Reliability Analysis Of SA based Software Deployment with Consideration of System Deployment

Xihong Su , Zhibo Wu , Hongwei Liu , Xiaozong Yang & Xiao-Zong Yang

Intelligent Automation and Soft Computing, Vol. 17, No. 6, pp. 749-758, 2011
Copyright © 2011, TSI® Press
Printed in the USA. All rights reserved

RELIABILITY ANALYSIS OF SA BASED SOFTWARE DEPLOYMENT WITH CONSIDERATION OF SYSTEM DEPLOYMENT

XIHONG SU, ZHIBO WU, HONGWEI LIU, XIAOZONG YANG, DECHENG ZUO
School of Computer Science and Technology
Harbin Institute of Technology
Harbin, 150001, China

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 mobile environments. There are two important facets related to this domain: software deployment and reliability. SA based software deployment models help to analyze reliability of system deployments. Though there exist many approaches for architecture-based reliability estimation, little work has been done in incorporating the influence of system deployment and hardware resources. In this paper, a new approach of estimating system reliability at architectural level is proposed. The approach incorporates the influence of system deployment and hardware resources. Additionally, there are many factors influencing system deployment, such as possible restrictions on component location. In order to make fully use of these factors, the multi-dimension factors on system deployment are translated into degree matrices of component dependence and host node dependence. An approximate algorithm, Greedy_Deploy algorithm based on greedy algorithm is presented. On the basis of matrices of component dependence and host node dependence, the Greedy_Deploy algorithm is used to deploy software components on host nodes. In the evaluation, Greedy_Deploy algorithm shows better performance than adaptive greedy algorithm.

Key Words: software architecture, software deployment, system reliability, component, host node

1. INTRODUCTION

In the past a few decades, software systems are continuously growing in size and complexity. This pattern is further evident in an emerging class of distributed, decentralized and embedded 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[1]. Software engineering researchers and practitioners have successfully dealt with the increasing complexity of software systems by employing the principles of software architecture [2]. Software architecture is a collection of models that capture a software system principle design decisions in the form of components, connectors, and configurations [3].

In the recent decades, architecture-based reliability analysis has received a great deal of attention [4-13]. First, these models have some assumptions. For example, researchers assume the reliabilities of individual components in a system are known [4-6], the behaviors of components satisfy Markov property [7], architecture style is pipeline [8], and so on. Second, most of these reliability estimation approaches are typically limited to certain classes or exclusively concentrates on software reliability, neglecting the influence of system deployment and hardware resources [9-13]. System deployment architecture is the allocation of system software components on its host nodes. SA based software deployment models help to analyze system reliability of different deployments. A SA based software deployment reliability estimation model incorporating the influence of system deployment and hardware resources is proposed. The reliabilities of individual components are also assumed to know in the system. Additionally, there are many factors influencing system deployment, such as software architectural styles, possible restrictions on component location, component size, available memory on host nodes and so on. The multi-dimension influences on system deployment are translated into degree matrices of component dependence and host node dependence in our model.

This paper is organized as follows. Section 2 explains the related concepts. Section 3 provides a brief description of problem statement, and presents an approximate algorithm, Greedy_Deploy algorithm. Section 4 executes Greedy_Deploy algorithm to deploy 14 components on 4 host nodes, and analyzes the experimental results. Finally, the last section concludes the paper.

2. TERMINOLOGY AND SCOPE

2.1 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, reconfiguration and un-installation of the software [14].

SA based software deployment provides an architectural description of software and hardware model during deployment stage. It helps to analyze quality attributes of different deployments and select a reasonable deployment [15].

2.2 Component Dependence

Given an undirected graph $G=(C,E)$, C describes the set of system software components, $C=\{C_1,C_2,\ldots,C_n\}$, $|C|=n$, E describes the set of interactions among software components, $E=\{E_1, E_2, \ldots, E_s\}$, $|E|=s$.

Definition 1 Component dependence: $\forall C_i,C_j \in C$, if $(C_i,C_j)\in E$ or $\exists$ $m_1, m_2,\ldots, m_k$, $k \le n\text{-}2$, $(C_i,C_{m_1})\in E,(C_{m_1},C_{m_2})\in E$,…, $(C_{m_k},C_j)\in E$, the relationship between component C_i and C_j is called as component dependence.

Definition 2 Degree of component dependence: $\forall C_i,C_j \in C$, $cc_{i,j}$ describes the degree of component C_i depends on C_j . $cc_{i,j}$ is a real number [0,1]. $\forall C_i \in C$, $cc_{i,i}=0$.

2.3 Host Node Dependence

Given an undirected graph $G=(H,Q)$, H describes the set of host nodes in the system, $H=\{H_1,H_2,\ldots,H_m\}$, $|H|=m$, Q describes the set of physical links among host nodes, $Q=\{Q_1,Q_2,\ldots,Q_p\}$, $|Q|=p$.

Definition 3 Host node dependence: $\forall H_i,H_j \in H$, if $(H_i,H_j)\in Q$ or $\exists$ $x_1, x_2, \ldots, x_z$, $z \le m\text{-}2$, $(H_i,H_{x_1})\in Q,(H_{x_1},H_{x_2})\in Q$,…, $(H_{x_z},H_j)\in Q$, the relationship between host node H_i and H_j is called as host node dependence.

Definition 4 Degree of host node dependence: $\forall H_i, H_j \in H$, $hh_{i,j}$ describes the degree of host node H_i depends on H_j. $hh_{i,j}$ is a real number [0,1]. $\forall H_i \in H$, $hh_{i,i}=0$.

2.4 System Reliability

System reliability estimations are often obtained on the basis of the reliability information of subsystems or components [16]. In this paper, component failure and host node failure are regarded as the main sources of the degradation of system reliability. System reliability can be calculated in formula (1). R_s is the system reliability, m is the number of host nodes, w is the number of components deployed on H_i, psh_j is the failure probability of C_j and ph_i is the failure probability of H_i

$$R_s = 1 - \bigcup_{i=1}^{m} ph_i - \bigcup_{i=1}^{m} (1 - \prod_{j=1}^{w} (psh_j \times (1 - ph_i))) \tag{1}$$

3. PROBLEM STATEMENT AND ALGORITHM

3.1 Problem Statement

In the face of host node failure and component failure, the allocation of system software components on host nodes greatly influences system reliability. There are many factors influencing system deployment, such as available memory on each host node, component size, software architectural styles and so on. Determining the system deployment architecture that maximize system reliability is very difficult for the given target environment. Exact algorithm tries every possible deployment and finds at least one optimal deployment. In general, the complexity of exact algorithm is $O(m^n)$ (m is the number of host nodes, n is the number of software components). Therefore, it is very important to design approximate algorithms to deploy components on host nodes.

3.2 Greedy_Deploy Algorithm

A greedy algorithm always makes the choice that looks best at the moment. That is, it makes a locally optimal choice in the hope that this choice will lead to a globally optimal solution [17]. In this section, on the basis of greedy algorithm, the Greedy_Deploy algorithm is presented, as shown in Figure 1. F is the frequency matrix of interaction among software components.

```
Greedy_Deploy algorithm
    {
      1. Generate degree matrix CC of component dependence, CC=MulTranMa();
      2. Divide n components into m component subsets based on CC, ComSub=Div(n,m,CC);
      3. Calculate dependence degree among component subsets based on F, MDC=Dccs(F, ComSub);
      4. Deploy m component subsets on m host nodes, Deploy(MDC,HH);
    }
```

Figure 1. Greedy_Deploy algorithm

```
MulTranMa()
   {
     1. List relevant factors influencing system deployment;
     2. Set importance degree for each relevant factor;
     3. switch (relationship of these factors)
          {
            case 1  : LinearMethod();
            case 2  : LogarithmMethod();
            case 3  : ExponentMethod();
            default : OtherMethod();
          }
     4. Obtain original degree matrix OCC of component dependence;
     5. CC=Normalize(OCC);
   }
```

Figure 2. MulTranMa method

As seen in Figure 2, degree of component dependence may be a function of these relevant factors. How to combine these factors depends on the concrete system. The function may be linear function, exponential function, logarithm function and other function relationship. Then, degree of component dependence is normalized to be a real number in [0, 1].

In Figure 3, C is the set of software components, $C=\{C_1,C_2,\ldots,C_n\}$. RemainCom is the set of remaining software components. ComSub is the array of component subsets. ComSub[i] describes the *i*th component subset.

```
Div(n,m,CC)
    {
      RemainCom=C;
      for (i=1;i<=m;i++)
         {
           Find the max value from CC, for example, $CC_{ab}$;
           Delete row a,b and column a,b from CC;
           RemainCom=RemainCom-$\{C_a\}$-$\{C_b\}$;
           ComSub[i]=$\{C_a , C_b\}$;
         }
        while ($\text{RemainCom} \neq \phi$)
         {
           Choose the max value $CC_{ed}$ from CC, $C_e \in \text{ComSub[i]}, C_d \in \text{RemainCom}$
           RemainCom=RemainCom-$\{C_d\}$;
           Delete row d and column d from CC;
           ComSub[i]= ComSub[i] □ $C_d$;
         }
    }
```

Figure 3. Div method

In Figure 4, CH (ComSub[i]) describes the set of candidate host nodes that ComSub[i] may be deployed on. H is the set of host nodes, $H=\{H_1,H_2,\ldots,H_m\}$. RH is the set of remaining host nodes. MDC is the degree matrix of component dependence among component subsets. HH is the dependence degree among host nodes.

```
Deploy(MDC,HH)
 {
   1. Initialization.RH=H, $\forall i$, CH(ComSub[i])=$\phi$ , SumCS=$\cup$ ComSub[i] .
   2. Choose the max value from MDC, for example, $mdc_{a,b}$
      and choose the max value from HH, for example, $hh_{e,f}$
              CH(ComSub[a])=CH(ComSub[a])$\cup$ {$H_e$, $H_f$}
              CH(ComSub[b])=CH(ComSub[b])$\cup$ {$H_e$, $H_f$}
   3. for ComSub[i] $\in$ SumCS
      Find the duplicate host nodes in CH (ComSub[i]), for example, $H_d$, Del(ComSub[i], $H_d$);
   4. for ComSub[y] $\in$ SumCS
      if (($\exists H_z \in$ CH(ComSub[y])) && ($H_z \notin$ RH) && ($\exists H_q \in$ CH(ComSub[y])) &&($H_q \in$ RH))
          Del(ComSub[y], $H_q$)
   5. Loop 2 to 4 until all components have been deployed on host nodes.
 }
```

Figure 4. Deploy method

In Figure 5, $f_{i,j}$ is the frequency of interaction between component C_i and C_j . In Figure 6, Deployment [ComSub[i]] represents the host node that ComSub[i] has been deployed on.

```
Dccs(F,ComSub)
 {
  for (k=1;k<=m;k++)
    for (p=1;p<=m;p++)
      {
        for ( $\forall C_q \in$ ComSub[k] && $\forall C_g \in$ ComSub[p] )
              temp[p]=temp[p]+$f_{q,g}$
          sum[k]=sum[k]+temp[p];
            $mdc_{k,p}$=temp[p]/sum[k]   ;
      }
 }
```

Figure 5. Dccs method

```
Del(ComSub[x],Hy)
  {
    SumCS=SumCS-ComSub[x];
    Deployment[ComSub[x]]=Hy;
    Delete row x and column x from MDC;
    Delete row y and column y from HH;
    RH=RH-{Hy};
  }
```

Figure 6. Del method

4. EXPERIMENT

The experimental system consists of 4 host nodes and 14 software components. Experimental inputs include failure probabilities of components, failure probabilities of host nodes, factors influencing system deployment, degree matrix of host node dependence and the frequency matrix of interaction among software components. Then the Greedy_Deploy algorithm is used to deploy 14 components on 4 host nodes.

4.1 Inputs

The factors influencing system deployment include component size, available memory on each host node, required memory for each component, possible restrictions on component location, software architectural style, logical links among components and so on.

psh_i represents failure probability of component C_i, as shown in Table I.

Table I. Failure probability of component

Parameter	psh_1	psh_2	psh_3	psh_4	psh_5	psh_6	psh_7
Value	0.0004	0.0011	0.0007	0.0009	0.0013	0.0052	0.0021
Parameter	psh_8	psh_9	psh_{10}	psh_{11}	psh_{12}	psh_{13}	psh_{14}
Value	0.0108	0.0012	0.002	0.0078	0.0019	0.0184	0.013

ph_i represents failure probability of host node H_i, as shown in Table II.

Table II. Failure probability of host node

Parameter	ph_1	ph_2	ph_3	ph_4
Value	0.0075	0.0038	0.0019	0.0003

Matrix F describes the frequency of interaction among components. Each entry $f_{i,j}$ is an integer number in [0, 7].

$$F=\begin{bmatrix}
0&0&0&0&2&0&0&7&0&0&0&0&0&4\\
0&0&0&6&0&7&0&0&0&0&0&0&3&0\\
0&0&0&6&7&0&0&0&0&0&0&0&7&4\\
0&6&6&0&0&0&0&0&0&0&0&4&0&0\\
2&0&7&0&0&0&1&0&0&0&0&0&0&0\\
0&7&0&0&0&0&0&0&0&7&4&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0&0&6\\
7&0&0&0&0&0&0&0&5&6&7&0&0&2\\
0&0&0&0&0&0&0&5&0&0&0&5&0&0\\
0&0&0&0&0&7&0&6&0&0&4&0&4&0\\
0&0&0&0&0&4&0&7&0&4&0&0&0&0\\
0&0&0&4&0&0&0&0&5&0&0&0&3&0\\
0&3&7&0&0&0&0&0&0&4&0&3&0&0\\
4&0&4&0&0&0&6&2&0&0&0&0&0&0
\end{bmatrix}$$

Matrix HH describes the degree of host node dependence among host nodes.

$$HH=\begin{bmatrix}
0&0.1728&0.7272&0.1\\
0.2051&0&0.17&0.6249\\
0.5530&0.2020&0&0.2450\\
0.0507&0.1442&0.8051&0
\end{bmatrix}$$

4.2 Results

The importance degrees of these factors are assumed to be real numbers in [0, 0.5].The factors influencing system deployment are supposed to satisfy linear function relationship. By LinearMethod, construct the degree matrix CC of component dependence among software components.

$$CC=\begin{bmatrix}
0 & 0.2168 & 0.3097 & 0.4439 & 0.8133 & 0.1363 & 0.1351 & 0.9776 & 0.0550 & 0.3580 & 0.4256 & 0.3167 & 0.1630 & 0.4796 \\
0.2319 & 0 & 0.2916 & 0.1443 & 0.6271 & 0.7129 & 0.2625 & 0.0272 & 0.1824 & 0.3738 & 0.4762 & 0.3499 & 0.8021 & 0.2113 \\
0.4856 & 0.0528 & 0 & 0.6376 & 0.9275 & 0.2690 & 0.2875 & 0.0294 & 0.3931 & 0.4877 & 0.6562 & 0.0084 & 0.1990 & 0.5459 \\
0.6522 & 0.8481 & 0.3743 & 0 & 0.1121 & 0.2207 & 0.0609 & 0.3576 & 0.2154 & 0.0910 & 0.1962 & 0.3969 & 0.2293 & 0.3911 \\
0.5568 & 0.3518 & 0.8068 & 0.2999 & 0 & 0.6170 & 0.4960 & 0.3134 & 0.2998 & 0.3382 & 0.1628 & 0.2334 & 0.3371 & 0.2411 \\
0.4787 & 0.7937 & 0.0974 & 0.3516 & 0.3438 & 0 & 0.1863 & 0.3109 & 0.0768 & 0.5149 & 0.6133 & 0.0850 & 0.3683 & 0.0929 \\
0.6222 & 0.0683 & 0.2584 & 0.2688 & 0.8995 & 0.2816 & 0 & 0.3100 & 0.3525 & 0.2077 & 0.6505 & 0.2376 & 0.5175 & 0.1870 \\
0.9442 & 0.6544 & 0.1930 & 0.6302 & 0.0034 & 0.0178 & 0.0134 & 0 & 0.7045 & 0.0682 & 0.2886 & 0.4148 & 0.6705 & 0.8445 \\
0.4927 & 0.1102 & 0.3333 & 0.5159 & 0.3155 & 0.3299 & 0.1233 & 0.7207 & 0 & 0.4250 & 0.0311 & 0.9697 & 0.5206 & 0.1595 \\
0.6437 & 0.6114 & 0.2744 & 0.2110 & 0.4010 & 0.3796 & 0.1567 & 0.3965 & 0.1557 & 0 & 0.6799 & 0.4138 & 0.6609 & 0.4714 \\
0.2265 & 0.3066 & 0.3974 & 0.6461 & 0.2945 & 0.9413 & 0.6171 & 0.9992 & 0.0074 & 0.2216 & 0 & 0.4688 & 0.3710 & 0.2017 \\
0.0133 & 0.1252 & 0.0429 & 0.5365 & 0.3928 & 0.5802 & 0.5893 & 0.1348 & 0.7221 & 0.1611 & 0.2280 & 0 & 0.6416 & 0.6913 \\
0.4675 & 0.1662 & 0.4393 & 0.1633 & 0.0059 & 0.4496 & 0.2938 & 0.2233 & 0.6685 & 0.6382 & 0.6585 & 0.2632 & 0 & 0.4120 \\
0.9096 & 0.1311 & 0.8386 & 0.3099 & 0.6820 & 0.6434 & 0.3697 & 0.1326 & 0.5799 & 0.6641 & 0.6711 & 0.4820 & 0.4869 & 0
\end{bmatrix}$$

Because the system contains 4 host nodes, 14 components should be divided into 4 component subsets. By Div method, component subset 1 is $\{C_2,C_6,C_8,C_{10},C_{11},C_{13}\}$, component subset 2 is $\{C_9,C_{12}\}$, component subset 3 is $\{C_1,C_{14}\}$ and component subset 4 is $\{C_3,C_4,C_5,C_7\}$. By Dccs method, degree of component dependence among the four component subsets is calculated, as shown in Figure 7. By Deploy method, 14 components are deployed on 4 host nodes, as shown in Figure 8.

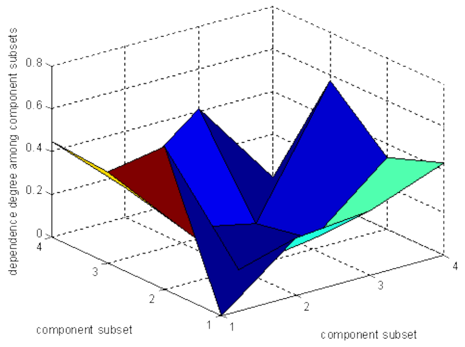

Figure 7. Degree of component dependence among component subsets

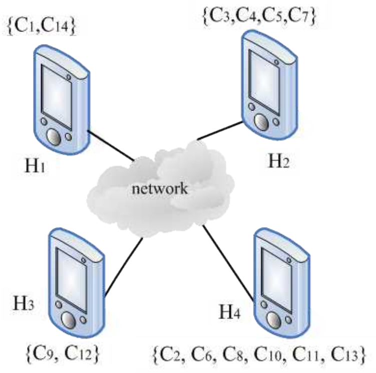

Figure 8. System deployment architecture by Greedy_Deploy algorithm

The adaptive greedy algorithm [18] trys to select the “best” host node, which has the highest sum of network reliabilities with other host nodes and the maximum memory capacity in the system. This algorithm selects the “best” component with the highest frequency of interaction with other components in the system. In general, the complexity of the adaptive greedy algorithm is $O(n^3)$. The Greedy_Deploy algorithm does not only incorporates with the influence of network reliability, memory capacity, frequency of interaction among software components, but also incorporates with the influence of component size, software architectural styles and so on. The Greedy_Deploy algorithm translates the multi-dimension factors into degree matrices of component dependence and host node dependence, and deploys components on host nodes. In general, the complexity of the algorithm is $O(mn^2)$ (m is the number of host nodes, and n is the number of components in the system. In general, m << n). Deployment and reliability analysis of the two algorithms are shown in Table III. As presented in Table III, Greedy_Deploy algorithm gets higher reliability of the deployment architecture than adaptive greedy algorithm.

Table III. Deployment and reliability analysis of two algorithms

Algorithm	H_1	H_2	H_3	H_4	Reliability
Greedy_Deploy	C_1, C_{14}	C_3, C_4, C_5, C_7	C_9, C_{12}	$C_2, C_6, C_8, C_{10}, C_{11}, C_{13}$	0.9732
Adaptive Greedy	C_3, C_5	C_6, C_9, C_{12}	C_4, C_8, C_{10}	$C_1, C_2, C_7, C_{11}, C_{13}, C_{14}$	0.9158

5. CONCLUSIONS

Software deployment is separated from developing process and has become an independent stage. The research of software deployment began in recent years and the SA based software deployment has an even shorter history. In this paper, the new reliability estimation model at architectural level is proposed and the Greedy_Deploy algorithm is presented. In contrast with existing architecture-based models, the model contains more influencing factors, such as system deployment. The Greedy_Deploy algorithm can reduce the complexity of finding good deployment architecture from $O(n^3)$ to $O(mn^2)$, and save much processing time. Experimental results show that Greedy_Deploy algorithm can obtain a higher deployment reliability of about 6.27% than adaptive greedy algorithm.

ACKNOWLEDGMENT

The work described in this paper is supported by High Technology Research and Development Plan of China (Project No. 2008AA01A201), High Technology Research and Development Plan of China (Project No. 2006AA01A103), High Technology Research and Development Plan of China (Project No. 2009AA01A404), High Technology Research and Development Plan of China (Project No.2008AA01A204).

REFERENCES

1. C. Seo, S. Malek, G. Edwards, D. Popescu, N. Medvidovic, B. Petrus and S. Ravula, "Exploring the role of software architecture in dynamic and fault tolerant pervasive systems", *SEPCASE*, IEEE Computer Society Washington, DC,USA, 2007.
2. S. Malek, H. R. Krishnan and J. Srinivasan, "Enhancing middleware support for architecture-based development through compositional weaving of styles", *Journal of Systems and Software*, Elsevier B. V., Amsterdam, Netherlands, 2010.
3. N. Medvidovic and S. Malek, "Software deployment architecture and quality-of-service in pervasive environments", *ESSPE*, ACM New York, NY, USA, 2007.
4. K. G. Popstojanova and K. S. Trivedi, "Architecture-based approaches to software reliability prediction", *Journal of Computer & Mathematics with Applications*, Elsevier Science Ltd, 2003.
5. R. H. Reusser, H. W. Schmidt and I. H. Poernomo, "Reliability prediction for component-based software architectures", *Journal of Systems and Software*, Elsevier B.V., Amsterdam, Netherlands, 2002.

6. W. Wang, Y. Wu and M. Chen, "An architecture-based software reliability model", *PRDC*, IST, Pennsylvania, 1999.
7. K. Seigrist, "Reliability of systems with markov transfer of control", *IEEE Transaction on Software Engineering*, IEEE Press Piscataway, NJ, USA, 1988.
8. S. S. Gokhale and S. Yacoub, "Performance analysis of a pipeline software architecture", *COMPSAC*, IEEE Computer Society Washington, DC, USA, 2005.
9. S. Gokhale, M. R. Lyu and K. S. Trivedi, "Reliability simulation of component-based software systems", *ISSRE*, IEEE Computer Society, Washington, DC, USA, 1998.
10. S. Gokhale,W. E. Wong, K. S. Trivedi and J. R. Horgan, "An analytic approach to architecture-based software performance and reliability prediction", *Performance Evaluation*, Elsevier Science Publishers B.V., Amsterdam, Netherlands, 2004.
11. S. Yacoub, B. Cukic and H. Ammar, "A scenario-based analysis for component-based software", *IEEE Transaction on Reliability*, IST, Pennsylvania State, 2004.
12. V. Cortellesa and V. Grassi, "A Modeling approach to analyze the impact of error propagation on reliability of component-based systems", *Lecture Notes in Computer Science*, Springer Berlin, Heidelberg, 2007.
13. A. Hassan, A. Guedem, W. Abdelmoez, D. Eldin and M. Nassar, "Architectural level risk analysis using UML", *IEEE Transactions on Software Engineering*, IST, Pennsylvania, 2003.
14. C. Vo and T. Torabi, "A framework of over the air provider-initiated software deployment on mobile devices", *ASWEC*, IEEE Computer Society, Washington, DC, USA, 2008.
15. M. Hong and S. J. Rong, "Progress of research on software architecture", *Journal of Software*, Science Press, Beijing China, 2006.
16. T. Jin, "Hierarchical variance decomposition of system reliability estimates with duplicated components", *IEEE Transaction on Reliability*, IEEE Press, Washington, DC, USA, 2008.
17. C. Thomas, L. Charles, R. Ronald and S. Clifford, *"Introduction to algorithms"*, MIT Press, Cambridge, 2001.
18. M. Mikic-Rakic, N. Medvidovic, "Support for disconnected operation via architectural self-reconfiguration", *ICAC*, IEEE Computer Society Washington, DC, USA, 2004.

ABOUT THE AUTHORS

X. Su 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.
Corresponding author, Email:suxihong07@gmail.com

Z. Wu 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.

H. Liu received the Ph.D. degree in school of computer science and technology from Harbin Institute of Technology, Harbin, China, in 2004. He is a professor in school of computer science and technology at Harbin Institute of Technology. His research interests include fault-tolerant computing, software reliability evaluation, software testing. He is a senior member of the CCF.

X. Yang received the B.S. degree in school of computer science and technology from Harbin Institute of Technology, Harbin, China, in 1964. He is a professor and Ph.D. supervisor in school of computer science and technology at Harbin Institute of Technology. His research interests include mobile computing, software architecture, fault-tolerant computing and system reliability estimation. He is a senior member of the CCF.

D. Zuo received the Ph.D. degree in school of computer science and technology from Harbin Institute of Technology, Harbin, China, in 2001. 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.