SA Based Software Deployment Reliability Estimation Considering Component Reliability of Exponential Distribution

Xihong Su
School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China
Email: suxihong07@gmail.com

Hongwei Liu, Zhibo Wu, Xiaozong Yang, Decheng Zuo
School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China
Email: {lhw, wzb, xzy, zdc}@ftcl.hit.edu.cn

Abstract—Although many approaches for architecture-based reliability estimation exist, these approaches are typically limited to certain classes or exclusively concentrate on software reliability, neglecting the influence of hardware resources, component reliability, component replica and software deployment. In this paper, a reliability estimation model based on software architecture (SA) is presented. This model incorporates the influence of software deployment, component reliability and component replica. Component lifetimes can be modeled by exponential distribution. The approach of calculating system reliability considering component replica and component reliability is proposed. The influences of different deployment architectures on component reliabilities and system reliability are investigated. The improvement of system reliability by redeployment or component replica is discussed.

Index Terms—software architecture, software deployment, component reliability, exponential distribution, component replica, system redeployment

I. INTRODUCTION

The past few decades have witnessed an unrelenting pattern of growth in 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, decrease in its cost, emergency of wireless ad hoc networks, proliferation of sensors and handheld computing device, etc. Studies have shown that a promising approach to resolve the challenges of developing large scale software system is to employ the principles of software architectures [1,2]. Software architecture provides abstractions for representing the structure, behavior, and key properties of a software system. They are described in terms of software components (computational elements), connectors (interaction elements), and their configurations [3, 4].

Many approaches have begun to predict reliability at the level of architectural models, or at least in terms of high-level system structure [5-12]. Firstly, these researchers acknowledge that reliabilities of components have a significant impact on system reliability, but they almost invariably assume that the reliabilities of the components in a system are known. The few researchers consider component-level reliability [8, 10], assume that the reliabilities of a given component elements, such as its services, are known. Secondly, these approaches are typically limited to certain failure classes or exclusively concentrate on software reliability, neglecting the influence of hardware resources, software deployment and component replica. In a given context, some of these deployment configurations are obviously more effective than others in terms of reliability. Additionally, if replica of critical software components exists, failure of one host node does not mean that the whole system fails. Therefore, we propose a new SA based reliability estimation model, incorporating the influence of software deployment, component replica and component reliability.

The rest of this paper is organized as follows. Section II describes system deployment architecture. Section III investigates component replica and component reliability. Section IV proposes an approach of calculating system reliability. Section V gives the experiments. Section VI presents the conclusions and directions of future study.

II. SYSTEM DEPLOYMENT ARCHITECTURE

In this section, we present an overview of whole structure of system deployment architecture. System
deployment architecture is the allocation of the system software components on its hardware host nodes [13]. We also introduce the frequency matrix \( CC \) of interaction among software components.

### A. Whole Structure

The basic entities of SA based software deployment reliability estimation model include host nodes, software components and services. In details, the model consists of:

1) a set \( H \) of host nodes, \( H = \{H_1, H_2, \ldots, H_n\} \), which represents the host nodes of a system.

2) a set \( C \) of components, \( C = \{C_1, C_2, \ldots, C_n\} \). Each component may have multiple component replica.

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

All components of set \( C \) should be deployed on \( m \) host nodes. Matrix \( CH \) describes how to deploy components on host nodes.

\[
CH = \begin{bmatrix}
C_1 & C_2 & \cdots & C_n \\
H_1 & [c_{1,1} & c_{1,2} & \cdots & c_{1,n}] \\
H_2 & [c_{2,1} & c_{2,2} & \cdots & c_{2,n}] \\
\vdots & \vdots & \ddots & \vdots \\
H_m & [c_{m,1} & c_{m,2} & \cdots & c_{m,n}] 
\end{bmatrix}
\]

Each entry \( c_{i,j} \) in matrix \( CH \) may be 1 or 0:

- \( c_{i,j} = 1 \), if component \( C_i \) is deployed on host node \( H_j \)
- \( c_{i,j} = 0 \), if component \( C_i \) is not deployed on host node \( H_j \)

For a system comprising \( m \) host nodes and \( n \) software components, the number of system deployment architectures is \( m^n \). In general, determining the system deployment architecture that will maximize its reliability for the given target environment is an exponential complexity problem.

### B. Frequency of Interaction among Software Components

Matrix \( CC \) describes frequency of interaction among software components, \( C = \{C_1, C_2, \ldots, C_n\} \) is a set of components. Each entry \( c_{i,j} \) in matrix \( CC \) represents the frequency of interaction between software component \( C_i \) and \( C_j \). \( c_{i,j} \) is an integer number in \([a, b]\).

The values of \( a \) and \( b \) depend on concrete system.

\[
CC = \begin{bmatrix}
CC_{1,1} & CC_{1,2} & \cdots & CC_{1,n} \\
CC_{2,1} & CC_{2,2} & \cdots & CC_{2,n} \\
\vdots & \vdots & \ddots & \vdots \\
CC_{n,1} & CC_{n,2} & \cdots & CC_{n,n} 
\end{bmatrix}
\]

### III. Component Reliability and Component Replica

System reliability can be appropriately evaluated through component reliability. When there is not enough failure data, it is very common to make certain reasonable assumptions. A common assumption is that the system lifetime follows an exponential distribution. In this section, we suppose that all components follow exponential distribution and particularly depend on frequency of interaction among software components, as shown in formula (1). \( RC_i \) is the reliability of component \( C_i \). Component failure rate \( \lambda C_i \) is a function of interaction frequency \( fC_i \) of component \( C_i \).

\[
RC_i = e^{-\lambda C_i fC_i}, i = 1, 2, \ldots, n
\]

For a system consisting of \( m \) host nodes and \( n \) software components, if component \( C_i \) has been identified as a candidate for replication, \( C_i \) should be replicated \( m-1 \) times. That is, \( C_i \) has \( m-1 \) component replicas or \( m \) components provide the same service. With redundant copies, a replicated component can continue to provide a service in spite of the failure of some of its copies.

\[
\lambda C_i = \begin{cases} 
\lambda C_i^*, & \text{if } C_i \text{ is replicated } \\
\lambda C_i^*, & \text{otherwise } 
\end{cases}, i = 1, 2, \ldots, n
\]

\[
\lambda C_i' = fC_i \times \rho_c + \lambda_0 + f_{\text{other}} \times \rho_{\text{other}}
\]

\[
\rho_c \text{ frequency criticality of interaction among software components.}
\]

\[
\rho_{\text{other}} \text{ criticality of other factors, it is a real number in } [-1, 1].
\]

\[
\lambda_0 \text{ initial value of component failure probability.}
\]

\[
fC_i \text{ frequency sum of interaction between component } C_i \text{ and other components deployed on other host nodes}
\]

The value of \( fC_i \) can be obtained by matrix \( CC \) in formula (5). \( H(C) \) describes the host node that component \( C_i \) is deployed on.

\[
fC_i = \sum_{C_j \in H(C)} cc_{i,j}
\]

### IV. System Reliability

It is often infeasible or difficult to directly estimate complex system reliability through large sample system-level test. Such difficulties may arise when the system-level test is costly or leads to destruction of the system itself. Nevertheless, system reliability can be appropriately evaluated through the component reliability information [14]. In this section, we suppose that the main sources of system failure are software component
failure and host node failure. Therefore, system reliability is calculated in formula (6).

\[
R_{\text{system}} = 1 - P_{\text{system}} = 1 - \left( \sum_{i=1}^{n} (1 - p_{hi}) - \sum_{i=1}^{m} p_{ci} \right)
\]

\[p_{ci} = 1 - \prod_{j \in T, j \neq i} (1 - p_{hi}) \times e^{-\lambda_{ci} t}\]

\[R_{\text{system}} \quad \text{system reliability}
\]
\[P_{\text{system}} \quad \text{failure probability of system}
\]
\[H(C_i) \quad \text{the host node that } C_i \text{ is deployed on}
\]
\[p_{hi} \quad \text{failure probability of components deployed on host node } H_i
\]
\[p_{ci} \quad \text{failure probability of host node } H_i
\]

V. EXPERIMENTS

In this section, four experiments are based on such a system consisting of eight original software components and four host nodes. Each original software component can provide a different service. The inputs of the experiments include randomly generated frequency matrix of interaction among software components and failure probabilities of four host nodes. Additionally, four experiments are based on three different deployment architectures.

A. Inputs

Matrix \(CC\) is randomly generated frequency matrix of interaction among software components. Each entry in matrix \(CC\) is an integer number \([0, 7]\). We can calculate the value of \(f_{C_i}\) for component \(C_i\) on the basis of matrix \(CC\).

\[
CC = \begin{bmatrix}
0 & 3 & 0 & 0 & 0 & 0 & 7 & 1 \\
3 & 0 & 7 & 0 & 2 & 2 & 0 & 0 \\
0 & 7 & 0 & 6 & 0 & 5 & 0 & 0 \\
0 & 0 & 6 & 0 & 3 & 0 & 0 & 7 \\
0 & 2 & 0 & 3 & 0 & 7 & 0 & 4 \\
0 & 2 & 5 & 0 & 7 & 0 & 0 & 3 \\
7 & 0 & 0 & 0 & 0 & 3 & 0 & 0 \\
1 & 0 & 0 & 7 & 4 & 0 & 0 & 0
\end{bmatrix}
\]

Failure probabilities of four host nodes are real numbers in \([0, 0.02]\). \(p_{hi}\) represents the failure probability of host node \(H_i\), \(p_{h1} = 0.0093\), \(p_{h2} = 0.0195\), \(p_{h3} = 0.0069\), \(p_{h4} = 0.0101\).

B. Deployment Architecture and Frequency of Software Component Interaction

These eight components should be deployed on four host nodes. On the basis of frequency matrix of interaction among software components, we obtain three typical deployment architectures.

1) Deployment architecture

We use matrix \(CH_1\) to describe the first deployment architecture. \(CH_1\) shows that \(C_4, C_5, C_8\) are deployed on host node \(H_1\), \(C_2, C_7\) are deployed on \(H_2\). \(C_1, C_6\) are deployed on \(H_3\), \(C_3\) is deployed on \(H_4\).

\[
CH_1 = \begin{bmatrix}
0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\]

We use matrix \(CH_2\) to describe the second deployment architecture. Matrix \(CH_2\) shows that \(C_1\) is deployed on host node \(H_1\), \(C_2, C_6\) are deployed on \(H_2\). \(C_3, C_7\) are deployed on \(H_3\), \(C_4, C_5\) are deployed on \(H_4\).

\[
CH_2 = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 1 & 1 & 1 & 0 & 0 & 0 & 0
\end{bmatrix}
\]

We use matrix \(CH_3\) to describe the third deployment architecture. \(CH_3\) shows that \(C_1, C_3\) are deployed on host node \(H_1\), \(C_2, C_7\) are deployed on \(H_2\). \(C_4, C_6\) are deployed on \(H_3\), \(C_5\) is deployed on \(H_4\).

\[
CH_3 = \begin{bmatrix}
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}
\]

2) Frequency of component interaction

On the basis of matrix \(CC\), we calculate \(f_{C_i}\) of each component \(C_i\) in formula (5). \(F_1\) represents the set of \(f_{C_i}\) of the first deployment architecture.

\[
F_1 = [8 \ 11 \ 18 \ 6 \ 9 \ 14 \ 7 \ 1]
\]

Therefore, it is easy to know that \(f_{C_1} = 18\) and \(f_{C_4} = 1\) in the first deployment architecture.

Similarly, \(F_2\) represents the set of \(f_{C_i}\) of the second deployment architecture. \(F_2\) represents the set of \(f_{C_i}\) of the third deployment architecture.

\[
F_2 = [4 \ 7 \ 5 \ 10 \ 9 \ 10 \ 3 \ 12]
\]
\[
F_3 = [11 \ 14 \ 18 \ 16 \ 17 \ 10 \ 12]
\]

C. Experiments

The system includes eight software components. Each software component can provide a different service. The values of relevant parameters are \(f_{other} = 0\) and \(P_{other} = 0\).
1) Experiment one

In this experiment, \( \lambda_0 = 0.0004 \) and \( \rho_c = 0.00001 \). System reliability of three different deployment architectures can be shown in Fig.1. EFDSR represents system reliability of the first deployment architecture. ESDSR represents system reliability of the second deployment architecture. ETDSR represents system reliability of the third deployment architecture.

As seen in Fig. 1, with the increasing of system run-time, system reliability of the second deployment architecture is obviously highest among the three. The impact of the first deployment architecture on system reliability is similar to the third one during [0, 70] hours.

A detailed analysis of component reliabilities of the three different deployment architectures is shown in Tab.I.

Table I. The Experimental Results for Component Reliabilities of Three Deployment Architectures

<table>
<thead>
<tr>
<th>Component</th>
<th>First deployment architecture</th>
<th>Second deployment architecture</th>
<th>Third deployment architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>( C_2 )</td>
<td>RM</td>
<td>RH</td>
<td>RL</td>
</tr>
<tr>
<td>( C_4 )</td>
<td>RH</td>
<td>RM</td>
<td>RL</td>
</tr>
<tr>
<td>( C_5 )</td>
<td>RS</td>
<td>RS</td>
<td>RM</td>
</tr>
<tr>
<td>( C_8 )</td>
<td>RH</td>
<td>RS</td>
<td>RS</td>
</tr>
</tbody>
</table>

RH represents the highest of component reliability among the three deployment architectures. RM represents the medium value of component reliability among the three deployment architectures. RL represents the lowest of component reliability among the three deployment architectures. RS describes that there is a negligible difference of component reliability between two deployment architectures.

As seen in Tab.I, component reliabilities of \( C_2 \) and \( C_4 \) are obviously different. There is a negligible difference of component reliabilities of \( C_3 \) and \( C_8 \) between two deployment architectures. Therefore, if \( C_2 \) is the most important component, we will deploy components on host nodes according to the second deployment architecture.

2) Experiment two

In this experiment, the first deployment architecture is the basis of the experiment. The value of \( \rho_c \) may be different. These values are \( \rho_c = 0.00001 \), \( \rho_c = 0.00002 \) and \( \rho_c = 0.00004 \). The influence of the values of \( \rho_c \) on system reliability can be shown in Fig. 2. With the higher value of \( \rho_c \), system reliability becomes the lower. With the increasing of system run-time, the difference of system reliabilities with different values of \( \rho_c \) is more apparent.

Figure 2. System reliability of different value of \( \rho_c \)

The influence of the values of \( \rho_c \) on reliability of component \( C_3 \) is shown in Fig. 3.

Figure 3. Component \( C_3 \) reliability of different values of \( \rho_c \)

With the higher value of \( \rho_c \), reliability of component \( C_3 \) becomes the lower. With the increasing of system run-time, the difference of reliability of
component $C_3$ of different values of $\rho_c$ is more apparent.

3) Experiment three

In this experiment, we investigate the impact of system redeployment on system reliability. The value of $\rho_c$ is 0.00004. Before thirty hours, system runs the third deployment architecture. At thirty hours, system reliability is less than 0.8. After thirty hours, we improve system reliability by redeployment, as shown in Fig. 4.

After thirty hours, system begins to redeploy and run the second deployment architecture. In Fig. 4, BDA represents system reliability of third deployment architecture during $[0, 30]$ hours. BDE represents system reliability without system redeployment during $[30, 80]$ hours. ADA represents system reliability with system redeployment during $[30, 80]$ hours.

After system redeployment, component reliability has changed. Component reliability can be calculated in formula (8).

$$RC = \begin{cases} e^{-(\lambda + \rho_c) t}, & 0 \leq t \leq 30 \\ e^{-(\lambda + \rho_c) t}, & 30 < t \end{cases} , i = 1, 2, \ldots, 8$$

$bfC_i$ represents interaction frequency of component $C_i$ before redeployment. $afC_i$ represents interaction frequency of component $C_i$ after redeployment.

4) Experiment four

In this experiment, we investigate the difference of system reliability by replicating components and system redeployment. The value of $\rho_c$ is 0.00004. When system reliability drops to some value, system reliability need to be improved by system redeployment or replicating components, as shown in Fig. 5.

After system redeployment, component reliability has changed. Component reliability can be calculated in formula (8).

$$RC = \begin{cases} e^{-(\lambda + \rho_c) t}, & 0 \leq t \leq 30 \\ e^{-(\lambda + \rho_c) t}, & 30 < t \end{cases} , i = 1, 2, \ldots, 8$$

$bfC_i$ represents interaction frequency of component $C_i$ before redeployment. $afC_i$ represents interaction frequency of component $C_i$ after redeployment.

As seen in Fig. 5, system reliability with replicating multiple components is higher than system redeployment. System reliability with replication is higher than replicating single component. System reliability will be improved by replicating components and system redeployment. However, if replicating components, we need to take into consideration the computational resources required and those available at each host node; if system redeployment, we need to consider the time to redeploy and the cost of system redeployment.

VI. CONCLUSIONS AND FUTURE RESEARCH

It is very important to estimate system reliability based on software architecture. Reliability is one of the most critical extra-functional properties of a software system. This paper analyzes the defects of existing architecture-level reliability estimation approaches, and proposes a novel system reliability estimation model incorporating the influence of component reliability, software deployment and component replica. Different deployment architectures have a significant influence on system reliability and component reliability. We present how to calculate system reliability and component reliability. We present the approaches of improving system reliability and the conditions of applying these approaches. In future research, system reliability estimation model based on SA will include other influence factors, such as software architectural styles, component replica strategies and so on.

ACKNOWLEDGMENT

The authors wish to thank Tao Liu for his kindly help with checking.
REFERENCES


Xihong Su was born in 1981 in China. She received the M.S. degree in school of computer science and technology from Harbin Institute of Technology, Harbin, China, in 2007. She is currently pursuing the Ph.D. degree at Harbin Institute of Technology. Her research interests include software architecture, software deployment and system reliability estimation.

Hongwei Liu was born in 1971 in China. He received the Ph.D. degree in school of computer science and technology from the Harbin Institute of Technology, PR China, in 2004. He is a professor of the School of Computer Science and Technology at the Harbin Institute of Technology. His main research interests are fault-tolerant computing, software reliability evaluation, software testing. He is a senior member of the CCF.

Zhibo Wu was born in 1947 in China. He received the Ph.D. degree in school of computer science and technology from Harbin Institute of Technology, Harbin, China, in 1980. He is a professor and Ph.D. supervisor in school of computer science and technology at Harbin Institute of Technology. His research interests include fault-tolerant computing, mobile computing and system reliability estimation. He is a senior member of the CCF.

Xiaozong Yang was born in 1939 in China. He received his B.S. degree in School of Computer and Science from Harbin Institute of Technology, in 1964. He is currently a professor and Ph.D. supervisor in School of Computer Science and Technology at Harbin Institute of Technology (CS HIT), the computer science institute director, the wearable computing Engineering and Research center Director of CS, HIT, vice director of fault tolerant computing committee under China computer federation and the head of wearable computing group under china computer Federation. His main research interests include computing architecture, fault tolerant computing. He has implemented many research projects and has received progress in his research work, and won about 10 awards of ministries. He published 2 books and about 100 papers, 10 PHD students graduated from his group, and was appointed as conference chair, session chair of China conference and international conference.

Decheng Zuo was born in 1972 in China. He received the Ph.D. degree in School of Computer and Science from Harbin Institute of Technology, in 2001. He is a professor of the School of Computer Science and Technology at the Harbin Institute of Technology. His main research interests include fault-tolerant computing, mobile computing and system reliability estimation. He is a senior member of the CCF.