International Journal of Innovative Research and Advanced Studies (IJIRAS)
Volume 4 Issue 10, October 2017

ISSN: 2394-4404

Adaptive Neuro-Fuzzy Approach To Web-Based Enterprise Software Evaluation

Mfreke Umoh

Department of Computer Science, Akwa Ibom State Polytechnic, Ikot Osurua, Nigeria

Chidiebere Ugwu

Department of Computer Science, University of Port Harcourt, Port Harcourt, Nigeria

Enoch Nwachukwu

Department of Computer Science, University of Port Harcourt, Port Harcourt, Nigeria

Abstract: ***This paper developed an intelligent web-based enterprise software evaluation system ($IWBES_2$) using a hybrid approach called adaptive neuro-fuzzy inference system (ANFIS) by combining fuzzy logic technique and neural network model. The neural network was designed using Tagaki Sugeno inference mechanism while Gaussian membership function (Gaussmf) was used to map the input parameters to the output parameter. The system was implemented in MATLAB$^{(R)}$ R2015a using a total of 682 dataset collected from CISCO workstation 3750 in Akwa Ibom State Transport Corporation (AKTC) data warehouse, Uyo. Back-propagation and hybrid learning methods were deployed in training the network comprising software quality attributes of functionality, reliability, usability, efficiency, maintainability, and portability after a pre-processing analysis by principal component analysis (PCA) for dimension reduction of the dataset. The performance evaluation of the system was carried out using mean square error (MSE) estimator. Results indicate training MSE values of 0.024642 and 0.047303 at 300 epochs for hybrid learning algorithm and back-propagation method, respectively. Results revealed that hybrid learning algorithm processes faster than back-propagation method in the evaluation of software quality attributes. The system performance in terms of software quality prediction using hybrid method was better than back-propagation with average error of 0.024642 and 0.047283, respectively further revealing that usability and portability software attributes had no much effect on software design while reliability, functionality, efficiency and maintainability influences the overall software quality performance most. Therefore, ANFIS evaluation of $IWBES_2$ with hybrid learning method performed better than back-propagation and is suitable for web-based software quality prediction.***

Keywords: ANFIS, PCA, software quality attributes, web-based software evaluation

I. INTRODUCTION

The primary goal of any evaluation is to check results of actions, in order to improve the quality of the actions or to choose the best action alternative. Evaluating software aids assessment of the various aspects of a system for a decision among several prototypes or for comparing several versions of a software system (Roberts and Morgan, 1983). Using a prepared list of criteria along with some practical experimentation, a software evaluation makes it possible to determine if the products would be helpful to the client or if some other combination of software products would serve to better advantage. Software can be evaluated with respect to different criteria or metrics such as functionality, reliability, usability, efficiency, maintainability, and portability. Software evaluation, therefore, is a task, which results in one or more reported outcomes (Suchman, 1967); and is dependent on the current knowledge of science, methodological standards applicable to software development, which plays very critical role within Human-Computer Interaction (Asuquo *et al.*,

2008). The standard provides a framework for organizations to define a quality model for a software product.

Figure 1 shows software quality model comprising six software quality metrics and each quality sub-characteristics (ISO 9241,). Functionality is a set of attributes that bear on the existence of a set of functions and their specified properties. The functions are those that satisfy stated or implied needs such as suitability, accuracy, interoperability, security, etc. while reliabilityis a set of attributes that bear on the capability of software to maintain its level of performance under stated conditions for a stated period of time. This implies compliance to fault tolerance and recoverability. On the other hand, usability of a product is the extent to which the product can be used by specific users to achieve specific goals with effectiveness, efficiency, and satisfaction in a specific context of use. The context of use is defined in terms of the user, the task, the equipment, and the environment. Ideally, usability of a software product implies understandability, learnability, operability, and attractiveness. Efficiency is a set of attributes that bear on the relationship between the level of performance of the software and the amount of resources used, under stated conditions, which implies time behaviour and resource utilization. Maintainability refers to a set of attributes that bear on the effort needed to make specified modifications for system's stability while portability is a set of attributes that bear on the ability of software to be transferred from one environment to another which implies adaptability, installability, and co-existence. Each quality sub-characteristic is further divided into attributes. As a result, the notion of user extends to operators as well as to programmers, which are users of components such as software libraries.

Software evaluation is usually performed at the end of the developing phase, using experimental designs and statistical analysis but can, however, be used as a tool for information gathering within iterative design. This situation has been improved in recent years in a number of ways (Whitefield, 1991).

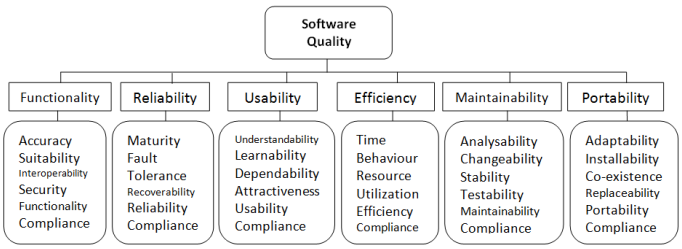


Figure 1: Software quality model (adopted: ISO/IEC 9126)

Most times the software bought by clients or organizations do not meet their needs despite the huge amount of resources and significant portion of organisations' capital budgets consumed. The problem of poor quality product and software failure has caused more than inconvenience especially in this era of ubiquitous computing whereby users asses the system anywhere anytime. Software errors have caused human fatalities now that most of the systems used at home, in the hospital, and industry are embedded systems. The causes have ranged from poorly designed user interfaces, specification misinterpretation, to direct programming errors. Due to the cost of maintaining faulty or unreliable systems, it is more cost effective to detect potential software quality problems earlier rather than later in software development. An application with good structural software quality costs less to maintain and is easier to understand and change in response to pressing business needs. Moreover, poor structural quality is strongly correlated with high-impact business disruptions due to corrupted data, application outages, security breaches, and performance problems.

The quality attribute of software products are often neglected at the developmental stage due to lack of knowledge perhaps from the domain expert. Most works evaluating web-based software products employed the use of multi-criteria decision making technique (Dubey *et al.*, 2012; Ioannis, 2013), which lack the capability to handle uncertainty and imprecision inherent in attributes used in software quality measurement. To avoid the problem of software ineffectiveness, this work develops an intelligent web-based software evaluation system using metrics such as functionality, reliability, usability, efficiency, portability and maintainability. The paper presents a hybridized approach by designing an adaptive Neuro-fuzzy inference system (ANFIS) for evaluating and selecting the most appropriate software among alternatives. Consequently, the parameters of the fuzzy system are tuned by neural networks to efficiently evaluate software metrics in order to ascertain the level of software quality of clients or organizations.

The rest of the paper is structured as follows. Section 2 presents a critical review of related works while section 3 presents the design of the adaptive Neuro-fuzzy approach to software evaluation. In section 4, the results obtained from MATLAB implementation of the intelligent web-based enterprise software evaluation system ($IWBES_2$) are presented. Finally, section 5 gives the concluding remarks.

II. RELATED WORKS

Software testing can be conducted as soon as executable software exists. The overall approach to software development often determines when and how testing is conducted. For example, in a phased process, most testing occurs after system requirements have been defined and then implemented in testable programs. In contrast, under an Agile approach, requirements, programming, and testing are often done concurrently. Testing cannot establish that a product functions properly under all conditions but can only establish that it does not function properly under specific conditions (Cem, 1999). The scope of software testing often includes examination of code as well as execution of that code in various environments and conditions as well as examining the aspects of code: does it do what it is supposed to do and do what it needs to do. In the current culture of software development, a testing organization may be separate from the development team.

Software testing can provide objective, independent information about the quality of software and risk of its failure to users and/or sponsors (Cem, 2006).Information derived from such test may be used to correct the process by which software is developed (Kolawa, 2007). However, the primary purpose of testing is to detect software failures so that defects may be discovered and corrected. Software faults typically occur when the programmer makes an error (mistake) through misinterpreted specifications, which results in a defect (fault,

bug) in the software source code. If this defect is executed, in certain situations, the system will produce wrong results, causing a failure, though not all defects will necessarily result in failures when the environment is changed (e.g. different hardware platform or interaction with different software).

Artificial intelligence (AI) methods have been widely applied in students classification and performance modeling, medical disease diagnosis, stock market forecasting, electric load prediction, supply chain management, network traffic control, image processing and feature extraction, etc. with capabilities to handle linguistic uncertainties by modeling vagueness and unreliability of information. AI methods mainly comprise fuzzy logic, neural networks, genetic programming, and hybrid approaches such as Neuro-fuzzy systems, genetic fuzzy systems, and genetic programming neural networks, etc. Recently, many authors have shifted to Neuro-fuzzy domains for much improved results and a huge number of related applications are being developed (Taylan *et al.* 2009; Sindal *et al*. 2009; Iraji *et al.* 2012; Khamenah *et al.* 2012; Obi and Imainvan 2011; Abbas *et al.* 2011; Kablan 2011; Giovans, 2012; Fang 2012; Gomathi *et al.* 2010; Jafari *et al.* 2011, Sood and Aggarwal, 2011; Sudheer and Mathur 2012). Neuro-fuzzy systems (NFS) refer to a combination of artificial neural networks and fuzzy logic (Jang, 1993). The basic idea behind this NFS is that it combines human-like reasoning style of fuzzy systems with the learning and connectionist structure of neural networks. NFS provides powerful and flexible universal approximations with the ability to explore interpretable IF-THEN rules. The use of NFS is proliferating into many sectors in our social and technological life and software evaluation cannot be an exemption.

Sevarac (2006) presented a Neuro-fuzzy system for student modeling. The proposed system performed classification of students based on qualitative observations of their characteristics. Taylan *et al*. (2009) proposed adaptive Neuro fuzzy inference system (ANFIS) using genetic algorithm (GA) to assess student's academic performance. Obi and Imainvan (2011) analyzed Alzheimer disease diagnosis using Neuro fuzzy inference procedure. Agboizebeta and Chukwuyeni (2012) have focused on thyroid disorder with the help of Neuro fuzzy expert system using a set of symptoms. The system designed was an interactive system that was able to interact with the patient briefing his current condition. Khameneh *et al.* (2012) has performed investigation using ANFIS to detect abnormality in red blood cell and classify blood samples into normal and abnormal category. Their motivation behind the research work was to identify, classify, diagnosis different types of disease for different fields. Sindal *et al.* (2009) developed Neuro fuzzy call admission control algorithm to reduce voice data traffic in CDMA cellular network by transmitting signals at lower power level. Abbas *et al.* (2011) proposed Neuro fuzzy system for best route selection to avoid traffic congestion. They used Neuro fuzzy logic and ant colony system (ACS) algorithm for routing. Makhsoos *et al.* (2009) worked on face recognition using multi-layer perceptron (MLP) network by combining fuzzy logic, neural network and mixture of experts. They presented the importance of combining these two technologies (neural networks and fuzzy logic) in face recognition. Gomathi *et al.* (2010) proposed neuro fuzzy approach for facial expression recognition system to recognize the human facial expressions like happy, fear, sad, angry, disgust and surprise using local binary pattern (LBP) histogram. This model reported 95.29% of classification accuracy.

Dubey *et al.* (2012) proposed a methodology for quantifying the usability of software using a fuzzy multi-criteria weighted average approach. Fuzzy logic helped to deal with the uncertainty and imprecision of the importance and rating of attributes on which usability depends. This approach was chosen due to the highly unpredictable nature of the attributes on which usability depends. Ioannis (2013) presented an integrated solution through which significant improvement may be achieved, based on Multiple Criteria Decision Aid (MCDA) methodology and the exploitation of packaged software evaluation expertise in the form of an intelligent system. The MCDA methodology consists of different methods categorized into three classes, viz; multiple attribute utility method, outranking method and interactive method. Ioannis (2013) only addressed generalized issues of software evaluation that can be applied to diverse end user domains but was not specified to web-based software which is the main interest of this research. Alexiei (2014) proposed an Intelligent Usability Evaluation (IUE) tool to automate the usability evaluation process. Two set of tools - those predicting the usage of websites such as Cognitive Walkthrough for the Web (CWW) and those making use of conformance to standards such as Usability Evaluation framework (USEFUL) were used. These tools evaluated the usability of a website by employing the heuristic evaluation technique which references the set of research-based usability guidelines. However, there was no integration of the intelligent tool directly within website development environment. In this way, usability violations could surface in real time as the website is being created. This non intelligent approach involves the use of real persons; like the focused group and cognitive walkthrough. Farooq (2013), proposed a set of guidelines which define a protocol to carry out comparative study of software testing techniques. Certain factors were considered necessary for comparison during such test. These include number of faults, fault rate, fault type, size (test case generated), coverage, time (i.e. execution time), software/program type, experience of subjects, reliability improvement. The fact that a single technique cannot give the require result informs the selection of a combination of appropriate testing techniques to mitigate targeted software development faults.

Marza *et al.* (2008) developed and evaluated a Neuro-fuzzy model to estimate software projects development time which is one of the challenging tasks for software developers. The authors used MATLAB 7.4 to process the fuzzy logic, neural network and Neuro-fuzzy systems. Khyati *et al.* (2013) proposed ANFIS-based software effort evaluation, which combines best features of fuzzy logic and parallel processing neural networks. It possesses fast convergence and has more accuracy than back-propagation neural network. Pergola (2013) considered web-based learning systems as a supplement to their Textbooks and found out that there are critical differences across publishers with respect to the system's interface, functions, content, features, and support.

These differences directly impact the effectiveness of a web-based learning system as an instructional tool and as a corollary, which, perhaps may impact the ultimate utility of the associated textbook as a pedagogical resource. As such, an evaluation of available web-based learning systems became an essential component of the textbook review process, comprising a meticulous evaluation of the system's functionality and features in light of instructor and student needs and preferences. This guidance was presented in the form of a framework based on key processes underlying the systems development life cycle (SDLC). The SDLC phases most relevant to evaluating web-based learning systems include systems survey, system analysis, and systems selection processes. Zulkefi *et al.* (2012) made used of Software Usability Measurement Inventory (SUMI) to evaluate a tool for cost estimate called WebCost. WebCost was developed using Java and Eclipse editor as a standalone application. SUMI was selected to serve as an evaluation tool to measure effectiveness in terms of interface and provide precise results. It became a solution to the recurring problem of measuring users' perception/satisfaction of the usability of software by providing a valid and reliable method for the comparison of products and different versions of the same products, as well as providing diagnostic information for future developments. This usability instrument composed of a validated 50-item paper-based questionnaire in which respondents score each item on a three-point scale of agreed, undecided and disagree. The questionnaire was designed to measure the effects, efficiencies, simplicity, helpfulness and control of a product. This approach was time consuming, involving users both as evaluator and participants which may not yield accurate result based on certain factors like diverse user roles and preferences.

Chai-Lee (2012) carried out web site evaluation focusing on Web site attributes, organization and technology, enumerating the most common website criteria applicable as quality, functionality, credibility, reliability, attractiveness, systematic structure and navigation. For Mike (2011), a criteria-based quantitative assessment of the software in terms of sustainability, maintainability, and usability can inform high-level decisions on specific areas for software improvement. The assessment involves checking whether the software, and the project that develops it, conforms to various characteristics or exhibits various qualities that are expected of sustainable software. The more characteristics that are satisfied the more sustainable the software. He pointed out that not all qualities have equal weight. In performing the evaluation, one may want to consider how different user classes affect the importance of the criteria. For example, for usability, understandability - a small set of well-defined, accurate, task-oriented user documentation may be comprehensive for users but inadequate for developers. Arockiam, (2010) suggested a service oriented architecture (SOA) reliability evaluation model using two attributes: availability - which is the quality attribute of whether the web service is present or ready for immediate use, and accessibility - which is the quality attribute of service that represents the capability of serving a web service request. Zhou (2009) proposed a website quality metrics and methods to measure the website interface (aesthetic) and reputation quality factors. The study built a website evaluation tool with four layers, viz; Tree-Traversal, Parsing, Data Metrics, and Graphical User Interface to measure website quality automatically. Certain specific technologies such as Data Crawler, Traversal, Recursive Algorithm, Data Analysis and Transmission were used in the program design. Zhou's framework only highlighted the web quality hierarchy and lacked other major components of its kind. Ivory (2001) explored the development of an automated Web evaluation methodology and tools where an extensive survey of usability evaluation methods for Web and graphical interfaces was presented. The work presented a new methodology for HCI: a synthesis of usability and performance evaluation techniques, which together build an empirical foundation for automated interface evaluation. The general approach involves: 1) identifying an exhaustive set of quantitative interface measures; 2) computing measures for a large sample of rated interfaces; 3) deriving statistical models from the measures and ratings; 4) using the models to predict ratings for new interfaces; and 5) validating model predictions.

This work presents an intelligent framework for evaluated web-based software products by combining two promising AI techniques – neural networks and fuzzy logic. The hybrid model has the ability to train the dataset, enable linguistic representation of model's inputs and output to tolerate imprecision and is suitable for quality evaluation as many software attributes are measured on nominal or cardinal scale type which is particular case of linguistics values. The learning capability of neural networks facilitates flexibility in problem modeling with embedded knowledge to provide support for attribute quantification. There is also the advantage of speedy computation and improve performance evaluation through the elimination of construction errors thereby increasing the quality of enterprise software.

III. DESIGN OF THE INTELLIGENT WEB-BASED ENTERPRISE SOFTWARE EVALUATION SYSTEM

A. FRAMEWORK OF THE AUTOMATED WEB-BASED SOFTWARE EVALUATION SYSTEM

Figure 2 shows the designed framework of the intelligent web-based enterprise software evaluation system ($IWBES_2$). It integrates the different components of the entire system and comprises the intelligent front end (user interface) and the kernel (processing logic). The intelligent front end has tools that support the collection of necessary information to guide the evaluator in the correct application of the chosen methodology and provides expertise on software attribute evaluation. Expert assistant provided by the expert module helps the human evaluator assign values to attributes of the software evaluation model and manage past evaluation results, in order to be reused in new evaluation problem. The kernel consists of two major sub-components – the *knowledge base* containing the database model, quality model base, fuzzy logic model, neuro-fuzzy model, and - the *decision support engine* containing ANFIS and the emotional and cognitive fillers along with a self-training facility. Following is a description of some of these components.

ISSN: 2394-4404

DATABASE MODEL

The database model is made up of classes with objects and associated attributes. The database is designed in MySQL.

QUALITY MODEL BASE

The quality model base in figure 1 with a three-level structure comprising quality characteristics, quality sub-characteristics and measurable criteria (indicators) is adopted. In the first level, the software evaluation model presents six quality characteristics which include functionality, reliability, usability, efficiency, maintainability, portability. The second level characteristic is broken down by several sub-characteristics. Each sub-characteristic is inherited from parental quality characteristics. The third level is the measurable indicators. Finally, the values of the metrics are evaluated using a mathematical model.

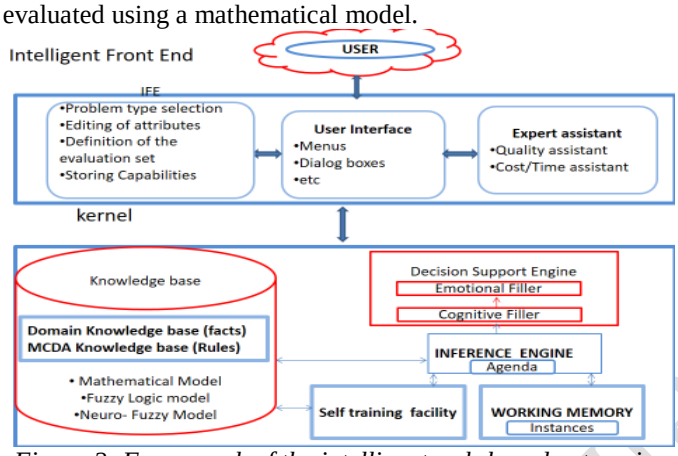


Figure 2: Framework of the intelligent web-based enterprise software evaluation system

FUZZY LOGIC MODEL

The fuzzy logic model comprises a knowledge base and a processing stage. Numerical crisp variables are the input of the system in the processing stage. These variables are passed through a fuzzification unit where they are transformed into linguistic variables and a mapping of attributes to MF is performed to determine their degree of membership. The result becomes the fuzzy input of the inference engine. This fuzzy input is transformed by rules of the inference engine to fuzzy output. These linguistic results are then changed by a defuzzification unit into numerical values that becomes the output of the system.The block diagram of the fuzzy logic model is as shown in Figure 3 with fuzzification, inference engine, rulebase, and defuzzification as components. A total of 15,625 fuzzy rules were obtained from six linguistic variables of functionality, reliability, usability, efficiency, maintainability, and portability from triangular membership function but reduces to 13 rules with gauss2 membership function and sub clustering. The linguistic terms used for each of the variables are very high (VH), high (H), moderate (M), low (L), very low (VL). The Takagi-Sugeno inference mechanism was adopted for rules formulation.

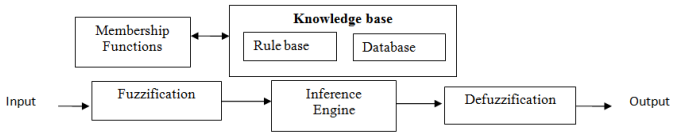


Figure 3: Fuzzy logic model

Two common techniques of defuzzification are the CENTROID and MAXIMUM methods. However, the CENTROID method was employed in this work, where the crisp value of the output variable is obtained by finding the variable value of the center of gravity of the MF for the fuzzy value. Gaussian MF was adopted due to its ability to handle high dimension of linguistics terms and data clustering. Gaussian curve MF depends on two parameters: the cluster center (C), which shows the results of clustering algorithms in the form of matrix, and sigma (σ) which is used to determine the value of Gaussian MF parameters. The general form of a Gaussian MF is shown in equation (1).

$$\square(x, \sigma, c) = \square^{-}\frac{(x-c)}{2\sigma^{2}} \quad (1)$$

Where, x is the variable, σ is the sigmoid function, c is the cluster center.

The linguistic values for $IWBES_2$ variables and the MF for each variable are as shown in equations2 - 7, respectively. The MF graph for all input and output parameters are shown in figures 4 and 5, respectively.

$$\text{Var}(x)\begin{cases} VL & \text{if } x < 0.2 \\ L & \text{if } 0.2 \geq x \leq 0.4 \\ M & \text{if } 0.4 \geq x \leq 0.6 \\ H & \text{if } 0.6 \geq x \leq 0.8 \\ VH & \text{if } 0.8 \geq x \leq 1.0 \end{cases} \quad (2)$$

$$VL(x)\begin{cases} \text{if } x < 0.2 \\ \square^{-}\frac{(x-c)}{2\sigma^{2}} & \text{if } 0 \leq x \leq 0.2 \\ & \text{if } x \geq 0.2 \end{cases} \quad (3)$$

$$L(x)\begin{cases} \text{if } x < 0.2 \\ \square^{-}\frac{(x-c)}{2\sigma^{2}} & \text{if } 0.2 \leq x \leq 0.4 \\ & \text{if } x \geq 0.4 \end{cases} \quad (4)$$

$$M(x)\begin{cases} \text{if } x < 0.4 \\ \square^{-}\frac{(x-c)}{2\sigma^{2}} & \text{if } 0.4 \leq x \leq 0.6 \\ & \text{if } x \geq 0.6 \end{cases} \quad (5)$$

$$H(x)\begin{cases} \text{if } x < 0.6 \\ \square^{-}\frac{(x-c)}{2\sigma^{2}} & \text{if } 0.6 \leq x \leq 0.8 \\ & \text{if } x \geq 0.8 \end{cases} \quad (6)$$

$$VH(x)\begin{cases} \text{if } x < 0.8 \\ \square^{-}\frac{(x-c)}{2\sigma^{2}} & \text{if } 0.8 \leq x \leq 1.0 \\ & \text{if } x \geq 1.0 \end{cases} \quad (7)$$

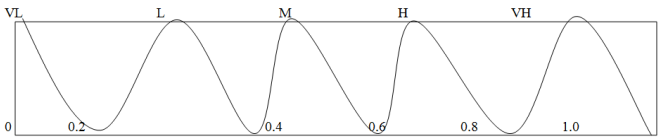


Figure 4: Input MF

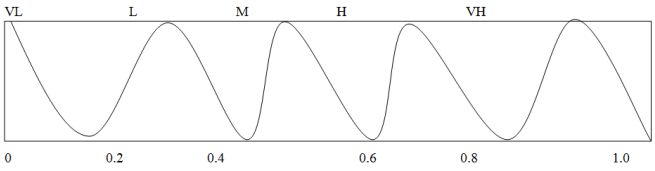


Figure 5: Output MF

The general form of the fuzzy rules is shown in equation (8) as:

If x is A and y is B, then f = px + qy +r (8)

where, x and y are input variables; A and B are fuzzy sets of the input variables; f is the output variable; p, q, and r are consequent parameters. Some of the rules are as follows:

- ✓ IF (Functionality is VH) and (Reliability is VH) and (Usability is VH) and (Efficiency is VH) and (Maintainability is VH) and (Portability is VH) THEN (Quality is out1cluster1) (1)
- ✓ If (Functionality is H) and (Reliability is H) and (Usability is H) and (Efficiency is H) and (Maintainability is H) and (Portability is H) then (Quality is out1cluster2) (1)

The fuzzy rules output are then mapped to a crisp point during defuzzification using the formula in equation (9).

$$\square (X, \square, C) = \square^{-} \frac{(X-C)^2}{2\square^2} \quad (9)$$

where, x is the variable, $\square$ is the sigmoid function, c is the peak of the bell

NEURO-FUZZY MODEL

A NFS is a fuzzy system that uses a learning algorithm inspired by neural network theory to determine its parameters (fuzzy sets and fuzzy rules) for processing data samples. The adaptive Neuro-fuzzy inference system (ANFIS) designed from the Sugeno FIS structure is shown in Figure 6 as a multilayer feed-forward framework with five layers (Takagi and Sugeno, 2015). The Sugeno FIS is useful for modelling nonlinear systems by interpolating between multiple linear models. ANFIS comprises two components, viz FIS and ANN, thus exhibit the capability to capture the benefits of both techniques in one framework. Its inference system corresponds to a set of IF-THEN rules with learning capability to approximate nonlinear functions. ANFIS constructs fuzzy MF parameters (adjusted by a hybrid learning algorithm) to approximate precisely, the model parameters. The MFs are useful in causing interaction of the input parameters during the training phase, until an optimal performance is achieved. The hybrid algorithm of ANFIS combines the gradient descent and least square methods. Gradient descent-based approaches and back-propagation techniques are some important learning algorithms in feed forward neural networks. The supervised learning stage was actualized by selecting fuzzy input parameters to build the fuzzy rules.

The nodes in layer 1 represent the fuzzy input linguistic variables which converts input values to the next level while those in layer 2 represent the input MFs which perform fuzzification of the input linguistic variables. The nodes in layer 3 represent the fuzzy rules. This layer is where the max-min operations are performed to determine the firing strength of the associated rules and overall system output. The nodes in layer 4 represent the output MFs that handle the weight assigned directly by an expert or from the historical data while layer 5 handles defuzzification process for evaluating the entire system output.

Figure 7 shows the sub-clustering algorithm for FIS generation and ANFIS optimization. Subtractive clustering assumes that each data point is a potential cluster center. The algorithm does the following:

STEP 1: Calculate the likelihood that each data point would define a cluster center, based on the density of surrounding data points.

STEP 2: Choose the data point with the highest potential to be the first cluster center.

STEP 3: Remove all data points near the first cluster center. The vicinity is determined using cluster Influence Range.

STEP 4: Choose the remaining point with the highest potential as the next cluster center.

Step 5: Repeat steps 3 and 4 until all the data is within the influence range of a cluster center.

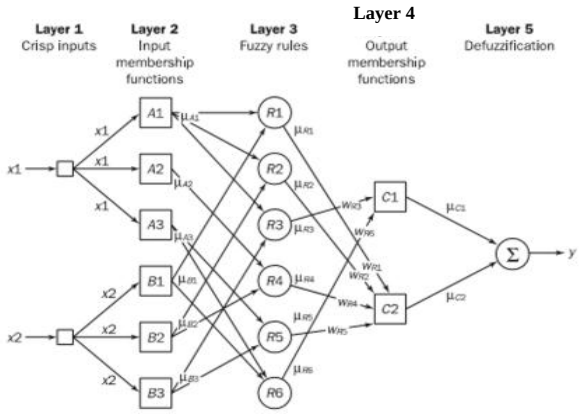


Figure 6: ANFIS architecture

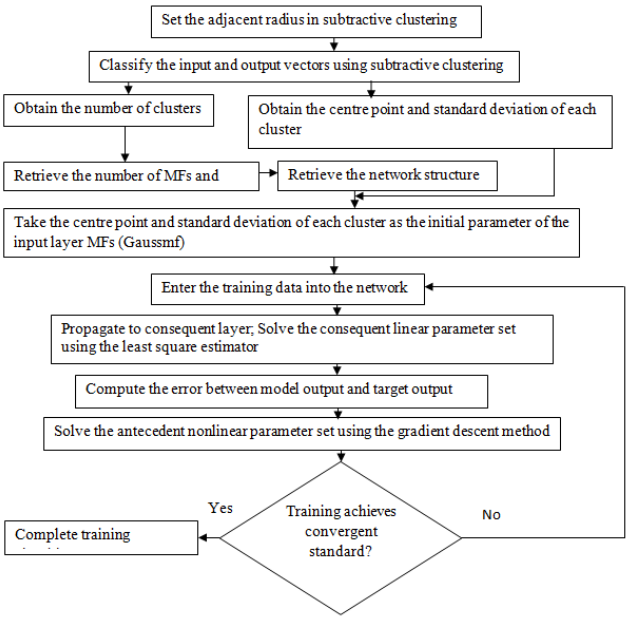


Figure 7: Sub-clustering for FIS generation and ANFIS optimization

IV. DISCUSSION OF RESULTS

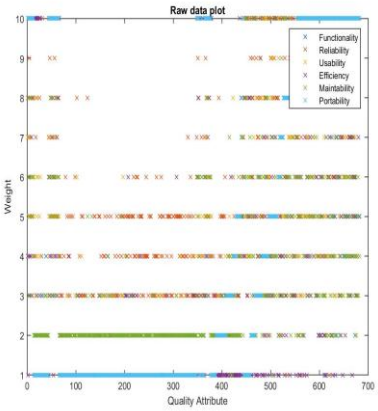
A. DATA DESCRIPTION AND IMPLEMENTATION PROCEDURE

The automated web-based software evaluation system was implemented on windows 10 platform, using PHP 5.6.25 from WAMP server 3.0.6 as the front-end engine, MySQL

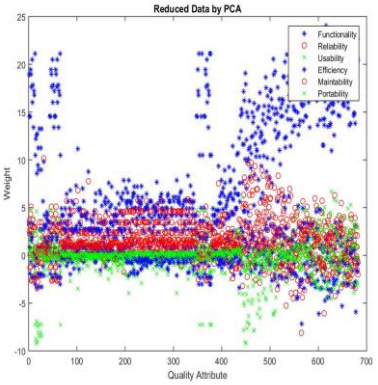
ISSN: 2394-4404

database 5.7.14 from WAMP server 3.0.6 and fuzzy logic tool box of MATLAB 2015 as the back-end engine. A total of 682 dataset was collected from AKTC Head Office, Uyo, Nigeria. The dataset covered data from CISCO WS-3750 Network Operating System, Transport Manager Software and Miktrotik Cloud router interface to Tally ERP9. The data obtained advanced in the following major stages, viz; data collection and pre-processing, FIS implementation and ANFIS implementation.

The dataset consist of 6 attributes which are metrics used as input to the fuzzy logic model. The attributes appear in their weights, totalling a maximum of 10 as shown in figure 8(a). The pre-processing implementation was carried out using principal component analysis (PCA) to reduce the dimensions of the dataset which consists of a large number of interrelated variables and obtain the most prominent parameters containing useful information in terms of rows of data, thereby deleting some missing attributes. The processed data is shown in figure 8(b). For instance, rows 1-147and 444-526 were combined to form training data, rows 296-443 and 610-682 were combined to form checking data while rows 148-295 and 527-609 were combined to form testing data. To reveal whether there is need to re-organize the data or choose another data for FIS, which forms input for ANFIS, the correlation between training and checking data was performed and the result is presented in figure 9. The checking data appears in the plot as pluses superimposed on the training data. The horizontal axis is marked dataset index. This index indicates the row from which input data value was obtained (whether or not the input is a vector or a scalar). Figure 10(a) and (b) show the FIS rule generation and rule viewer indicating six input parameters and one output, respectively.



(a) Raw data



(b) Pre-processed data by PCA

Figure 8: Quality attributes from dataset

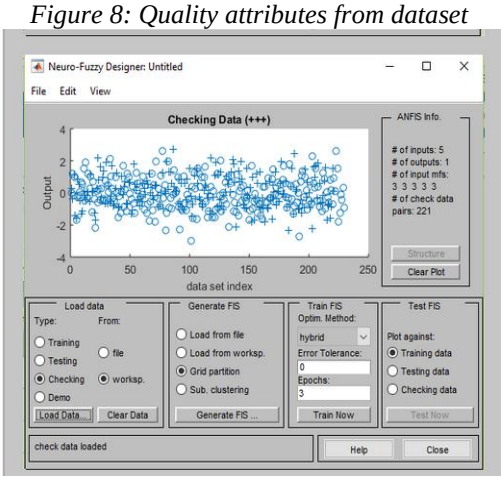
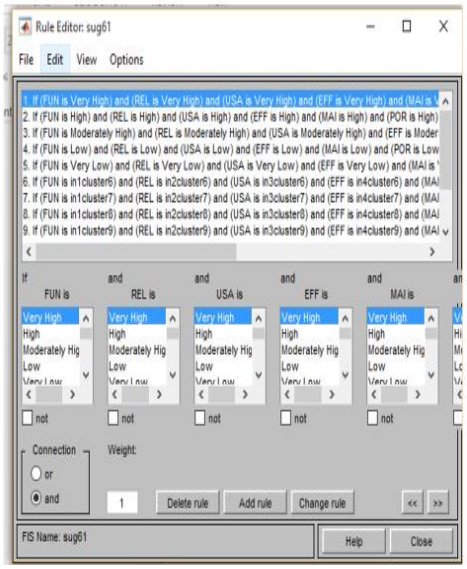
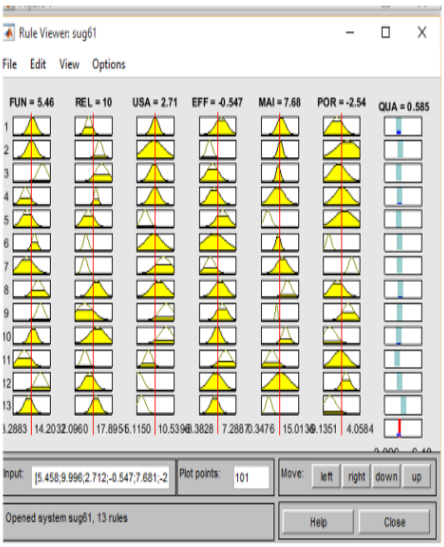


Figure 9: Plot of checking and training data



(a) Rule editor

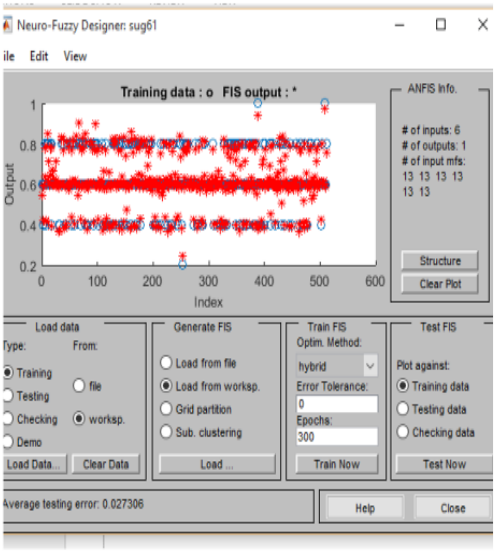


(b) Rule viewer

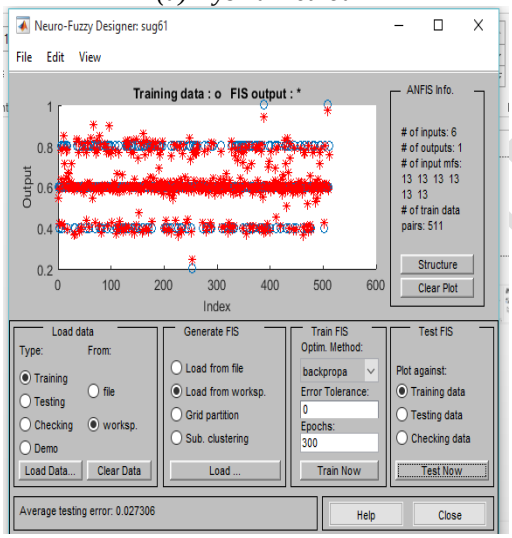
Figure10: FIS window

The training, checking and testing data, loaded either from file or workspace, help generate FIS to train and test FIS

as well as perform ANFIS training and testing. The output information shows six input and one output parameters with 511 trained data pairs. The optimisation method employed back propagation or hybrid algorithm. Figure 11(a-b) shows the ANFIS training output having thirteen rules with average error value of 0.027306 at 300 epochs. The training data are in blue colour while the FIS output are in red colour.



(a) Hybrid method



(b) Back propagation algorithm

Figure 11: ANFIS training output

B. ANFIS EVALUATION RESULTS

The system performance was evaluated comparing hybrid and back-propagation methods of ANFIS. The performance evaluation model adopted was the mean square error (MSE). MSE measures the differences between the values predicted by a model and the values actually observed. Results were evaluated to determine how well the network (model) output fits the desired output, using the MSE performance criterion expressed in equation (10) as:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i^* - y_i)^2 \quad (10)$$

where, y_i^* is the desired output, y_i is the network output, and N is the number of data items.

Table 1 summarizes the ANFIS training information deployed. Furthermore, results from tables 2 and 3 revealed that the ANFIS model is significant in optimizing software quality prediction, as the MSE values of training and test data gave close results for both hybrid and back-propagation methods. However, results indicate that the system performance in terms of software quality prediction using hybrid method was better than back-propagation method since the former yielded lower MSE values than the later at the same number of epochs. Furthermore, results showed that hybrid learning algorithm processes faster than the back-propagation algorithm in the evaluation of software quality attributes. The average error of 0.047283 was observed at epoch 300 in the back-propagation learning algorithm while an average error of 0.024642 was observed at the same training epoch in hybrid learning algorithm. The subtractive clustering algorithm was able to reduce the dimension of the fuzzy rules from 15625 to 13 rules, thereby reducing computational complexity. Results indicate that at epoch 300, the testing error values of 0.024642 and 0.047293 were observed between the computed data and the desired output for hybrid method and back-propagation algorithm, respectively. Over-fitting was resolved by testing the FIS trained data against the checking data, and choosing the MF parameters to be those associated with the minimum checking error if these errors indicate model over-fitting. The result generated from the FIS training shows that usability and portability are software attributes with less hierarchy and so these attributes had no much effect on software design. Software performance according to the data shows that reliability, functionality and maintainability influences the overall software quality most.

S/N	Parameter	Sub-clustering method
1	Number of nodes	191
2	Linear parameters	91
3	Non-linear parameters	156
4	Total number of parameters	247
5	Number of training data pairs	511
6.	Number of checking data pairs	511
7	Total number of fuzzy rules	13
8	Training MSE	0.027311
9	Validation MSE	0.052537
10	Testing MSE	0.052537

Table 1: ANFIS training information

S/N	Number of epochs	Training error	Checking error	Testing error	Average error
1	50	0.027065	0.02665	0.026852	0.02706
2	100	0.026347	0.026073	0.025819	0.026344
3	150	0.025506	0.025051	0.025257	0.02505
4	200	0.024816	0.024642	0.024642	0.024642
5	250	0.024642	0.024642	0.024642	0.024642
6	300	0.024642	0.024642	0.024642	0.024642

Table 2: ANFIS performance with hybrid algorithm

S/N	Number of epochs	Training error	Checking error	Testing error	Average error
1	50	0.047429	0.047421	0.047425	0.047421
2	100	0.047413	0.047399	0.047406	0.047399
3	150	0.04739	0.047373	0.047381	0.047373
4	200	0.047363	0.047343	0.047354	0.047345
5	250	0.047334	0.047314	0.047324	0.047314
6	300	0.047303	0.047283	0.047293	0.047283

Table 3: ANFIS Performance with back-propagation algorithm

V. CONCLUDING REMARKS

This paper shows how ANFIS model can be used to significantly optimize the prediction of quality attributes in software. The collected dataset was subjected to PCA for the purpose of dimension reduction and normalized by computing the covariance between the variables. Findings revealed that system performance in terms of software quality prediction using hybrid method resulted in lower MSE values, faster processing speed, and was better than back-propagation method for all considered epochs. Results revealed that usability and portability software attributes had no much effect on software design while reliability, functionality, efficiency and maintainability influences the overall software quality performance most. Hence, users of software systems should place importance to software quality attributes based on how relevant the attributes are for the success of the organization. Software developers ought to employ test driven approaches during software development in order to minimize error. Future works shall consider the use of extreme learning machine to improve learning rate and proffer solution to problems of stopping criteria, number of epochs and local minima inherent in conventional gradient descent algorithms.

REFERENCES

[1] Abbas, S., M.S. Khan, K. Ahmed, M. Abdullah, U. Farooq (2011). Bio-inspired neuro-fuzzy Based dynamic route selection to avoid traffic congestion, International Journal of Scientific & Engineering Research 2(6).
[2] Agboizebeta, I. A., O.J. Chukwuyeni (2012). Application of neuro-fuzzy expert systemfor the Probe and prognosis of thyroid disorder, International Journal of Fuzzy Logic Systems, 2(2).
[3] Agboizebeta, I. A., O.J. Chukwuyeni (2012). Cognitive neuro-fuzzy expert system for Hypertension Control, Computer Engineering and Intelligent Systems 3 (6), 21–32.
[4] Alexiei Dingli, Sarah Cassar (2014). An Intelligent Framework for Website Usability-Advancesin Human Computer Interaction.
[5] Asuquo, D. E., Williams, E. E., Oluwade, B. A., and Bassey, P. C. (2008). Educational Websites Usability Evaluation, Journal of Engineering and Applied Sciences, 3(7), 574-582.
[6] Cem Kaner, Falk, Jack; Nguyen, Hung Quoc (1999). Testing Computer Software, 2nd Ed. New York: John Wiley and Sons, Inc
[7] Cem, Kaner (2006). Exploratory Testing. Florida Institute of Technology, Quality Assurance Institute Worldwide Annual Software Testing Conference, Orlando, and FL. Retrieved November 22, 2014.
[8] Chai-Lee Goi (2012). A Review of Web Evaluation Criteria for E-Commerce Web Sites, International Journal of Digital Information and Wireless Communications 2(2): 197-201
[9] Dubey Sanjay Kumar, Anubha Gulati, Ajay Rana, (2012). Usability evaluation of software System using fuzzy Multi criteria approach. International Journal of Computer Science Issues, 9(3),
[10] Fang, H. (2012). Adaptive neurofuzzy inference system in the application of thefinancial Crisis Forecast, International Journal of Innovation, Management and Technology, 3(3), 250–254.
[11] Farooq, U., M.S. Khan, K. Ahmed, M.A. Saeed, S. Abbas, (2011). Autonomous system Controller forvehicles using neuro-fuzzy, International Journal of Scientific & Engineering Research 2 (6) (2011).
[12] Giovanis, E. (2012). Study of discrete choice models and adaptive neuro-fuzzy inference System in the prediction of economic crisis periods in USA, EconomicAnalysis & Policy 42 (1), 79–95.
[13] Gomathi, V., K. Ramar, A.S. Jeevakumar (2009). A neuro fuzzy approach for facial expression recognition using LBP histograms, International Journal of Computer Theory and Engineering 2(2), 1793–8201.
[14] Ioannis Stameles (2013). Knowledge Based Evaluation of Software System, International Journal of Digital Information and Wireless Communications, 2(2): 197-201
[15] Iraji, Md. S., M. Aboutalebi, N.R. Seyedaghaee, A. Tosinia (2012). Students' classification with adaptive neuro fuzzy, International Journal of Modern Education andComputer Science 7, 42–49.
[16] ISO/IEC (1991). ISO/IEC 9126. Information technology - Software product evaluation –Quality characteristics and guidance for their use.
[17] Ivory, Melody Yvette (2001). An Empirical Foundation for Automated Web Interface Evaluation
[18] Jackson, M., Crouch, S. and Baxter, R. (2011). Software Evaluation: Criteria-based Assessment
[19] Jafari, A., A.K. Llkhchi, Y. Sharghi, K. Ghanavati (2011). Fracture density estimationfrom Petrophysical log data using the adaptive neuro-fuzzy inference system,Journal of Geophysics and Engineering 9 (1).
[20] Jang, J. S. R. (1993). Adaptive network based fuzzy inference systems, IEEE Transactions on Systems, Man and Cybernetics, 665–685.
[21] Kablan, A. (2009). Adaptive neuro-fuzzy inference system for financial trading usingintraday seasonality observation model, World Academy of Science, Engineering and Technology 34, 479–488.
[22] Khameneh, N. B., H. Arabalibeik, P. Salehian, S. Setayeshi (2012). Abnormal red bloodcells detection

using adaptive neuro-fuzzy system, Studies in Health Technology and Informatics 173, 30–34.
[23] Khyaatt, M., Mewada, Sinhat A. Verma, B (2013). Adaptive Neuro-Fuzzy Inference System (ANFIS) Based Software Evaluation, International Journal of Computer Science Issues, 10(5),
[24] Kolawa, Adam; Huizinga, Dorota (2007). Automated Defect Prevention: Best Practices in Software Management, Wiley-IEEE Computer Society Press.
[25] Makhsoos, N. T., R. Ebrahimpour, A. Hajiany (2009). Face recognition based onneuro-fuzzy system, International Journal of Computer Science and NetworkSecurity 9(4), 319-326.
[26] Marza, D., D. Seyyedi, L.F. Capretz (2008). Estimation development time of software projects using a neuro-fuzzy approach, World Academy of Science, Engineering and Technology, 22, 575–579.
[27] Obi, J. C. and A.A. Imainvan (2011). Decision support system for the intelligient identification of Alzheimer using neuro fuzzy logic, International Journal on SoftComputing, 2(2), 25–38.
[28] Pergola, Teresa M., L. Mlissa Walters (2013). Evaluating web-based learning systems.
[29] Roberts, T. L. and Moran, T. P. (1983). The evaluation of text editors: Methodology and empirical
[30] Sevarac, Z. (2006). Neuro fuzzy reasoner for student modelling, IEEE International Conference on Advanced Learning Technologies, 740–744.
[31] Sindal, R., S. Tokekar (2009), A neuro-fuzzy call admission control algorithm forvoice/data traffic in CDMA cellular network, in: IEEE International Advance Computing Conference, 827–832.
[32] Sood, A., S. Aggarwal (2011). Crossroads in classification: comparison and analysis of fuzzy and neuro-fuzzy techniques, International Journal of Computer Appli-cations 24 (2), 13–17
[33] Suchman, E. A. (1967). Evaluation Research: Principles and practice in public service and social action programs. New York: Russel.
[34] Sudheer, Ch., S. Mathur (2012), Modeling uncertainty analysis in flow and solutetransport model using adaptive neuro fuzzy inference system and particleswarm optimization, KSCE Journal of Civil Engineering 14 (6), 941–951.
[35] Taylan, O., B. Karagozoglu (2009). An adaptive neuro-fuzzy model for prediction ofstudent's academic performance, Computers & Industrial Engineering 57 (3), 732–741.
[36] Whitefield, A., Wilson, F. & Dowell, J. (1991). A framework for human factors evaluation. Behaviour and Information Technology, 10, 65–79.
[37] Zhou, Z. (2009). Evaluating Websites Using a Practical Quality Model.
[38] Zulkefli Mansor , Zarinah Mohd Kasirun, Saadiah Yahya, Noor Habibah Arshad (2012) International Journal of Computer Science Issues, 9(3),