

The first component is highly correlated with NOM, SIZE2, RFC, LCOM, WMC, SIZE1 and DAC. NOM is a better representative, however, because it is less correlated with the other two components. The second component is most highly correlated with MPC. The third component is most highly correlated with DIT. This suggests that NOM, MPC and DIT metrics should be focused on further analysis for this system.

We sorted the data according to the number of changes values and divided data into training, testing, and production sets using 3:1:1 ratio. Test set is used to prevent over training network so they will generalize well. We used the production data set to evaluate model performance. It can be tested the network's results with the data the network has never seen before.

We used Ward network and GRNN network for predicting number of changes. Table 2 shows the summary of Ward network design. In our General Regression neural network design, there were 71 neurons in hidden layer, 3 neurons in input layer and 1 neuron in output layer. The experimental results that are obtained from the Ward model and GRNN neural network model for QUES system are tabulated in Appendix A.

Table 1. Rotated principle components for QUES system

metrics	PC1	PC2	PC3
DIT	0.060	0.027	**0.966**
MPC	-0.023	**0.966**	0.037
RFC	**0.877**	0.333	0.043
LCOM	**0.869**	-0.156	0.059
DAC	**0.796**	0.027	0.427
WMC	**0.832**	0.258	-0.27
NOM	**0.971**	-0.132	0.097
SIZE1	**0.812**	0.475	-0.089
SIZE2	**0.963**	-0.093	0.190
Eigenvalues	5.384	1.388	1.248
% Variance	59.826	15.424	13.863
Cummulative % Variance	59.826	75.250	89.113

Table 2. Ward neural network architecture used for QUES system

	Slab1	Slab2	Slab3	Slab4	Slab5
No. of neurons	3	3	3	3	1

4.1. Goodness of fit test

To measure the goodness of fit of the model, we use the coefficient of multiple determination (R-square), the coefficient of correlation(r), r-square, mean square error, mean absolute error, minimum absolute error and maximum absolute error. These statistical measures are shown in Table 3. The correlation of the predicted change and the observed change is represented by the coefficient of correlation (r). An r value of 0.747 in Ward neural network and 0.8590 in GRNN network represents high correlations for cross-validation. The number of observations is 71. The significance level of a cross-validation is indicated by an p value. A commonly accepted p value is 0.05. An two tailed probability p values of 0.000 in both cross-validation shows a high degree of confidence for the successful validations. We conclude that the impact of model prediction is valid in the population.

Table 3. Experimental result for QUES system

	Ward	GRNN
R-square	0.5545	0.7220
r (correlation coefficient)	0.747	0.8590
r- square	0.558	0.7379
Mean square error	817.004	509.790
Mean absolute error	20.782	12.182
Min absolute error	0.094	0
Max absolute error	114.161	109.385
t values	9.329047	13.98484
p values	0.000	0.000

5. Prediction of number of faults

The second investigation is emphasized on the prediction of the number of faults. Faults are appeared when a program does not perform according to users' specification at testing and operations stages. The applications used in this prediction are three subsystems of a HMI (Human Machine Interface) software, which is a fully networked Supervisory Control and Data Acquisition system. This software, which consists of more than 200 subsystems and 3 million lines of code, has been used by many manufacturing companies for several years. Although each subsystem selected plays a different role in the system and performs a different functionality, they share some similar characteristics that meet with our selection criteria. Subsystem A is a user interface-oriented program that allows customers to configure the basic product operations and device communications. It consists of 20 classes that define 256 new, re-defined or virtual functions, and approximately 5,600 lines of code in length. Subsystem B is a real time data logging process that collects data as needed and logs data into the database, based on the user configuration. This subsystem defines 48 classes and 353 new, re-defined or virtual functions, comprising approximately 21,300 lines of code. Subsystem C is a communication-oriented program that acts as a router not only delivering messages between processes within the same host but also forwarding messages to other hosts. This subsystem defines 29 classes and 293 new, re-defined or virtual functions and contains approximately 16,000 lines of code [5].

In this investigation, we used WMC1, DIT, NOC, CBO, RFC, IC, CBM and NOMA as independent variables. Their correlation matrix is shown in Appendix D.

After standardizing the metric data, we performed the principal component analysis. Table 4 presents the relationship between the original object-oriented metrics and the domain metrics for HMI system.

After performing PCA, it identified three PCs, which capture 78.29% of the data set variance as shown in Table 4. The first component is highly correlated with RFC, WMC1, CBM, IC and NOMA. The second component is most highly correlated with NOC and CBO. The third component is most highly correlated with DIT.

Table 4. Rotated Principle Components for HMI system

Metrics	PC1	PC2	PC3
WMC1	**0.9068**	-0.0544	-0.1768
DIT	-0.0358	-0.0400	**0.9286**
NOC	-0.0274	**0.8710**	-0.0694
CBO	-0.0043	**0.8508**	0.0471
RFC	**0.9399**	-0.0818	-0.0919
IC	**0.6452**	0.1678	0.5140
CBM	**0.8636**	0.0811	0.2932
NOMA	**0.6216**	-0.1029	0.4508
Eigenvalues	3.256	1.539	1.462
% Variance	40.703	19.238	18.279
Cummulative % Variance	40.703	59.940	78.219

Table 5. Ward neural network architecture used for HMI system

	Slab1	Slab2	Slab3	Slab4	Slab5
No. of neurons	3	4	4	4	1

Ward Neural design summary is presented in Table 5. For General Regression neural network we use 97 neurons in the hidden layer as that is the number of patterns in the collected data. There are 3 neurons in the input layer and 1 neuron in the output layer in our GRNN network design. The neural network results that are obtained from the ward model and GRNN neural network model for HMI system are tabulated in Appendix B.

5.1. Goodness of fit test

Goodness of fit measures is shown in Table 6. The value of coefficient of correlation (r) value of 0.9476 in Ward neural network and 0.9531 in GRNN network represents high correlations for cross-validation. The p value for HMI system 0.000 and shows a high degree of confidence for the successful validations.

Table 6. Experimental result for HMI system

	Ward	GRNN
R-square	0.8715	0.9077
r (correlation coefficient)	0.9476	0.9531
r- square	0.8979	0.9084
Mean square error	1.584	1.138
Mean absolute error	0.823	0.765
Min absolute error	0.001	0
Max absolute error	6.211	4.295
t values	28.88816	28.88154
p values	0.000	0.000

6. Conclusion

This empirical study presents the prediction of faults and maintenance effort using two neural network models. From the results presented above, object-oriented metrics chosen in this study appear to be useful in predicting software quality. GRNN network model is found to predict more accurately than Ward network model. We also performed multivariate regression models to compare neural network models. Regression analysis results are shown in Appendix E.

Our future research direction aims to estimate the software readiness using neural network models. To estimate the 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 measures, inheritance related measures, complexity measures, cohesion measures, coupling measure, dynamic memory allocation measure, database operations measures and size measures will be used.

7. Acknowledgement

The authors would like to thank Associate Professor Dr. Mei-Hwa Chen, Computer Science Department, University at Albany, State University of New York, for allowing us to test our model with data they had collected from industrial real-time systems [5].

8. References

[1] El Emam, ”A primer on object-oriented measurement”, Proc. Seventh International Software Metrics Symposium, 2001, pp. 185 –187.

[2] 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.

[3] L.C. Briand, J.W. Daly, J.K. Wust, “A unified framework for coupling measurement in object-oriented systems”, IEEE Transactions on Software Engineering, 1999, pp. 91-121.

[4] El Emam, W. Melo, C.M. Javam, “The Prediction of Faulty Classes Using Object-Oriented Design Metrics”, Journal of Systems and Software, Elsevier Science, 2001, pp. 63-75.

[5] Mei-Huei Tang, Ming-Hung Kao, Mei-Hwa Chen,”An empirical study on object-oriented metrics”, Proceedings of the Sixth IEEE International Symposium on Software Metrics, 1999, pp. 242-249.

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

[7] El Emam, K., Benlarbi, S., Goel, N. and Rai, S.N., “The confounding effect of class size on the validity of object-oriented metrics”, IEEE Transactions on Software Engineering, vol. 27, pp. 630-650, 2001.

[8] F. Fioravant, “A metric framework for the assessment of object-oriented systems”, Proceedings of IEEE International Conference on Software Maintenance, pp. 557–560, 2001.

[9] 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.

[10] Ralf ReiBing, “Towards a Model for Object-Oriented Design Measurement”, Proceedings of the 5th International ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering, pp. 71-84, 2001.

[11] L. Briand, J. Wüst, John W. Daly and V. Porter, "Exploring the Relationships between Design Measures and Software Quality in Object-Oriented Systems", Journal of Systems and Software, 51(2000) p 245-273.

[12] M. Cartwright, and M. Shepperd, "An Empirical Investigation of Object Oriented Software System", IEEE Transactions on Software Engineering, vol. 26, pp. 786-796, 2000.

[13] L. Etzkorn, H. Delugach, “Towards a semantic metrics suite for object-oriented design”, Proc. 34th International Conference on Technology of Object-Oriented Languages and Systems, 2000, pp. 71 –80, 2000.

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

[15] D. Glasberg, K. El Emam, W. Melo, and N. Madhavji, "Validating Object-oriented Design Metrics on a Commercial Java Application". Technical Report, NRC/ERB-1080, NRC 44146, 2000.

[16] 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, 2000.

[17] 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 pp. 706 –720, 2002.

[18] Mie Mie Thet Thwin and Tong-Seng Quah, "Application of Neural Network for Predicting Software Development Faults Using Object-Oriented Design Metrics", Preceeding of 9th Internatinal Conference on Neural Information Processing, Singapore, 18-22 Nov, 2002, vol.5, pp. 2312, 2002.

[19] Jon T. S. Quah and Mie Mie Thet Thwin, "Prediction of Software Readiness Using Neural Network", Proceedings of 1st International Conference on Information Technology & Applications (ICITA 2002), Australia, 25-28 Nov, 2002.

[20] Wei Li and S. Henry, "Object-Oriented Metrics that Predict Maintainability", Journal of Systems and Software, 1993, pp. 111-122, 1993.

[21] Ping Yu, T. Systa, and H. Muller, "Predicting fault-proneness using OO metrics. An industrial case study", Proceedings. of 6th European Conference on Software Maintenance and Reengineering, 2002, pp. 99 –107, 2002.

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

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

[24] T.M. Khoshgoftaar, E.B. Allen, J.P. Hudepohl, and S.J. Aud, "Application of neural networks to software quality modeling of a very large telecommunications system", IEEE Transactions on Neural Network, vol. 8, pp. 902-909, 1997.

[25] NeuroShell 2 Help, Ward Systms Group, Inc. http://www.wardsystems.com.

Appendix A. Neural network results for QUES system

PC1	PC2	PC3	Observed change	Predicted change (Ward)	Predicted change (GRNN)
-0.13022	-1.64677	0.17929	6	12	7
-0.03039	-1.53992	0.24768	8	15	7
-0.3384	-1.44312	1.88978	9	20	9
-0.06182	-1.31265	0.15152	10	19	12
0.38454	-1.7059	-1.52689	14	50	24
-0.20789	-1.19234	0.30152	16	23	16
0.32482	-1.22358	0.13149	24	27	24
0.32482	-1.22358	0.13149	24	27	24
-0.51034	0.17986	3.72303	24	47	25
-0.52847	-0.37103	-1.49814	24	56	26
-0.45703	-0.15483	1.22508	25	55	25
-0.80264	0.53155	0.3626	26	70	59
-0.7083	-0.90828	-1.55791	26	44	26
-0.63144	-0.60917	-1.66117	28	51	26
0.17605	-1.07765	0.14892	28	26	24
-0.77388	-0.04587	0.29804	30	63	56
-0.77388	-0.04587	0.29804	30	63	56
-0.49412	-0.06956	0.19435	35	61	41
0.64859	-1.44491	0.16346	35	43	25
-0.77715	-0.69971	0.14314	38	51	52
0.68819	-0.78223	0.24383	38	42	38
0.78567	-0.05805	0.43647	38	56	47
-0.48266	-0.07729	0.15099	39	61	41
-0.52727	-0.1953	-1.57115	41	60	47
0.83053	-0.87824	0.17281	41	48	44
-0.93294	1.08382	0.31973	42	72	42
-0.82812	2.38145	0.49921	45	77	67
0.90263	-0.82797	-0.12397	45	54	45
-0.9431	2.11841	0.56337	46	75	86
0.34002	-0.59535	0.02217	47	40	38
-0.79137	0.53457	0.22929	48	70	70
-0.7862	0.43916	0.21822	48	69	76
-0.42231	0.40656	-1.23877	48	73	48
-0.54093	1.46946	0.46461	49	78	42
-0.76177	-0.30229	0.18144	52	59	67
-0.76478	-0.87679	0.33081	52	45	52
0.96278	-1.07335	0.15908	55	57	55
0.88353	0.26136	0.30633	56	66	58
-0.23708	1.60511	-1.31086	56	97	48
-0.84321	-0.49872	0.16672	62	56	65
-0.75731	-0.29668	0.18112	64	59	67
0.87751	0.03081	0.32011	68	60	58
-0.77864	-0.11337	0.19901	68	63	62
-0.28113	1.00686	-0.47416	70	82	57
2.02079	0.08606	-0.14748	70	99	72
1.81176	-0.139	0.08723	72	90	72
-0.7308	-0.06926	0.14359	75	63	64
-0.7241	-0.16134	0.13129	77	61	66
-0.75171	-0.17367	0.18025	78	61	65
-0.82307	0.15626	0.24549	79	66	65

1.57737	0.9669	-0.11944	80	92	80
1.41936	0.03292	-0.06634	81	73	81
-0.75592	-0.12406	0.16382	82	62	64
-0.71506	0.07609	0.2497	85	65	61
-0.66198	0.23844	0.18145	85	67	85
-0.72755	0.1471	0.15804	86	66	72
2.09191	0.50048	0.40952	88	93	88
2.10995	0.62771	0.00479	92	96	92
-0.71461	0.0112	0.12623	94	64	64
-0.01494	0.18925	-2.16237	98	76	48
-0.70921	0.458	0.14293	100	70	88
2.29648	1.40413	0.00921	101	108	101
0.65647	-1.64284	-3.65638	102	106	102
-0.65662	0.31795	0.16987	107	68	93
-0.64157	0.30697	0.17911	124	68	93
3.44685	0.82978	2.367	146	146	146
-0.72861	0.46487	0.17959	148	70	84
-0.73903	2.25183	0.30855	157	79	136
2.2554	0.18271	-0.33718	170	112	92
-0.59688	0.81619	0.16262	188	74	79
0.78041	3.48673	-2.90284	217	221	217

Appendix B. Neural network results for HMI system

pc1	pc2	pc3	actual	Predicted faults (GRNN)	Predicted faults (Ward)
-0.47911	0.79062	-0.02789	0	0	0
-0.7703	-0.35763	-0.34213	0	0	0
-0.40293	-0.39306	-0.5998	0	1	0
-0.69014	-0.104	0.18681	0	1	0
-0.0584	-0.28096	-0.80344	0	1	1
-0.46141	-0.23176	-0.5528	0	1	0
-0.63848	-0.31905	-1.01192	0	1	0
-0.50177	-0.5772	-0.28418	0	1	0
0.16403	-0.44226	-1.29833	0	1	2
-0.84585	-0.49019	1.11544	0	1	0
-0.7673	-0.44644	0.48293	0	1	0
-0.85166	-0.26875	1.96462	0	0	0
-0.51116	0.02657	3.4064	0	0	0
-0.35529	2.22454	0.64489	0	0	0
-0.77499	-0.25197	0.24186	0	1	0
-0.88265	-0.48719	1.14293	0	1	0
-0.61929	1.02751	-1.13168	0	0	0
-0.40885	0.11745	-1.2099	0	1	0
-0.59985	3.87507	-0.86161	0	0	0
-0.50717	-0.18786	-1.1422	0	1	0
-0.60115	-0.01121	-1.02648	0	1	0
-0.50228	-0.32963	-1.11537	0	1	0
-0.53881	-0.1712	-1.08141	0	1	0
-0.67117	-0.52063	-0.42455	0	0	0
-0.59208	-0.52736	-0.4828	0	0	0
-0.73872	-0.2045	-0.3599	0	0	0
-0.34673	0.5668	-1.2802	0	0	0
-0.28493	0.09597	-1.34806	0	0	0
-0.20724	1.07102	0.06917	0	0	0
-0.21197	0.77324	0.10073	0	1	0
-0.56078	-0.52964	-0.50704	0	0	0
-0.73872	-0.2045	-0.3599	0	0	0
-0.7022	-0.36292	-0.39386	0	0	0
-0.27518	-0.2978	0.14517	0	1	1
-0.49272	-0.22947	-0.52857	1	0	0
-0.33868	-0.39967	-0.6436	1	1	0
-0.09922	-0.16542	0.04729	1	2	1
-0.45972	-0.17793	-1.13966	1	1	0
-0.13741	-0.13137	1.15569	1	2	1
-0.72658	0.52299	1.24796	1	2	0
-0.31712	0.07205	1.74602	1	1	1
-0.73714	-0.19171	1.05849	1	1	0
0.45253	0.09601	3.38005	1	1	1
-0.56248	-0.34856	0.5976	1	1	0
-0.65886	-0.30338	0.41951	1	1	0
0.00195	0.09154	0.9481	1	3	1
-0.66682	-0.26433	0.17197	1	1	0
-0.57293	-0.46829	0.3562	1	1	0
-0.44387	-0.2412	0.76912	1	2	1
-0.16736	-0.30272	0.81768	1	2	1
-0.46543	0.43214	-1.15177	1	0	0
0.21242	-0.07986	-0.60118	1	1	1
-0.13191	0.41002	-1.33605	1	0	0
-0.08498	-0.25779	-1.13254	1	1	1
-0.64481	-0.21135	-0.4326	1	0	0
-0.61214	-0.37109	-0.45863	1	0	0
-0.69698	-0.51906	-0.40358	1	0	0
-0.66018	-0.52207	-0.43108	1	0	0
-0.66018	-0.52207	-0.43108	1	0	0
-0.21504	-0.35091	-1.33673	1	1	1
-0.04826	-0.51376	-0.31157	2	1	1
0.23161	-0.33306	1.2071	2	3	2
0.36896	-0.0493	-0.91835	2	1	2
0.01816	-0.23027	1.23626	2	3	2
0.381	-0.24819	-0.04207	2	3	2
0.13769	-0.48375	-0.72503	2	1	2

0.41256	-0.29215	0.19709	2	3	2
-0.22579	0.27821	1.77756	2	1	1
-0.10349	-0.21101	1.01439	2	2	1
0.37882	0.44969	0.57222	2	3	2
-0.24968	-0.07865	0.97012	2	2	1
1.28103	0.56944	0.17506	2	3	4
0.07336	-0.29396	0.05332	2	2	1
-0.29523	-0.39262	-0.70889	2	1	1
1.2526	-0.02	-0.2411	3	5	4
0.25962	0.06929	1.16248	3	3	2
0.17405	7.87669	-0.36831	3	3	3
0.09298	-0.06407	0.16382	3	2	1
-0.02974	-0.48558	-0.70129	3	1	1
0.43861	0.12657	-0.23085	4	3	2
0.95509	-0.22802	-0.14143	4	4	4
-0.07355	-0.05584	0.99195	4	3	1

-0.54869	-0.48264	-1.08315	4	1	0
0.06765	-0.66473	-0.73192	4	1	2
0.17777	-0.56151	-1.56538	5	1	2
1.28224	0.10586	1.46292	5	5	5
0.60391	-0.27949	0.92218	5	5	3
0.47778	-0.37408	0.03155	5	3	3
1.10359	-0.28756	1.11927	6	7	5
1.43123	-0.10319	0.07027	6	5	5
0.77266	-0.28437	0.08146	6	4	3
2.54536	0.48796	1.95135	8	8	8
1.74663	0.27797	-0.31074	8	7	6
2.93229	-0.27842	-1.07141	9	8	11
2.50374	-0.33844	-0.27325	10	8	10
0.84043	-0.2646	0.90815	10	6	4
6.45325	-0.48	-1.18791	28	28	27

Appendix C

Correlation Matrix

		DIT	MPC	RFC	LCOM	DAC	WMC	NOM	SIZE2	SIZE1
Correlation	DIT	1.000	.018	.108	.123	.392	-.134	.125	.203	.012
	MPC	.018	1.000	.331	-.102	.015	.136	-.114	-.083	.368
	RFC	.108	.331	1.000	.820	.638	.738	.812	.805	.800
	LCOM	.123	-.102	.820	1.000	.560	.574	.884	.835	.541
	DAC	.392	.015	.638	.560	1.000	.570	.809	.886	.639
	WMC	-.134	.136	.738	.574	.570	1.000	.702	.690	.894
	NOM	.125	-.114	.812	.884	.809	.702	1.000	.987	.695
	SIZE2	.203	-.083	.805	.835	.886	.690	.987	1.000	.709
	SIZE1	.012	.368	.800	.541	.639	.894	.695	.709	1.000

Appendix D

Correlation Matrix

		WMC1	DIT	NOC	CBO	RFC	IC	CBM	NOMA
Correlation	WMC1	1.000	-.091	-.041	-.044	.879	.417	.642	.468
	DIT	-.091	1.000	-.053	-.003	-.044	.361	.177	.325
	NOC	-.041	-.053	1.000	.503	-.075	.093	.003	-.145
	CBO	-.044	-.003	.503	1.000	-.073	.066	.060	.023
	RFC	.879	-.044	-.075	-.073	1.000	.507	.728	.505
	IC	.417	.361	.093	.066	.507	1.000	.759	.414
	CBM	.642	.177	.003	.060	.728	.759	1.000	.594
	NOMA	.468	.325	-.145	.023	.505	.414	.594	1.000

Appendix E

Regression Analysis Results

	Ques system	HMI system
R-square	0.42365	0.843596
r (correlation coefficient)	0.650884	0.918619
r- square	0.42364	0.843861
t values	7.1217203	22.65902
p values	<0.0001	<0.0001