http://www.ftsm.ukm.my/apjitm
Asia-Pacific Journal of Information Technology and Multimedia
Jurnal Teknologi Maklumat dan Multimedia Asia-Pasifik
Vol. 5 No. 2, December 2016: 75 - 84
e-ISSN: 2289-2192

MEASURING RELIABILITY OF ASPECT-ORIENTED SOFTWARE USING A COMBINATION OF ARTIFICIAL NEURAL NETWORK AND IMPERIALIST COMPETITIVE ALGORITHM

MOHAMMAD ZAVVAR
SHOLE GARAVAND
MOHAMMAD REZA NEHI
AMANGALDI YANPI
MEYSAM REZAEI
MOHAMMAD HOSSEIN ZAVVAR

ABSTRACT

Aspect-oriented software engineering provides new ways to produce and deliver products and ultimately leads to reliable software. Reliability is an important issue contributing to the quality of software. Thus, software engineers need proven mechanisms to determine the extent of software reliability. In this paper, a method for measuring reliability is proposed which takes advantage of a Multilayer Perceptron Artificial Neural Network (MLPANN). Furthermore, an Imperialist Competitive Algorithm (ICA) is used to optimize the weights to improve network performance. Finally, relying on Root Mean Square Error (RMSE), the proposed approach is compared to a hybrid Genetic Algorithm- Artificial Neural Network (GA-ANN) method. The results show that the proposed approach exhibits lower error.

Keywords: Aspect-Oriented Software reliability, Software Quality, Artificial Neural Network, Imperialist Competitive Algorithm, Root Mean Square Error.

INTRODUCTION

Aspect-oriented software engineering provides new ways to modularize software systems by separating multiple concerns or interventional and automatic defragmentation of the system. Ultimately, this leads to products with greater quality and reliability (Kienzle et al., 2010; Kiczales et al., 1997). Aspect-oriented software includes operational and non-operational concerns. However, most methods neglect to take non-operational conditions into account, which poses serious problems for software development. It is important to review the operating conditions of non-operational features in sensitive software systems. High reliability ensures that information is exchanged accurately at the time of urgency in the network.

Additionally, reliability is a prerequisite to accessibility in order to provide services and security on the network. Therefore, the realization of goals such as customer satisfaction and attaining maximum profits depends on providing high reliability components. As a result of the rapid changes in software engineering, the industry is constantly trying to develop new tools and methodologies. Small teams continue with the same conditions and without development activities; however, a precursor is to create a new software system. Currently, software engineers aim to upgrade current business systems and they try to create configurations which are capable competing in the market in addition to having high performance (Bosch, 2010). Hence, instead of coding, software engineering focuses on architectural design, component selection and system integration tasks.

Software engineering is an active process that is used for both top-down and bottom-up methods simultaneously. The architecture is the result of a top-down method. Control and application integration cannot be carried out using old software. Therefore, system architecture needs to enable such tasks and help teams in charge of changing the system (Bosch, 2010).

Researchers have noted that the quality of system architecture is the most important factor contributing to the success of software engineering. An adequate software architecture is characterized by three factors as follows (Palviainen et al., 2011):

1. Changes are easy to implement and the old components can be easily replaced with new ones.
2. The quality of the architecture and implementation can be readily measured.
3. Guarantees that requirements are fulfilled and the system works efficiently.

In this paper, measuring reliability is proposed as a part of software quality measurement. Reliability is one of the issues for performance quality. It is defined as operations without software defects in specific environments and within a specified time (Rodrigues et al., 2005). Four dimensions of reliability can be identified as follows (Rainer & Hall, 2003):

1. Probabilities: In the confidence interval, the system may have a chance of success.
2. Operations of target: The system is not considered reliable if target operation are not performed seamlessly.
3. Time dependence: In certain timeframe or before a certain time t, the system performs operations without the smallest defects and errors.
4. Special conditions: Under certain circumstances or in particular terms based on hardware or communication devices, the system performs operations.

In this paper, these components are considered and evaluated as important themes. Software reliability is important for stakeholders, especially resellers and end-users who increasingly complain about reliability. Among other important factors such as scalability and storage efficiency, reliability has a significant impact on the software lifecycle, because it tends to interfere with its implementation. Reliability is a broad term: each software program that runs in a distributed environment also encompasses various definitions (Ahmed & Wu, 2013). Predicting software reliability applications is different from one environment to another. In recent decades, commercial applications in distributed environments have allowed vendors access to various technologies and components.

In addition, software reliability is a vital factor in determining the quality of software. In many industries, reliability measurement based on the practical application of software in a variety of industries, is associated with reduced financial and human risks. The practice enables developers to examine software quality factors and improve their design and code. It is important that software engineers be aware of the extent of reliability in their products. This fact highlights the significance of software reliability.

Thus, given the importance of reliability in software, researchers need to find smart and efficient ways to accurately measure the reliability of aspect-oriented software. So, this paper aims to propose a novel method for measuring software reliability using Multilayer Perceptron Artificial

Neural Networks (MLPANN) and the Imperialist Competitive Algorithm (ICA) to optimize the weights leading to improved network performance. The proposed approach in this paper uses a perceptron multilayer network combined with the ICA to measure software reliability. The method is compared to a hybrid Genetic Algorithm-Artificial Neural Network (GA-ANN).

BACKGROUND

A review of the literature reveals that research on aspect-oriented software reliability measurement is scarce. This section provides a brief overview of previous works in the field of reliability and aspect-oriented software.

Cacho et al. (2006) proposed a set of metrics and a set of rules for the measurement of products based on different aspects. This aspect-oriented framework allows developers to evaluate their products in design and implementation phases. The metrics include Concern Diffusion over Components (CDC), Concern Diffusion over Operations (CDO) and Concern Diffusions over LOC (CDLOC) for separation of concerns, as well as Coupling Between Components (CBC) and Depth Inheritance Tree (DIT) for attachment or coupling and Lack of Cohesion in Operations (LCOO) for cohesion. Finally, a tool for the proposed method was produced.

Aspect-oriented programming is a technique whereby secondary operations and support business logic are distinguished from the main program (Dadhich & Mathur, 2012). A part of this program is added to the maintenance of the facility to be constructed. Apart from the operating conditions, in software development, operating conditions and other factors such as reliability, compatibility and performance need to be taken into consideration. The quantitative aspect-oriented software reliability model ISO / IEC 9126 was proposed by the authors. As a result of the unpredictable nature of software quality features, a fuzzy multi-criteria method was used to improve the proposed approach.

Ahmed & Wu (2013) focused on the reliability of distributed systems. Since failure detection and the programs aimed at their discovery can provide interactions among end-users, the methods of accuracy and reliability were discussed and explained in more detail and in terms of individual elements why patterns of complete reliability is required to run applications in a distributed system. Also, an analysis of recent studies was performed to predict the reliability of large-scale distributed applications in four levels. Firstly, the conditions to enhance reliability, and the importance of certain issues of reliability in different environments such as cloud computing, service-oriented architecture and grid computing were expressly stated. Secondly, factors and specific challenges as well as defect detection, error discovery, and defect elimination were described. Thirdly, different patterns, factors and challenges in predicting and evaluating software applications in the distributed system were discussed. Finally, the limitations of current patterns were expressed and recommended further investigation into forecasting and analyzing the reliability of distributed applications.

According to Palviainen et al. (2011) reliability is the most important factor in security-critical systems, including, clinics, medical centers and traffic controllers. Moreover, given that reliability is one of the most important features of system quality, important decisions must be taken in designing logic to significantly impact the reliability of software systems. Therefore, in order to achieve this goal, the authors considered architecture and components that need to be evaluated. They carefully discussed reliability evaluation during design and implementation.

Zhang and Pham (2000) showed the empirical findings of 13 trading companies that participated in a study aimed at better understanding the factors affecting reliability. The available tools were reviewed and a detailed analysis of the most important factors in the reliability software was conducted. A more in-depth investigation on the perspective of the participants, managers, system engineers, programmers, testers and others involved in the project development was emphasized. The relative weight of the two methods, including Analysis of Variance (ANOVA) was used to analyze the factors affecting the reliability of software. Their paper serves as a basis for further evaluation and review of important features in the software development.

ARTIFICIAL NEURAL NETWORK

Artificial Neural Networks (ANN), are a novel computational method with applications in machine learning and knowledge representation, which apply the knowledge gained in response to higher projected output of complicated systems. The main idea of these networks is inspired by the way the biological nervous system function, to process data and information in learning and knowledge creation (Gupta, 2013). The key element of this idea is the creation of new structures for information processing systems. The system comprises an extremely large number of interconnected neurons to solve a problem. In many complex mathematical problems, to solve complex nonlinear equations, a multilayer perceptron network can be easily used to tune the weight of the parameters (Yetilmezsoy & Demirel, 2008). In this type of ANN, a three-layer structure is used: (1) an input layer; (2) a hidden layer; and (3) an output layer that produces the result. An example of a network Multilayer perceptron is shown in Figure 1.

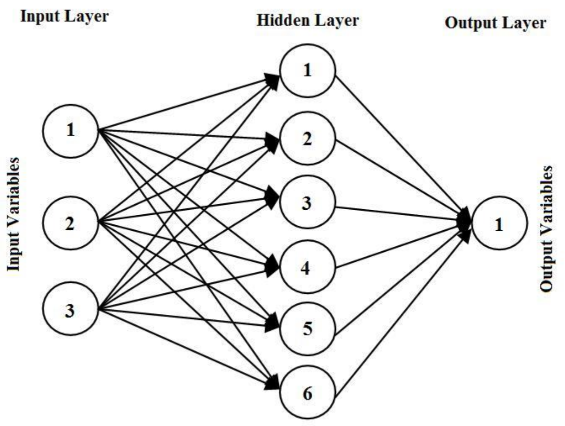

FIGURE 1. Multilayer Perceptron structure with hidden neurons and output neurons with linear function

In this network, the actual output is compared with the desired output and the weights are tuned using a supervised propagation algorithm to form the appropriate pattern. For input pattern p, square output error for all cells output layer as follows:

$$E_p = \frac{1}{2}(d^p - y^p)^2 = \frac{1}{2}\sum_{i=1}^{s}(d_j^p - y_j^p)^2 \tag{1}$$

where d_j^p is the desired output for the j-th cell in the output layer, s is output vector dimension, y^p is the real output vector, and d^p is the desired vector output . The total square error E to P model is obtained as follows:

$$E = \sum_{p=1}^{P} E_p = \frac{1}{2}\sum_{p=1}^{P}\sum_{j=1}^{s}(d_j^p - y_j^p)^2 \tag{2}$$

The objective in adjusting the weights is to minimize the cost function E, using gradient descent. The equations updating the weights are as follows:

$$w_{ij}(t+1) = w_{ij}(t) + \eta\Delta_{w_{ij}}(t)$$
$$+\alpha\Delta_{w_{ij}}(t-1)$$

(3)

where $\Delta_{w_{ij}}(t) = -(\frac{\partial E_p}{\partial w_{ij}(t)})$ ، η denotes the learning factor, α the moment factor, $w_{ij}(t+1)$ represents the new weight and $w_{ij}(t)$ is the previous weight. In this method, weights are repeatedly updated for all learning patterns.

The process of learning stops when the total error for pattern p is less than the specified threshold, or the number of training iterations exceeds the specified value. It should be noted that in the training method the probability of convergence to local minima decreases.

IMPERIALIST COMPETITIVE ALGORITHM (ICA)

The ICA was developed in 2007, with the aim of optimizing the social and political processes with a modern modeling approach (Atashpaz-Gargari & Lucas, 2007). In this algorithm, to solve the optimization problem, **N** countries are considered each of which is shown with a vector and represents a point in an n-dimensional space. Among these points, the ones having the lowest cost according to the function of optimization are considered as a colony and the rest as the colonizer. For every settler, first, the normalized cost is calculated as follows:

$$C_n = \max\{c_i\} - c_n$$

(4)

where C_n is the imperialist cost of n, $\max\{c_i\}$ is the maximum cost among the imperialists and **c_n** represents the normalized cost of these imperialists. With the cost normalized, normalized relative strength of each imperialist is calculated as follows:

$$p_n = \left| \frac{C_n}{\sum_{i=1}^{N_{imp}} C_i} \right|$$

(5)

Therefore, the number of the imperialist colonies will be equal to:

$$N.C_n = round\{p_n.(N_{col})\}$$

(6)

where $N.C_n$ is the initial numbers of colonies of an empire, and N_{col} is also the total number of existing colony countries in the population of the initial countries. ICA begins with the initial state of all empires. Figure 2 shows the colonies of an imperialist country.

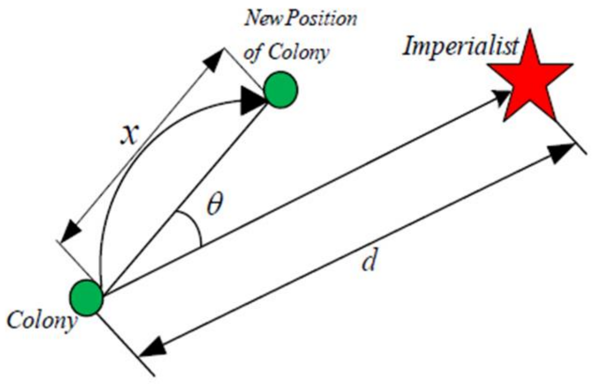

FIGURE 2. An overview of the colonies to the imperialist

As it is shown in Figure 2, for x, the following is true:

$$x \sim U(0, \beta * d) \tag{7}$$

where β is a number larger than one and close to two, the value β =2 can be a good choice. The coefficient $\beta > 1$ causes the colonized country to become closer to the colonizer while approaching in from different directions. To expand the search area around the colony, a deviation angle equal to θ that follows a uniformly distributed random track is added to the original vector:

$$\theta \sim U(-\gamma, \gamma) \tag{8}$$

where γ is a parameter that controls the span of groups angle – equal to $\frac{\pi}{4}$ in this paper. The next step is a colonial competition. At this stage, the weakest of the weak colonial empire is selected and is assigned to a powerful empire (not necessarily the most powerful empire) but the probability of an empire being selected is proportional to its strength. Finally, when an empire loses all its colonies that empire is removed from the list of empires and it is assigned to another empire in a competition. The evolutionary process is performed in a loop that continues to satisfy a stop condition.

THE PROPOSED METHOD

In a neural network, the aim of the learning process is to determine the optimal values of weights and other parameters such as bias and the slope of the function in order to obtain the highest efficiency. To estimate a function using a neural network, estimation error is defined the difference between the network and actual outputs for the training and testing data sets. Here, the goal is to reduce the error by appropriately adjusting the weights and other network parameters. The commonly used method is back propagation. In this method which is inspired by gradient-based optimization methods, the weights of each stage of learning are calculated through the weights of the previous phase and move one step in the opposite direction of the error function gradient.

Since each type of neural network learning leads to an optimization problem, a way forward is to use evolutionary algorithms. In this paper, the ICA is used to determine the weight of the neural network, then the performance of this algorithm in determining the weights of neural networks is compared to that of a GA. The steps involved in determining the weights of the neural network using ICA are shown in Figure 3.

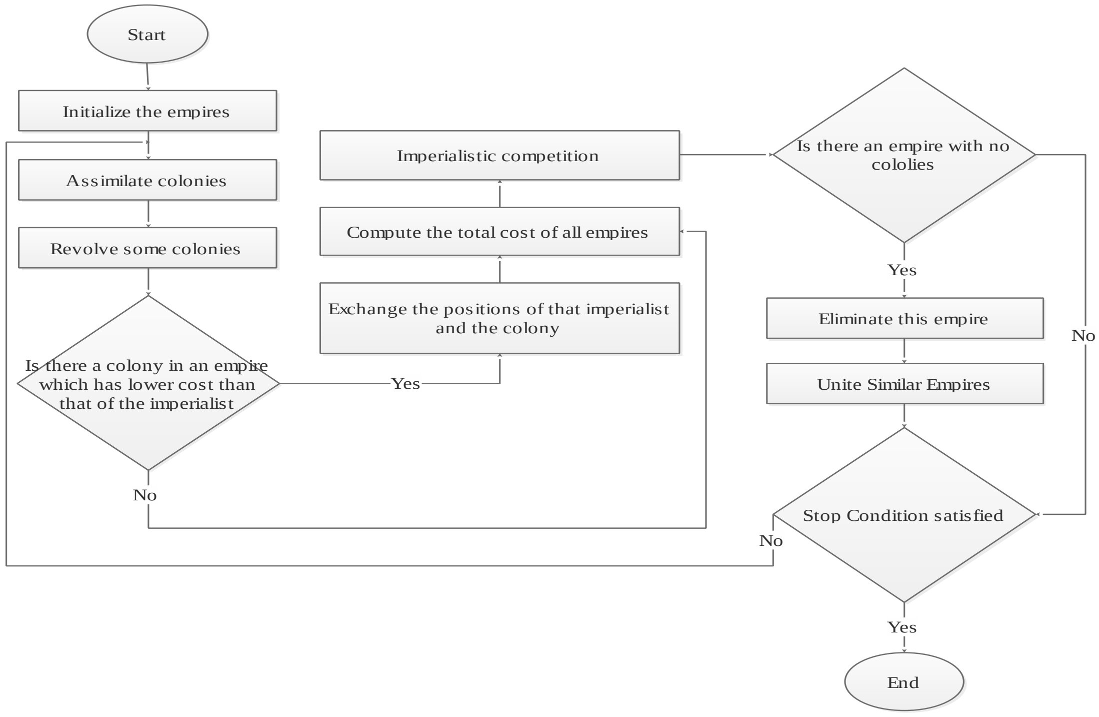
Start
Initialize the empires
Assimilate colonies
Revolve some colonies
Is there a colony in an empire which has lower cost than that of the imperialist
Yes
No
Exchange the positions of that imperialist and the colony
Compute the total cost of all empires
Imperialistic competition
Is there an empire with no cololies
Yes
No
Eliminate this empire
Unite Similar Empires
Stop Condition satisfied
No
Yes
End

FIGURE 3. Stages of determining the weights of the neural network by algorithms of colonial competition

RESULTS AND DISCUSSIONS

In this section, the results of optimization algorithm of colonial competition for learning neural network are studied and compared with those of a procedure based on GA.

As in Zavvar and Ramezani (2015a), a collection of data on software projects based on aspects is used. This dataset contains 1000 samples with four inputs and one output which is the estimated value of reliability for the considered software. The input of this dataset is as follows:

1. Concern Diffusion over Operations
2. Degree of Scattering across Classes
3. Degree of Scattering across Methods
4. Lines of Concern Code

In this study, 70 percent of data were used for training, 15 percent for validating and 15 percent for testing the neural network. The neural network used in this article, is an MLP neural network composed of two layers with seven neurons in the hidden layer. The number of training transitions equals 20 and trainlm was used for training network from the function. Also, in the ICA, a total of 200 countries were considered, the number of primary Empires equals to 30 and the algorithm is executed 50 times.

For comparison, the proposed method was compared to a GA with a population of 200, a 10 percent probability of mutation, a 60 percent probability of crossover and a total of 50 generations.

To evaluate and compare the proposed method with other methods, the RMSE criterion was used. As mentioned in Zavvar and Ramezani (2015b), one of the criteria used in evaluating models is the Root Mean Square Error (RMSE) between the expected and predicted values. The

RMSE is a suitable tool for comparing prediction errors in a dataset. This value is calculated using the following equation:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(y_i - y)^2}{n}} \tag{9}$$

First, the individual differences are calculated and squared. Then, this mean difference is applied and finally the square root of the average number is presented yielding RMSE. The measure is a criterion for the accuracy of the results and usually smaller values indicate better model fit. In addition, the results of both methods are shown with RMSE criterion. Price and regression curves are shown for each of the methods in Figure 4.

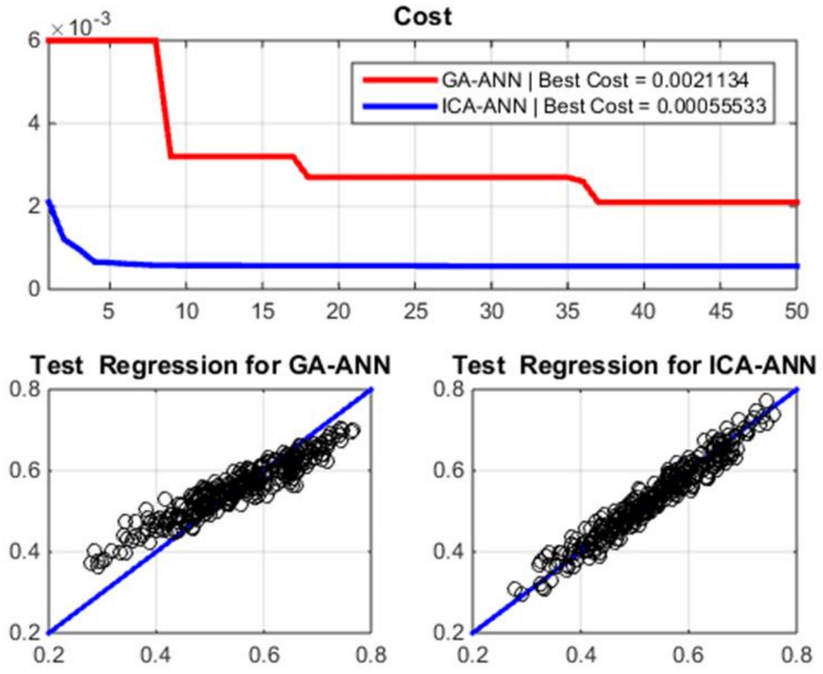

FIGURE 4. Price and regression curves for each of the methods

In Figure 4 (top), the cost of the ICA-ANN method is compared to that of GA-ANN. As shown, the ICA-ANN clearly outperforms GA-ANN. To achieve certain cost, ICA-ANN requires less time. Also, after 50 iterations, the GA-ANN method achieves a cost of 0.0021134 while the ICA-ANN obtains 0.00055533 which shows that the proposed methodology exhibits a smaller final cost. Also, the regression graph and scattering plots are shown around the regression data for each of the methods. Moreover, at the bottom of Figure 4, regression graph and distribution of data for each of the methods are shown in the Test phase.

Figure 4 shows that the proposed method yields values in the test that are well around the center. In Figure 5, RMSE and performance chart of each of these methods in measuring reliability are shown.

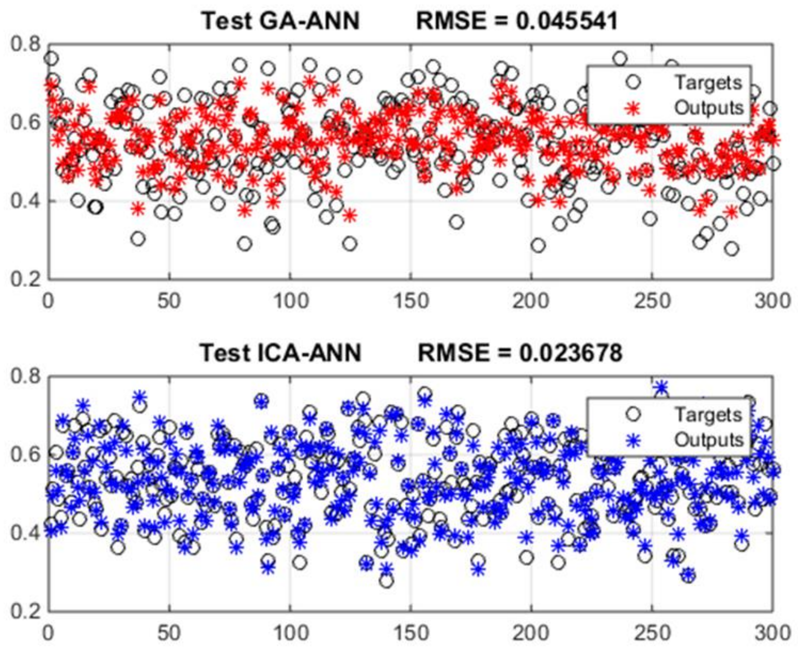

FIGURE 5. Performance chart for each of the methods

In Figure 5, the value of the reliability for each instance of data is shown along with the predicted values for each sample. As seen, output of the colonial competitive algorithm-based approach is closer to the targets. With respect to the performance of each method, the value of RMSE was calculated. RMSE in the proposed approach was equal to 0.023678 while the GA-ANN generated 0.045541 which indicates that the proposed method had fewer errors and better accuracy in evaluating aspect-oriented software reliability.

CONCLUSION

Upon discussing the importance of measuring the reliability of aspect-oriented software and examining its influencing factors, a method was presented based on ICA-ANN for measuring the reliability of aspect-oriented software to improve the network performance. The ICA was used to optimize the weights of the network. Then, the prediction accuracy of the proposed method was compared with a hybrid GA-ANN method in terms of RSME. The proposed approach resulted in an RMSE of 0.023678 while the value for ICA-ANN was equal to 0. 045541 indicating that the proposed method offers better performance in measuring aspect-oriented software reliability. This paper only focused on four effective characteristics on reliability, providing a method for evaluating aspect-oriented software reliability that also considers other features needs further studies that will be discussed in the future.

REFERENCES

Ahmed, W. & Wu, Y.W. 2013. A survey on reliability in distributed systems. *Journal of Computer and System Sciences*, 79(8):1243-1255.

Atashpaz-Gargari, E. & Lucas, C. 2007. Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition. *IEEE Congress on Evolutionary Computation*, 25-28 Sept., Singapore, 4661-4667.

Bosch, J. 2010. *Architecture in the age of compositionality*. Heidelberg: Springer-Verlag.

Cacho, N., Sant'Anna, C., Figueiredo, E., Garcia, A., Batista, T. & Lucena, C. 2006. Composing design patterns: a scalability study of aspect-oriented programming. *Proceedings of the 5th international conference on Aspect-oriented software development.* New York: ACM, 109-121.

Chen, D. J., Sheng, M. C. & Horng, M. S. 1993. Real-time distributed program reliability analysis. *Proceedings of the Fifth IEEE Symposium on Parallel and Distributed Processing.* Dallas: TX, 771-778.

Dadhich, R. & Mathur, B. 2012. BM, Measuring Reliability of an Aspect Oriented Software Using Fuzzy Logic Approach. *International Journal of Engineering and Advanced Technology (IJEAT)*, 1(5):233-237.

Grassi, V. 2005. *Architecture-based reliability prediction for service-oriented computing*. Heidelberg: Springer-Verlag.

Gupta, N. 2013. Artificial neural network. *Network and Complex Systems*, 3(1):24-28.

Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Lopes, C., Loingtier, J. M. & Irwin, J. 1997. *Aspect-oriented programming.* Heidelberg: Springer-Verlag.

Kienzle, J., al-Abed, W., Fleurey, F., Jézéquel, J. M. & Klein, J. 2010. *Aspect-oriented design with reusable aspect models*. Heidelberg: Springer-Verlag.

Palviainen, M., Evesti, A. & Ovaska, E. 2011. The reliability estimation, prediction and measuring of component-based software. *Journal of Systems and Software*, 84(6):1054-1070.

Rainer, A. & Hall, T. 2003. A quantitative and qualitative analysis of factors affecting software processes. *Journal of Systems and Software*, 66(1):7-21.

Rodrigues, G. N., Rosenblum, D. S. & Uchitel, S. 2005. *Reliability prediction in model-driven development*. Heidelberg: Springer-Verlag.

Yetilmezsoy, K. & Demirel, S. 2008. Artificial neural network (ANN) approach for modeling of Pb (II) adsorption from aqueous solution by Antep pistachio (Pistacia Vera L.) shells. *Journal of Hazardous Materials*, 153(3):1288-1300.

Zavvar, M. & Ramezani, F. 2015a. Comparison of ANFIS with MLP ANN in Measuring the Reliability based on Aspect Oriented Software. *International Journal of Modern Education & Computer Science*, 7(9):29-35.

Zavvar, M. & Ramezani, F. 2015b. Measuring of Software Maintainability Using Adaptive Fuzzy Neural Network. I*nternational Journal of Modern Education & Computer Science*, 7(10):27-32.

Zhang, X. & Pham, H. 2000. An analysis of factors affecting software reliability. *Journal of Systems and Software*, 50(1):43-56.

Zheng, Z. & Lyu, M. R. 2008. Ws-dream: A distributed reliability assessment mechanism for web services. *IEEE International Conference on Dependable Systems and Networks with FTCS and DCC (DSN)*, 23 September 2008, Anchorage, 392-397.

Zheng, Z. & Lyu, M. R. 2010. Collaborative reliability prediction of service-oriented systems. *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering*, 1-8 May, New York, 35-44.

Mohammad Zavvar,
Shole Garavand,
Mohammad Reza Nehi,
Amangaldi Yanpi,
Meysam Rezaei,
Mohammad Hossein Zavvar
Department of Computer Engineering,
Sari Branch, Islamic Azad University, Sari, Iran
zavvar.developer@gmail.com, shole_geravnd@yahoo.com, nehi@gmail.com, amangaldi.yanpi@gmail.com, mem_re69@yahoo.com. hossein.zavvar@outlook.com

Received: 22 September 2016
Accepted: 23 January 2017