2010 First International Conference on Pervasive Computing, Signal Processing and Applications

A Comprehensive Approach of SA Based Software Deployment Reliability Estimation in Neural Networks

Xihong Su, Zhibo Wu, Hongwei Liu, Decheng Zuo, Xiaozong Yang
School of Computer Science and Technology
Harbin Institute of Technology
Harbin, China
suxh@ftcl.hit.edu.cn

Abstract—**Software architecture (SA) has been widely advocated as an effective abstraction for modeling, implementing, and evolving complex software systems such as those in distributed, decentralized, heterogeneous and pervasive environments. We intend to investigate two problems related to the domain of those environments: software deployment and reliability. Though many approaches for Architecture-based reliability research, little work has been done in incorporating software deployment into SA based reliability estimation. This paper presents a new comprehensive approach of reliability estimation considering software deployment in neural networks. Additionally, we explain software component replica and different software component deployment architectures in neural networks.**

Keywords-software architecture; software deployment; neural networks; reliability; pervasive system

I. Introduction

Over the past a few decades, we have witnessed an unrelenting pattern of growth in the size and complexity of software systems, which will likely continue well into the foreseeable future. This pattern is further evident in an emerging class of embedded and pervasive software systems that are growing in popularity due to increase in the speed and capacity of hardware, and decrease in its cost, the emergence of wireless ad hoc networks, proliferation of sensors and handheld computing devices, and so on. Many studies have shown that a promising approach to resolve the challenges is to employ the principles of software architectures [1].

In the domain of pervasive systems, of particular interest is a specific facet of a software system's architecture-its deployment. Simply, a system's deployment architecture is the allocation of the system's software components (and connectors) to its hardware hosts. Deployment architecture is particularly important in pervasive environments, because a system will typically comprise many different, heterogeneous, mobile, and possibly mutable, execution platforms during its lifetime [2].

As a branch of the artificial intelligence, the neural network has been noticed broadly over the last years, and it has been developed greatly. It is used in researching non-linear problems by simulating the biology neural network. Arbitrary accuracy can be achieved by adding one or more hidden layers between the input layer and the output layer of the fee-foreword neural network. This had been provided by the neural network theory [3].

Many neural network approaches are used to predict software reliability. For example, Karunanithi first uses neural networks to predict software reliability. Xie proposes a modified Elman recurrent neural network to model and predict software failures [4]. Su and Huang use a dynamic weighted combinational model (DWCM) for software reliability prediction based on neural network approach [5]. However, these approaches do not predict reliability at architecture level. A few approaches for architecture-based reliability research, for example, Cheung computes the reliability of the whole system in different styles of architecture through obtaining the reliability of each component, having high costs due to the testing of each component to reach the reliability[6], and Ivo defines the problem space, challenges and strategies of architecture-based reliability estimation [7]. But they all do not consider software deployment and component replica. Therefore, in this paper, we predict system reliability at architecture-level in neural networks. Furthermore, we explain the component replica and software component deployment in neural networks.

II. TERMINOLOGY AND SCOPE

A. *Software Architecture*

A software architecture is an abstraction of the run-time elements of a software system during some phase of its operation. A system may be composed of many levels of abstraction and many phases of operation, each with its own software architecture [8].

Software architecture provides abstractions for representing the structure, behavior, and key properties of a software system. Those abstractions are described in terms of components (computational elements), connectors (interaction elements), and their configurations (also referred to as topologies).

978-0-7695-4180-8/10 $26.00 © 2010 IEEE
DOI 10.1109/PCSPA.2010.21

B. Software Deployment

Software deployment is referred to as a collection of activities, which are to make software available for use until uninstalling it from devices. These activities include delivery, installation, configuration, activation, updating, re-configuration, and un-installation of the software [9].

Software deployment is a post-production activity that is performed for or by the customer of a piece of software. Today's software often consists of a large number of components each offering and requiring services of other components. Such components are often deployed onto distributed, heterogeneous environments adding to the complexity of software deployment [10].

III. STRUCTURE OF RELIABILITY MODEL IN NEURAL NETWORKS

A. Whole Structure

The basic entities of SA based software deployment include hosts, components, and services. These entities seem appropriate to model component deployment in distributed systems. In details, a model consists of

1) a set of hosts,H,which represents the hardware nodes of a system.

2) a set of components,C. Each component has a set of component replica, which also includes the original component.

3) a set of services,S, which describes the different use cases that the whole system offers and can perform. A service is composed of the interaction of components in a system.

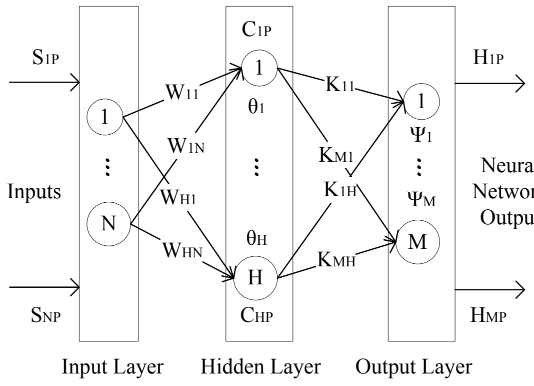

Figure 1. The structure of SA based software deployment in neural networks.

This model can be described in neural networks, as shown in Fig. 1. At first, we analyze the structure of SA based software deployment in neural networks. The network contains an input layer, one hidden layer and one output layer. The relationship needs to be present between all neurons of one layer and all neurons of next layer. The relationship is determined by weight coefficient.Pis the number of software deployment.

The input layer receives the inputs . Hidden layer has a lot of components, which need to be deployed on the hosts. In this method, we can obtain the following expression by feed-foreword neural network:

$$h_{ip} = f(\sum_{j=1}^{H} k_{ij} f(\sum_{k=1}^{N} w_{jk} S_{kp} - \theta_h) - \psi_m), \quad i = 1, 2, ..., M, p = 1, ..., P \tag{1}$$

$$C_{jp} = \sum_{k=1}^{N} w_{jk} S_{kp}, j = 1, ..., H \tag{2}$$

$$f(x - y) = f(x) \cdot f(y) \tag{3}$$

The matrix W and K can be described as follows:

$$W = \begin{array}{c} \\ C_1 \\ C_2 \\ \vdots \\ C_H \end{array} \begin{array}{c} \begin{array}{cccc} S_1 & S_2 & \cdots & S_H \end{array} \\ \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1N} \\ w_{21} & w_{22} & \cdots & w_{2N} \\ \vdots & \vdots & \vdots & \vdots \\ w_{H1} & w_{H2} & \cdots & w_{HN} \end{bmatrix} \end{array}$$

$$K = \begin{array}{c} \\ H_1 \\ H_2 \\ \vdots \\ H_M \end{array} \begin{array}{c} \begin{array}{cccc} C_1 & C_2 & \cdots & C_H \end{array} \\ \begin{bmatrix} k_{11} & k_{12} & \cdots & k_{1H} \\ k_{21} & k_{22} & \cdots & k_{2H} \\ \vdots & \vdots & \vdots & \vdots \\ k_{M1} & k_{M2} & \cdots & k_{MH} \end{bmatrix} \end{array}$$

$S = (S_1, S_2, \ldots, S_N)$ is the set of services, $C = (C_1, C_2, \ldots, C_H)$ is the finite set of components, and $H = (H_1, H_2, \ldots, H_M)$ is the set of host nodes. Each entry in matrix W and matrix K is assumed to be integer number [0,1]. Each component provides one service, and many components may provide the same service. Each component at most can be deployed onto one host node every software redeployment.

B. Component Replica

In a pervasive system, which consists of many hardware notes, a failure of a hardware node does not mean that the whole system fails. It can still operate if replica of critical software component exists. We use matrix W to explain the component replica. For example, there are a set of services $\{S_1, S_2, S_3\}$, a set of components $\{C_1, C_2, C_3, C_4, C_5\}$, a set of hosts $\{H_1, H_2\}$. C_1 and C_5 both provide the service S_1, while C_2 and C_4 both provide the service S_2, C_3 provides the service S_3. Therefore, if component C_5 is original component, then C_1 is a component replica. The matrix W can be described as follows:

$$W = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

C. Software Component Deployment

Since pervasive systems are likely to be long lived and dynamic, they will demand deployment, and redeployment solutions that are able to adjust to the system's changing execution context continuously [2]. The distribution of software components deployed onto hardware nodes (i.e., a system's software deployment architecture) greatly influence on the system's reliability in the face of components or hosts fail. We also use the above mentioned example, and assume that C_1, C_2 and C_5 are deployed onto H_1, C_3 and C_4 are deployed onto H_2. Then matrix K_{before} can be described as follows:

$$K_{before} = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

If system redeploy, C_3 need to be deployed on H_1, then matrix K_{after} can be described as follows:

$$K_{after} = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Here, if the number of components deployed onto a host changes, we regard it as different deployment architecture. After redeploymentthe number of components deployed on host 1 changes from 3 to 4. Therefore, the change of matrix K reflects the different software component deployment architecture.

IV. SYSTEM RELIABILITY

Reliability is defined as the probability for components or systems to function successfully against the expected environment for a given time. Reliability has long been treated as one of the most performance attributes [11].System reliability estimations are often obtained on the basis of the reliability information of subsystems or components [12].

The distribution of components over host nodes may change after system redeployment. We need to calculate the system reliability of different system deployment architectures.

A. Different Deployment Architecture

In some cases, engineers can not determine completely the non-functional property of a software system or its target environment prior to the system's initial deployment. As a result, the system's initial deployment architecture (i.e., allocation of software components to hardware hosts) may be unsatisfactory with the context of the actual running system. This is of particular concern in embedded and pervasive systems, which are affected by unpredictable movements of target hosts. Therefore, the system's deployment architecture may need to be altered by redeploying some components at runtime.

In this paper, we regard the components that provide the same service as the same type ones. We assume that the components of the same type can never be deployed on the same host. The constraint of the model prohibits this double deployment.

In generally, if a system consists of N hardware hosts and M software components, then the number of deployment architecture is N^M. However, if M software components includes component replica, that is, the number of components which provide service S_1 is M_1, the number of components which provide service S_2 is M_2 ,$\cdots$, the number of components provide service S_l is M_l. Additionally, $M = \sum\limits_{i=1}^{l} M_i$, then the number of different deployment architectures is

$$C_N^{M_1} \cdot C_N^{M_2} \cdot \cdots \cdot C_N^{M_l} = \frac{(N!)^l}{(\prod\limits_{i=1}^{l} (N - M_i)!) \cdot (\prod\limits_{i=1}^{l} M_i!)} \quad (4)$$

B. Component Failure Probability

Many approaches have been used in component reliability prediction. Some approaches assume that the reliability of the individual component in a system is known, or its service importance is known [13]. Some approaches model component behaviors by Markov models and assume that the software architect can provide the transition probabilities between individual components [14]. A few approaches consider the influence of component internal behavior on component reliability. How to calculate component itself reliability is not discussed. In this paper, we assume components of different type have different component failure probabilities. In this case, the components that provide the same service have the same component failure probability.

In our reliability model, the failure of a component is dependent on the host on which it is placed. We assume the failure probability of each host is p_h and component C_i itself failure probability is p_{C_i}. If a host fails, the components deployed on the host also fail. If a host dose not fail ($\bar{p_h} = 1 - p_h$), the failure probability of services depends on the component reliability. Therefore, through application of probability theory (i.e., condition probability and the law of total probability) the following formula for the component reliability can be derived.

$$p(C_i) = p_h + \bar{p_h} \cdot p_{C_i} \quad (5)$$

C. Component Replica and System Reliability

Many factors may impact a pervasive system's ability to function correctly. For example, since the computing platforms (e.g., PDA, celluar phones, etc) that are widely used in pervasive and ad-hoc environments have finite battery lives, a host may “go down”due to battery depletion. Consequently, all the services provided by the components on that host become unavailable to other hosts. But the whole system can not fail, if software services are independent or if replica of critical software components exists. Therefore, the replication of components is used as an improved system reliability strategy.

To compute system reliability, we consider a software system, which consists of a set of services $\{S_1, S_2, ..., S_l\}$. We assume that host failure probability is p_h. The same type components have the same failure probability, and different type components have different failure probability. The failure probability of service S_i is $p(S_i)$. The number of components which provide service S_i is M_i. The failure probability of component which provide service S_i is p_{sc_i}. The failure probability of system is p_{system}. Therefore, System reliability can be calculated as:

$$R_{system} = 1 - p_{system} = 1 - \bigcup_{i=1}^{N} p(S_i) \qquad (6)$$

In order to estimate $p(S_i)$, we discuss two cases:

1) all hosts that provide service S_i do not fail, and all components that provide service S_i fail.

$$p(S_i) = (1 - p_h)^{M_i} \cdot (p_{sc_i})^{M_i} \qquad (7)$$

2)not all hosts that provide service S_i fail. In this case,if components can be deployed onto normally functioned hosts, then the components all fail. The variate j represents the number of failed hosts that provide service S_i. It may be $1, 2, \ldots, M_i$. Then

$$p(S_i) = \sum_{j=1}^{M_i} (p_h)^j \cdot (1 - p_h)^{M_i - j} \cdot (p_{SC_i})^{M_i - j} \qquad (8)$$

Therefore, through application of probability theory, the following formula for system reliability can de derived.

$$\begin{aligned} R_{system} &= 1 - \bigcup_{i=1}^{l} p(S_i) \\ &= 1 - \bigcup_{i=1}^{l} (\sum_{j=1}^{M_i} (p_h)^j \cdot (1 - p_h)^{M_i - j} \cdot (p_{SC_i})^{M_i - j}) \end{aligned} \qquad (9)$$

We can obtain the number of component replica M_i by Matrix K and M. This reliability estimation considers the influence of host failure, component replica and software deployment on system reliability. However, in real environment, there are other factors that affect on system reliability, such as physical link reliability, host available memory, component required memory. These factors will be incorporated in reliability model in further study.

V. EXPERIMENTAL RESULTS

125 samples have been used for implementation as data sets. 100 samples are used for training data and 25 samples are used for testing data. In this model, the first layer has 35 neurons, the second layer has 40 neurons, and the third layer has 5 neurons. Comparative diagram of output of neural network and actual output of the model for training data and testing data have been shown in Fig. 2 and Fig. 3.

ANCDH1 represents the actual number of components deployed on the host 1, ENCDH1 represents the estimated number of components deployed on the host 1 in neural

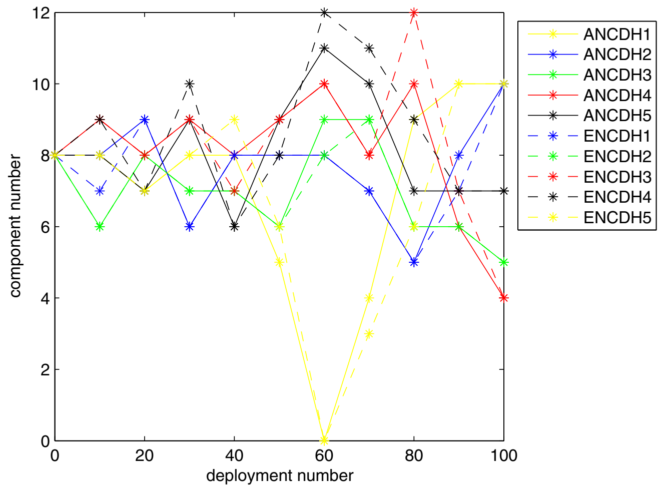

Figure 2. Comparative diagram of output of neural networks and actual output for the model for training data.

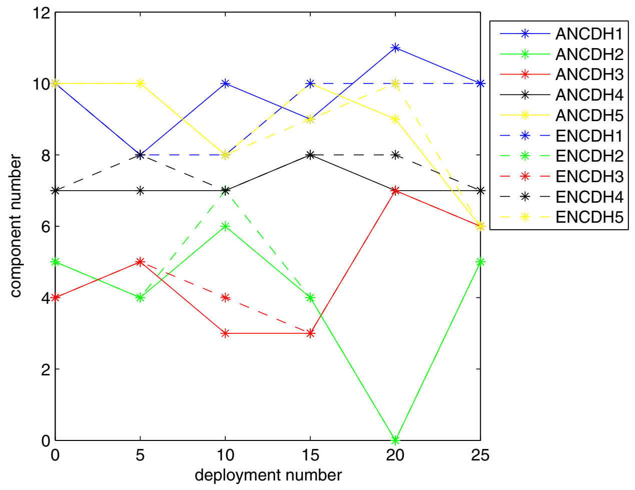

Figure 3. Comparative diagram of output of neural networks and actual output of the model for testing data.

networks and so on. In Fig. 2, because host 5 fails, the components deployed on host 5 all fail, and the number of components which can provide services is zero. After host 5 recovered from failure, we can redeploy components on host 5 again.

VI. CONCLUSIONS

Many researches have shown that a software system's deployment architecture can have a significant effect on the system's reliability. This is evident in pervasive systems, which are often long-lived, and execute in highly heterogeneous and unpredictable computing environments. Therefore, our reliability estimation model includes software deployment. This paper has three contributions. First, we put forward a new comprehensive of SA based software deployment reliability estimation in neural networks. Second, we attempts to define different software component deploymentand explain software component replica and software component redeployment in neural networks. Third, we calculate system reliability considering component replica during software deployment.

ACKNOWLEDGMENT

The work described in this paper is supported by Hi-Tech Research and Development Program of China (Project No. 2008AA01A201), National High-Tech Research and Development Plan of China (Project No. 2006AA01A103), National Nature Science Funds of China (Project No. 60503015).

REFERENCES

[1] C.Seo, S.Malek, G.Edwards, D. Popescu, et al., "Exploring the Role of Software Architecture in Dynamic and Fault Tolerant Pervasive Systems", Proc. IEEE Symp. Software Engineering for Pervasive Computing (SEPCASE'07), IEEE Press, May 2007, pp.9-15, doi: 10.1109/SEPCASE.2007.6.

[2] N. Medvidovic, S.Malek, "Software Deployment Architecture and Quality and Quality-of-Service in Pervasive Environments", Proc. IEEE Symp. Engineering of Software Services for Pervasive Environments (ESSPE'07), ACM Press, Sept. 2007, pp.47-51, Dubrovnik, Croatia.

[3] Z. hong-bin, J. Zhi-xin, "Research in Reliability Modeling of WEDM Based on Neural Network", Proc. IEEE Symp. Measuring Technology and Mechatronics Automation (ICMTMA'09), IEEE Press. April 2009, pp.277-280, doi 10.1109/ ICMTMA.2009.20.

[4] J.Zheng, "Predicting Software Reliability with Network Ensembles", Science, vol.36. March. 2009, pp.2116-2122, doi: 10.1016/j.eswa.2007.12.09.

[5] Y.S.Su, C.Y.Huang, "Neural-network-based approaches for software reliability estimation using weighted combinational models", Science, vol.80. April 2007, pp.606-615, doi:10.1016/j.eswa.2007.12.029

[6] M. Harirforoush, M.Seyyedi, N. Mirzaee, "the Evaluation of Reliability Based on the Software Architecture in Neural Networks", Proc. IEEE Symp. Software Engineering Advances (ICSEA'08). Oct.2008, pp.19-24, doi 10.1109/ ICSEA. 2008.56

[7] I. Krka, L. Cheung, G. Edwards, L. Golubchik, N. Medvidovic, "Architecture-based Software Reliability Estimation: Problem Space, Challenges, and Strategies",In the Proceedings of the DSN Workshop on Architecting Dependable Systems(WADS'08). Jun 2008, pp. 32-38, Anchorage,AK.

[8] R. Thomas, "Architectural Styles and the Design of Network-based Software Architectures", Ph.D.Dissertation, University of California. Irvine, 2000.

[9] C. Vo, T. Torabi, "A Framework of Over the Air Provider-initiated Software Deployment on Mobile Devices", Proc. IEEE Symp. Software Engineering (ASWEC'08), IEEE Press. March 2008, pp.633-638, doi:10.1109/ASWEC.2008.31.

[10] A. Dearie, "Software Deployment, Past, Present and Future", Proc. IEEE Symp. Future of Software Engineering (FOSE'07), IEEE Press. May 2007, pp.269-284, doi:10.1109/POSE.2007.20.

[11] W.Zheng, X.Liyang, "Dynamic Reliability Model of Components Under Random Load", Proc. IEEE Symp. Reliability, IEEE Press. pp.474-479, doi:10.1109/ TR.2008. 928184

[12] T. Jin, "Hierarchical Variance Decomposition of System Reliability Estimates with Duplicated Components", Proc. IEEE Symp. Reliability, IEEE Press. pp.564-573 ,doi:10.1109/TR.2008.2006643.

[13] L.Cheung, R.Roshandel, N. Medvidovic, L.Golubchik, "Early Prediction of Software Component Reliability", Proc. IEEE Symp. Software Engineering (ICSE'08), IEEE Press. May 2008, pp.111-120, doi:10.1145/1368088.1368104.

[14] H. Koziolek, "Parameter Dependencies for Component Reliability Specifications",Science, vol.253. Oct. 2009,pp.23-38, doi:10.1016/j.entcs.2009.09.026.