Performance of Aspect-Oriented Software Quality Modelling using Artificial Neural Network Technique

Pankaj Kumar
CSE Department
MU, Chittorgarh

Sarvottam Dixit
Faculty of E & T
MU, Chittorgarh

S. K. Singh
CSE Department
GCET, Greater Noida

ABSTRACT
An exact estimation is the key focus of any prediction model. Software quality is one of the basic research issue for software organizations. Of late, different uses of soft computing techniques has been endeavored. This has been conceivable because of the accessibility of informational collections of expansive a large number of finished projects. Among various soft computing techniques, Artificial Neural Network (ANN) based models are outstanding, which to a great degree needs more research work and tries to find the most sensible model for software quality in term of accuracy, evolvability, extensibility, sustainability, design stability, and configurability associated with the suitability of model.

This paper attempts to explore the performance of aspect-oriented quality modelling using artificial neural network technique. Software quality relies upon a few performance elements. Further, these elements are related to each other and influence the software development process and quality directly or indirectly.

Keywords
AOP, Aspect, AOSQ, Quality Model, ANN

1. INTRODUCTION
Software has turned out to be larger and more complex, as demand for functionality has increased along with computer power. The ideal philosophy of any software is to deliver a quality product that meets the client's needs on-time and on-budget. However, delivering quality software is troublesome and just ends up harder as the application and the group taking a shot at it start to develop [2, 3].

Software quality has obtained its significance in different application regions, for example, e-government applications, mobile applications, web applications, software engineering, and business process advancement. Using software quality measurement, software suppliers upgrade their administration strategies and decision support activities. Quality identifies with human-software interactions when a software product is utilized as a part of a specific goal. As of now, different quality models are bound to incapable estimation plan and numerous quality models are subjectively muddled [4, 5].

Software organizations tackle this problem by leveraging artificial intelligence (AI), machine learning (ML), and soft computing (SC) to identify issues before production and focus testing efforts on the highest risk areas of the quality.

Two of the biggest issues that software leaders face is cost of change and software quality. Cost of change is the cost associated with fixing an issue over time. The second major issue facing software leaders is quality. Time, budget, and/or people are most always a limiting factor and unfortunately quality assurance is one of the areas that suffers the most when a release is approaching. What is needed is a way to better understand the areas of the most quality issues and predict the areas with the highest quality risk. With this knowledge, teams and managers can direct their limited resources optimally.

An explanation for these problems in software is lack of quality. If software quality is to be improved a means to evaluate quality is required. Unfortunately, there is no direct way to measure this, and although currently available software tools attempt to assist in this regard, the assistance is rather limited. This has motivated efforts to develop predictive models of aspect-oriented software quality [6, 7].

Artificial intelligence (AI), machine learning (ML), and soft computing (SC) techniques are applied to develop model that can describe these quality classes. Artificial Neural Network (ANN) is selected for evaluation which is a part of soft computing techniques.

The reminder of this paper is composed as section 2 portrays the background study. Section 3 talks about aspect-oriented software quality. Section 4 discusses about the artificial neural network. Section 5 contains the proposed modelling. Finally, Section 6 closes with the distinctive perceptions.

2. BACKGROUND STUDY
The overall background studies related to software quality with artificial neural networks are represented in tabular form which is shown in table 2.

3. ASPECT-ORIENTED SOFTWARE QUALITY
Over the last 50 or so years, languages have continued to evolve in order to support their ever increasing usage. Program design and maintenance became issues with the dawn of software engineering.

Today, the leading programming paradigm used is Object Oriented programming (OOP). [7, 8] The aim behind the development of OOP was to organize the data of an application and its associated methods into coherent entities. It encourages software reuse. When the complex program contains:

- Crosscutting concern
- Code Scattering

OOP is impossible to show the clear and fine programming Structure.

Enter aspect-oriented programming (AOP). AOP could be the next step in the steady evolution of the OO paradigm, or perhaps it will evolve into a completely new paradigm independent of OO. Whatever the case, AOP offers a solution to a design and maintenance problem that has plagued software developers for years. That is, how can we create modules with little or no crosscutting concerns? AOP introduces the notion of Aspects, and shows how we can take crosscutting concerns out of modules and place them in a

centralized place. While this paradigm is still relatively new, it seems promising and perhaps given time will replace OO.

Xerox PARC (Palo Alto Research Center) has worked on developing programming techniques that makes it possible to express those programs that OOP fail to support. This programming technique is called Aspect-Oriented Programming (AOP). The term Aspect-Oriented Programming includes Multidimensional Separation of Concerns, Subject-Oriented Programming, Adaptive Programming and Composition Filters.

Aspect-Oriented Programming (AOP) is a new programming paradigm developed at PARC. Aspect-Oriented Programming aims to achieve separation of concerns by improving code modularization using Aspects. Most of the time, aspects represent non-functional requirements. In AOP, components and aspects are separate codes and the weaving process is done by a static (compile time) or dynamic (run time) compiler that is called an aspect weaver.

Any new addition to the existing code may further worsen the situation, if the integration is not carried out carefully. Since the target application will have its behavior changed, it can cause an impact on software quality parameters like reliability, functionality, performance and efficiency. The application of AOP paradigm can simplify the up gradation, maintenance and evolvability of the software. However, the incorrect usage of AOP paradigm may not lead to the desired quality level of the software. Further, existence large number of process paradigms and existence varied product standards, the assessment of quality becomes pertinent.

Several quality models derived for the ISO/IEC 9126 Quality Model. In this paper, we have reviewed number of software quality models that derived from ISO/IEC 9126 Quality Model is known as tailored software quality model.

Table 1: Tailored Software Quality Model

Tailored Software Quality Model	Year
Bertoa Model	2002
GEQUAMO (Generic, Multilayered and Customizable Model)	2003
CapGemini Open Source Maturity Model	2003
ALVARO Model	2005
RAWASHDEH Model	2006
Open BRR Model	2006
Andreu Model	2007
ISO 25010 Model	2008
SQO-OSS Model	2008
AOSQUAMO Model	2009
QualOSS Model	2009
Alvaro Model	2010
REASQ	2010
Upadhyay Model	2011
Al-Badareen Model	2012
QUAMOCO Model	2012
AOSQ Model	2012
Midas Model	2013
PAOSQMO Model	2016

Table 1 shows the tailored software quality model. Most of the tailored software quality models assess with its applications.

4. ARTIFICIAL NEURAL NETWORKS (ANN)

Numerous advances have been made in developing intelligent system, some inspired by biological neural networks. Researchers from many scientific disciplines are designing Artificial Neural Networks (ANNs) to solve a variety of problems in pattern recognition, prediction, optimization, associative memory, and control system.

Artificial Neural Networks are relatively crude electronic models based on the neural structure of the brain. The brain basically learns from experience. It is natural proof that some problems that are beyond the scope of current computers are indeed solvable by small energy efficient packages. This brain modeling also promises a less technical way to develop machine solutions. This new approach to computing also provides a more graceful degradation during system overload than its more traditional counterparts [1].

These biologically inspired methods of computing are thought to be the next major advancement in the computing industry. Even simple animal brains are capable of functions that are currently impossible for computers. Computers do rote things well, like keeping ledgers or performing complex math. But computers have trouble recognizing even simple patterns much less generalizing those patterns of the past into actions of the future [1].

An artificial neural network is an information processing system that has certain performance characteristics in common with biological neural networks. The model of ANN is shown in figure 1.

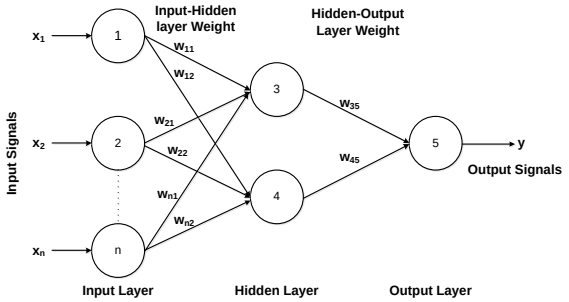

Figure 1: ANN Model

Artificial neural networks have been developed as generalizations of mathematical models of human cognition or neural biology, based on the assumption that:

a) Information processing occurs at many simple elements called neurons.

b) Signals are passes between neurons over connection links.

c) Each connection link has an associated weight, which, in a typical neural network, multiplies the signal transmitted.

d) Each neuron applies an activation function (usually nonlinear) to its net input (sum of weighted input signals) to determine its output signal.

Selection of artificial neural networks fall into the following five categories:

a) Prediction

b) Classification

c) Data Association

d) Data Conceptualization

e) Data Filtering

The computation in artificial neural network is done using following diagram shown in figure 2.

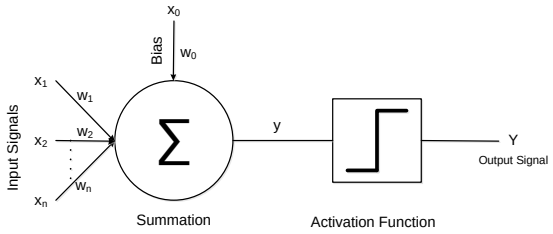

Figure 2: Computational Model of Artificial Neuron

Where, x_o is the bias and w_o is the corresponding weight. $x_1, x_2, x_3, \ldots\ldots, x_n$ are n input signals and $w_1, w_2, w_3, \ldots\ldots, w_n$ are corresponding weights.To calculate the final output, we use the following equation

$$y = x_0.w_0 + x_1.w_1 + x_2.w_2 + x_3.w_3 + \ \ldots\ldots\ldots\ldots + x_n.w_n$$

$$y = Bias + \sum_{i=1}^{n} x_i.w_i$$

or

$$y = x_0.w_0 + \sum_{i=1}^{n} x_i.w_i$$

Where, y is total inputted signals. To generate final output Y, we apply the activation function as $f(y)$.

$$Y = f(y) = f\left(Bias + \sum_{i=1}^{n} x_i.w_i\right)$$

5. PROPOSED MODELING

Different topologies of feedforward ANNs using the backpropagation learning algorithm were studied to approach the behavior of AOP project quality for various levels. The systematic modeling procedure proposed in this paper can be seen in figure 3.

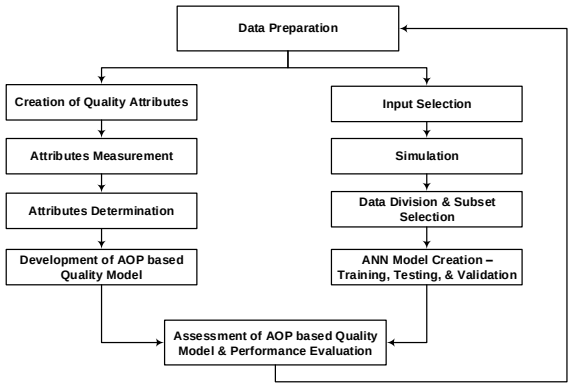

Figure 3: Proposed Methodology

The main steps for project modeling for quality ANN include: Data preparation, Input selection, Data division and selection of subsets, Model creation, and Performance evaluation. Another important step for the creation of ANN models for quality prediction. ANN uses historical data for prediction of parameters. While making ANN models, a few information might miss from the original database. Modelers for the most part supplant the missing information with the normal of the example or just erase or overlook the entire line, causing the loss of imperative information.

Following are the steps:

a) Data preparation for ANN models

b) Selection of inputs and outputs for the ANN model

c) Construction of the input database using Monte-Carlo method

d) Creation of the ANN model, division of models into subsets of parameters and performance evaluation

Table 1: Background Study

Year	Paper Title and Author Name	Summary
1997	Application of neural networks to software quality Modeling of a very large telecommunications system by T. M. Khoshgaftaar et.al.	Authors introduce the use of the neural networks as a tool for predicting software quality.
2004	Estimation of Software Defects Fix Effort Using Neural Network by Hui Zeng et.al.	Authors present a solution for estimating software defect fix effort using self-organizing neural networks.
2005	Empirical validation of object-oriented metrics on open source software for fault prediction by T. Gyimothy et.al.	Authors empirically validated Chidamber and Kemerer metrics on open source software for fault prediction. They employed regression (linear and logistic regression) and machine learning methods (neural network and decision tree) for model prediction.
2005	Evaluation of various training algorithms in a neural network model for software engineering applications	Authors propose to evaluate various training algorithms in a neural network model and shows

	by K. K. Aggarwal et.al.	which is the best suited for software engineering applications.
2008	Software Effort Estimation Using Soft Computing Techniques by P. S. Sandhu et.al.	The paper shows that soft computing technique– neuro-fuzzy can be applied for effort estimation by establishing its accuracy by comparing it with various other models. The estimation was done on NASA project data.
2008	Predicting Testing Effort using Artificial Neural Network by Yogesh Singh et.al.	The objective of this paper is to examine the application of ANN for software quality prediction using Object-Oriented (OO) metrics. Testing effort has been predicted using ANN method and independent variables are OO metrics given by Chidamber and Kemerer. The public domain NASA data has been used to find the relationship between OO metrics and testing effort.
2009	A Novel Neural Network Approach for Software Cost Estimation Using Functional Link Artificial Neural Network by B. T. Rao et.al.	A novel computationally efficient Functional Link Artificial Neural Network (FLANN) is proposed by authors to reduce the computational complexity so that the neural net becomes suitable for on-line applications.
2009	Application of Neural Networks in Software Engineering: A Review by Yogesh Singh et.al.	Authors discuss, how Neural Network (NN) can be used to build tools for software development and maintenance tasks.
2010	Test Effort Estimation Using Neural Network by Chintala Abhishek et.al.	The paper focuses on finding a method which gives a measure of the effort to be spent on the testing phase. This paper provides effort estimates during pre-coding and post-coding phases using neural network to predict more accurately also.
2011	Estimating Software Effort Based on Use Case Point Model Using Sugeno Fuzzy Inference System by Ali Bou Nassif et.al.	Authors propose a new regression model is created for software effort estimation based on use case point model. Furthermore, a Sugeno Fuzzy Inference System (FIS) approach is applied on this model to improve the estimation.
2015	Predictive accuracy comparison between neural networks and statistical regression for development effort of software projects by CuauhtémocLópez-Martín	Authors compare the neural network and statistical regression for development effort of software projects to get a better prediction of costs, schedule, and the risks of a software project, it is necessary to have a more accurate prediction of its development effort.
2016	Estimating Software Effort Using an ANN Model Based on Use Case Points by Ali Bou Nassif et.al.	Authors proposed a novel Artificial Neural Network (ANN) to predict software effort from use case diagrams based on the Use Case Point (UCP) model.
2016	Improving software quality using machine learning by K. Chandra et.al.	The objective of this paper is to equate and compare all of learning methods corresponding to performance parameter with its statistical method & methodology which would often results enhanced. Data points are the basis for prediction of models.
2016	A survey on machine learning techniques used for software quality prediction by S. Pattnaik et.al.	Authors conduct an extensive survey on various machine learning techniques like fuzzy logic, neural network, and Bayesian model, etc. used for software quality prediction along with an analytical justification for each of the proposed solutions.
2017	A Review of Improving Software Quality using Machine Learning Algorithms	Authors discuss a review of Improving Software Quality using Machine Learning Algorithms.

	by Jyoti Devi et.al.	
2018	Towards an intelligent fault prediction code editor to improve software quality using deep learning by Vasu Jindal	Author proposes a new intelligent Integrated Development Environment (IDE) which seamlessly allow programmers to test their code for faults using prior source code databases. The IDE is built upon deep learning models for making recommendations. The editor also gives scores to programmers on their program design. Author evaluate and validate our approach using famous NASA code repositories.
2018	Deep neural network based hybrid approach for software defect prediction using software metrics by C. Manjula et.al.	Authors introduce a hybrid approach by combining genetic algorithm (GA) for feature optimization with deep neural network (DNN) for classification. An improved version of GA is incorporated which includes a new technique for chromosome designing and fitness function computation. DNN technique is also improvised using adaptive auto-encoder which provides better representation of selected software features. The improved efficiency of the proposed hybrid approach due to deployment of optimization technique is demonstrated through case studies. An experimental study is carried out for software defect prediction by considering PROMISE dataset using MATLAB tool.

6. CONCLUSION

Relationships between static software measurements and quality components are often complex and nonlinear, constraining the precision of conventional methodologies. Artificial neural networks are adapt at modeling nonlinear practical connections that are hard to demonstrate with different strategies, and along these lines, are appealing for software quality displaying.

In this paper, we have abridged about the work done by the different authors and researchers, with the undertaking made to incorporate however many references as could reasonably be expected from year 1997 to 2018 and proposed an ANN model to audit the execution of AOP software quality.

7. REFERENCES

[1] Anil K. Jain, Jianchang Mao, and K. M. Mohiuddin, "Artificial Neural Networks: A Tutorial," in IEEE Computer Society, Volume: 29, Number: 3, Page No.:31-44, March 1996.

[2] Pankaj Kumar, S. K. Singh, and Sufia Nadeem Chishti "A Comprehensive Investigation of Quality of AOP-based Small-Scale Projects using Aspect-Oriented Software Quality (AOSQ) Model," in IEEE International Conference on Advances in Computing, Communication Control and Networking (ICAC3N'18), Greater Noida, (Uttar Pradesh), INDIA, Page No.: 63 - 67, October 2018.

[3] P. Kumar, and S. K. Singh, "Architectural Development of E-Learning Application using Aspect-Oriented Programming (AOP) Principles" International Journal of Computer Applications (IJCA), Foundation of computer science, New York, USA, Volume: 180, Issue: 2, Page No.: 21-25, December 2017.

[4] P. Kumar, and S. K. Singh, "A Framework for Assessing the Evolvability Characteristics along with Sub-characteristics in AOSQ Model Using Fuzzy Logic Tool," in IEEE 3rd International Conference on Computing, Communication and Automation (ICCCA-2017), Greater Noida, (Uttar Pradesh), INDIA, Page No.: 340 - 345, May 2017.

[5] P. Kumar, and S. K. Singh, "A Comprehensive Evaluation of Aspect-Oriented Software Quality (AOSQ) Model using Analytic Hierarchy Process (AHP) Technique," in IEEE 2nd International Conference on

[6] Advances in Computing, Communication & Automation (ICACCA-2016), Bareilly, (Uttar Pradesh), INDIA, Page No.: 1 - 7, September-October 2016.

[7] P. Kumar and S. K. Singh, "A Systematic Assessment of Aspect-Oriented Software Development (AOSD) using JHotDraw Application," in IEEE 2nd International Conference on Computing, Communication and Automation (ICCCA-2016), Greater Noida (Uttar Pradesh), INDIA, Page No.: 1 - 6, April 2016.

[8] P. Kumar "Aspect-Oriented Software Quality Model: The AOSQ Model," in Advanced Computing: An International Journal (ACIJ), Volume: 3, Number: 2, Page No.:105-118, March 2012.