Dogo Rangsang Research Journal UGC Care Group I Journal
ISSN : 2347-7180 Vol-10 Issue-01 No. 01 January 2020

APPROXIMATING CONSISTENCY OF CONSTITUENT CONSTRUCTED SOFTWARE USING ARTIFICIAL NEURAL NETWORK

Nalini Ku Sethi Asst. Prof. Einstein Academy of Technology and Management, Bhubaneswar,
Jitanshu Sekhar Patra Asst. Prof. Einstein Academy of Technology and Management, Bhubaneswar, India
Deepak Behera Student, Einstein Academy of Technology and Management, Bhubaneswar,

Abstract
Numerous software steadiness copies have been future but no one is proven to be best for various applications. Element Based Schemes attain springiness by plainly separating the stable parts of systems from the description of their composition. In order to approximation the consistency of Factor Based Software System, it is required to measure the consistency of each component of the system. However, due to the black-box nature of components, where the source code of these components are not available, it is difficult to use conventional metrics in Component Based Development as these metrics require analysis of source codes. Artificial Neural Networks have been recognized as attractive alternatives to the standard, well-established "hard computing" paradigms. So we adopt Soft Computing Techniques based approach to estimate the reliability of Component Based Software, based on the reliabilities of the individual components and the architecture of the system.
Keywords : Software Components, software reliability, Neural Network, Software Quality

1. Introduction

The concept of building software from components is not new. A "classical" design of complex Software systems always be- gins with the identification of system parts designated sub- systems or Blocks, and on a lower level modules, classes, procedures and so on. The reuse approach to Software de- velopment has been used for many years [1]. From the last many years it has shown that the Object-Oriented technol- ogy alone is not enough to cope with the rapidly changing requirements of present day applications. It's one of the rea- sons is that, the Object-Oriented (OO) methods encourage developing other methods that reflect the object of problem domain. Today's applications are large, complex and are not integrated. Although they come packaged with a wide range of features but most features can neither be removed, up- graded independently or replaced nor can be used in other applications. In particular, OO methods do not typically lead to designs that make a clear separation between computa- tional and compositional aspects. While Component Based Software Development (CBSD) is understood to require re- usable components that interact with each other and fit into system architectures, Today Component Based Software De- velopment (CBD) is getting accepted in the company or an industry as a new effective development paradigm. By using component we can reduce development cost and increase the reliability of entire software system. The major advan- tages of CBSD are in-time and high quality solutions. Higher productivity, flexibility and quality through reusability, replace- ability, efficient reusability and scalability are some additional benefits of CBSD.

In modern software systems, reliability is considered to be one of the most critical non-functional properties. To build software in cost-efficient, reliability should be estimated dur- ing design the system. The number of techniques for estimat- ing reliability of component based software system has been proposed, they have tended to oversimplify one or more of
(1) the definition of reliability of a software system, (2) param- eters influencing a system's reliability and (3) the challenge of estimating model parameters by assuming they are available or can easily be obtained [2]. The parameters in these soft- ware reliability models are usually directly obtained from the field failure data. Due to the dynamic properties of the system and the insufficiency of the failure data, the accurate values of the parameters are hard to determine.

Therefore to estimate the reliability of Component Based Software, Artificial Neural Network has been proposed here to estimate the reliability of Component Based Software, the reliability of the system is based on the reliabilities of the individual components and the architecture of the system.

2. Background Concepts

In this section, section 2.1 describes the Component Based Software Engineering (CBSE), section 2.1 describes about the Software Reliability and section 2.3 describes about the Artificial Neural Network (ANN).

2.1 Component Based Software Engineering (CBSE)

The reuse approach to software development has been used for many years. However, the recent emergence of new tech- nologies has significantly increased the possibilities of build- ing systems and applications from reusable components. Both customers and suppliers have had great expectations from CBD, but their expectations have not always been sat- isfied. Experience has shown that component-based devel- opment requires a systematic approach to and focus on the component aspects of software development. Traditional software engineering disciplines must be adjusted to the new approach, and new procedures must be developed. Compo- nent-based Software Engineering (CBSE) has become rec- ognized as such a new sub-discipline of Software Engineer- ing. Component Based Software Engineering (CBSE) also known as Component Based Development (CBD)) is a branch of software engineering which emphasizes the sepa- ration of concerns in respect of the wide-ranging functional- ity available throughout a given software system. Software engineers regard components as part of the starting platform for service-orientation. Components play this role, for exam- ple, in Web Services, and more recently, in Service-Oriented Architecture (SOA) - whereby a component is converted into a service and inherits further characteristics beyond that of an ordinary component. An individual component is a software package. To make a Component Based System the required components are put together, each component can commu- nicate to other component via their interface characteristics.

2.2 Software Reliability

The most important software product characteristics are "quality, cost, and schedule". Reliability is probably the most important of the characteristics inherits in the concept "Soft- ware Quality". Software reliability is a critical component of computer system availability. It is the probability that the soft- ware will work without failure for a specified period of time [3]. To estimate the reliability of software various Software Reli- ability Growth Model (SRGM) has been developed. During the past 30 years many SRGM have been proposed. There are many variables to be considered in developing a Software Reliability Growth Model methodology i.e. Amount of testing, Defect Data, Grouped Data, Growth Model Type, Statistical Techniques [4]. The basic approach is to model past failure Thus to obtain the expected reliability of the application, we need to obtain $E[R^{m1,i}]$ which is the expected reliability of component i for a single execution. From the Taylor series expression of the function of a random variable [18] we have:

$E[Ri^{m1,i}] = Ri^{E[m1,i]}+1/2(Ri^{E[m1,i]})+(\log Ri)(\log Ri)+Var[m1,i]$ (2) Where:
Var[m1,i] is the variance of individual component.

Since the number of visits to the absorbing state n is always data to predict feature behavior. This approach employee ei-1, E[m] = 1 and Var[m] = 0 and hence $E[R^{m1,n}] = R$. Equa-

$1,n \quad 1,n \quad n \quad n$

ther the observed number of failure discovered per time pe- riod or the observed time between failures of the software. The model therefore divided into two categories, depending upon the types

of data the model use, (i) Failure per time tion (1) can be written as:

$$E[R_i^{mI,i}] = Ri^{E[m1,i]} + \frac{1}{2}(Ri^{E[m1,i)}) + (\log Ri)(\log Ri) + Var[m1,i] \quad (3)$$

period and (ii) Time between failures.

2.3 Artificial Neural Network (ANN)

Neural networks are simplified models of biological nervous system and therefore have drawn their motivation from the kind of computing perform by a human brain [5]. An NN in general, is a highly interconnected network of a large number of processing elements called neurons in an architecture inspired by the brain. The intelligence of a neural network emerges from the collective behavior of neurons, each of which performs only limited operation. Even though each in- dividual neuron works slowly, they can still quickly find a solu- tion by working in parallel. Neural networks are composed of simple elements operating in parallel. These elements are inspired by biological nervous systems. As in nature, the connections between elements largely determine the net- work function. You can train a neural network to perform a particular function by adjusting the values of the connections (weights) between elements. The strength of the interconnec- tions between neurons is implemented by means of the syn- aptic weights used to store the knowledge. Typically, neural networks are adjusted, or trained, so that a particular input leads to a specific target output. The Fig 1 illustrates such a situation. The network is adjusted based on a comparison of the output and the target, until the network output matches the target. Typically, many such input/target pairs are neededto train a network.

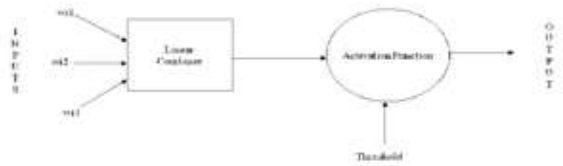


Fig 1: Basic Neural Network

3. Existing Technique

In component based software system, if a system consists ofn components with reliabilities denoted by R1,…, Rn respec- tively, the reliability of system is an execution path, 1, 3, 2,3, 2, 3, 4, 3, n, is given by Rs. Thus, the objective here is to estimate the reliability of a system by averaging over all path reliabilities. For this propose system consider the architecture of software as shown in Fig 2. Assume that, the application consists of n components, and has a single initial state de- noted by 1, and a single absorbing or exit state denoted by n.

The expected reliability of system is defined by the following equation:
The result is shown in Table 2 according to equation (2) and (3).

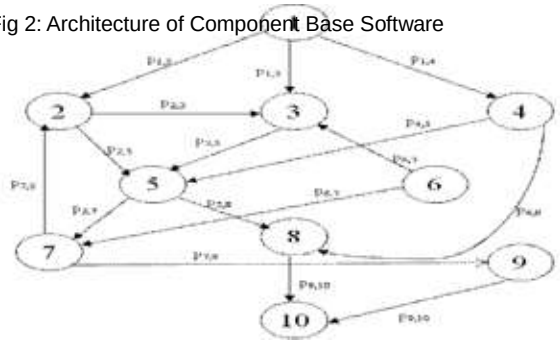
p1,2 = 0.60	p1,3 = 0.20	p1,4 = 0.20	
p2,3 = 0.70	p2,5 = 0.30		
p3,5 = 1.00			
p4,5 = 0.40	p4,6 = 0.60		
p5,7 = 0.40	p5,8 = 0.60		
p6,3 = 0.30	p6,7 = 0.30	p6,8 = 0.10	p6,9 = 0.30
p7,2 = 0.50	p7,9 = 0.50		
p8,4 = 0.25	p8,10 = 0.75		
p9,8 = 0.10	p9,10 = 0.90		

Table 1: Transaction probability of components

ComponentNo.	Assumed Reliability (Ra)	Mean(m)	Variance(v)	Estimated Reliability (Rs)
1	0.9990	1.0000	0.0000	0.9990
2	0.9800	0.9077	0.6444	0.9819
3	0.9900	0.9107	0.5499	0.9909
4	0.9700	0.4184	0.3928	0.9874
5	0.9500	1.3504	0.7185	0.9337
6	0.9950	0.2510	0.2319	0.9987
7	0.9850	0.6155	0.6261	0.9908
8	0.9500	0.8737	0.4255	0.9564
9	0.9750	0.3831	0.2462	0.9904
10	0.9850	1.0000	0.0000	0.9850
System Reliability				**0.8267**

Table 2: Estimated reliability using existing technique

Fig 2: Architecture of Component Base Software



4. Proposed ANN approach

In this section we proposed ANN approach to estimate the re- liability of Component Based Software (CBS). To estimate the reliability of CBS, one needs to establish a relationship of the factors with reliability to achieve the desired goal. Following parameter has been identified, which will influence reliability of CBS:

- Assumed Reliability (Ra) of component: We can assume

$$E[R] = E[\prod^{n} Ri^{m1,i}] \qquad (1)$$

the reliability of component according to its transaction s

Where:

$i=1$

probability from one component to other component.

- Mean (m) value of component: Denotes the number of $E[R_S]$ is the estimated reliability of the system.

$Ri^{m1,i}$,i is the reliability of individual component.

visits to state j starting from state i before the process absorbed. Changing in mean value of component may affect the reliability of component. We can calculate the mean value using DTMC [8] method.

- Variance (v) value of component: Denotes the number of visits to state j starting from state i before the process absorbed. Changing in mean value of component may affect the reliability of component. We can calculate the variance value using DTMC [8] method.

We assume that the architecture of the application modeled using a Desecrate Time Markov Chain (DTMC) and the time spent by the application in each component per visit is a ran- dom variable with known mean and variance. We also as- sume that the reliability of each component per visit is known. We assume that the application consists of n components, and has a single initial stage denoted by 1, and a single ab- sorbing state or exit state denoted by n. We also assume that the components fail independently of each other as well as successive executions.

Usually called “neural network” (NN) is a mathematically model or computational model that is inspired by the structure and/or functional aspects of biological neural networks. Here we propose multi layer feed forward network structure with back-propagation as shown in Fig 3. This feed- forward net- work consists of three layers namely input layer, hidden layer and output layer. Input layer has three input neuron where we apply the value of input parameter like Assumed reliabil- ity (Ra), Mean (m) and Variance (v) of individual component, Hidden layer has nine hidden neuron and output layer has one output neuron where can measure the estimated reliabil- ity (Rnn) of individual component. Liner function is used to calculate the output of the input layer neurons, while logistic function is used to calculate the output of hidden and output layer neuron(s) as shown in Fig 3.

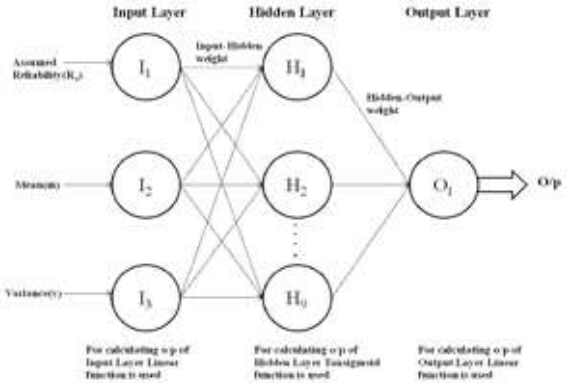


Fig 3: Multilayer Feed forward Network to estimate the reli- ability

5. Performance Evaluation using ANN

In this section we presents the results of the performance evaluation for the application of ANN based on the various parameters of the ANN as well as the various parameters of individual component in system like: Assumed Reliability (Ra), Mean (M), and Variance (V).

Here MATLB version 7.9.0.529(R2009b) is used for perform- ance evaluation to evaluate the system reliability of software using ANN

Experiment setup

The experiment setup is as shown below as per Fig 3

- ANN architecture: Multilayer feed-forward.
- Training Algorithm: Gradient descent with momentum back-propagation algorithm.
- Number of neurons in input layer: 3
- Number of neurons in output layer: 1
- Number of neurons in input layer: The number of hidden layer neurons can be determined through iteratively add- ing neurons until the network error falls under a certain training error goal.
- Transfer function: Tansigmoid and Liner function for hid- den and output layer respectively.

Now we perform number of experiment and take observation at various values of ANN architecture's parameters like: net- work learning rate, number of epochs, number of neuron(s) in hidden layer etc.

So at the parameters value Learning rate = 0.4, Number of Epochs = 1000 and Number of Neurons in hidden layer = 9, we get the accurate result than the existing techniques the results and its comparison graph is as shown in Table 3 and Fig 4 respectively.

Component	E_Rel_R_{nn}	A_Rel_R_a	E_Rel_R_S	Rnn vs R_a	Rnn vs R_S
1	0.9989	0.9990	0.9990	0.0001	0.0001
2	0.9798	0.9800	0.9819	0.0002	0.0021
3	0.9903	0.9900	0.9909	-0.0003	0.0006
4	0.9705	0.9700	0.9874	-0.0005	0.0169
5	0.9501	0.9500	0.9337	-0.0001	-0.0164
6	0.9952	0.9950	0.9987	-0.0002	0.0035
7	0.9846	0.9850	0.9908	0.0004	0.0062
8	0.9500	0.9500	0.9564	0.0000	0.0064
9	0.9745	0.9750	0.9904	0.0005	0.0159
10	0.9851	0.9850	0.9850	-0.0001	-0.0001
System_ Rel	**0.7986**	**0.7986**	**0.8273**	**0.0000**	**0.0287**

Table 3: Results and Comparison

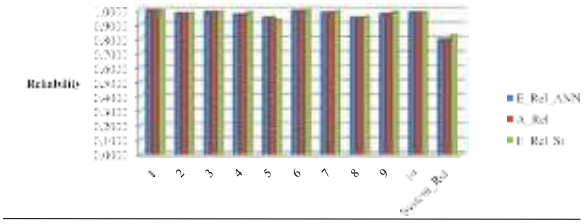


Fig 4: Results and Comparison

6. Conclusion and Future work

Form the section 5, the result Table 3 and Fig 4 we can con- clude that, using ANN approach we can easily estimate the reliability of each component as well as the reliability of Com- ponent Based Software, and the difference between the esti- mated system reliability and Assumed system

reliability is al- most zero. So that we can reduce the error up to 99% than the existing techniques. For the future work we can apply other Soft Computing techniques like Fuzzy Logic.

References:

[1] Ivica Crnkovic “Component-based Software Engineering – New Challenges in Software Development” in citeseerx. [2] Ivo Krka, Leslie Cheung, George Edwards, Leana Golubchik, and Nenad Medvidovic “Architecture-Based Software Reliability Estimation: Problem Space, Challenges, and Strategies” [3] J. D. Musa, A. Iannino, and K. Okumoto (1987). Software Reliability, Measurement, rediction and Application. McGraw-Hill. [4] J. D. Musa (1998). Software Reliability Engineering: More Reliable Software, Faster Development and Testing. McGraw-Hill. [5] Alan wood (September 1996) TANDEM Software Reliability Growth Models. Technical report 96.1, part no 130056. [6] Jung-Hua Lo, Sy-Yen Kuo, Michael R. Lyu*, and Chin-Yu Huang (2002), Optimal Resource Allocation and Reliability Analysis for Component-Based Software Applications, COMPSAC '02 Proceedings of the 26th International Computer Software and Applications Conference on Prolonging Software Life: Development and Redevelopment, pp 7-12, ISBN: 0-7695-1727-7 [7] Nachimuthu Karunanithi, Darrell Whitley, and Yashwant K. Malaiya (JULY 1992), Prediction of Software Reliability Using Connectionist Models, IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 18, NO. I. [8] Swapna S. Gokhale. Kishor S Trivedi (2002), Reliability Prediction and Sensitivity Analysis Based on Software Architecture, IEEE, Software reliability, pp 64-75, ISBN: 0-7695-1763-3.