doi:10.3772/j.issn.1006-6748.2012.03.010

Software architecture based software deployment reliability estimation considering architectural style①

Su Xihong (苏喜红)②, Wu Zhibo, Liu Hongwei, Yang Xiaozong, Zuo Decheng
(School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, P. R. China)

Abstract

Software today often consists of a large number of components offering and requiring services. Such components should be deployed into embedded, pervasive environments, and several deployment architectures are typically possible. These deployment architectures can have significant impacts on system reliability. However, existing reliability estimation approaches are typically limited to certain classes or exclusively concentrate on software reliability, neglecting the influence of hardware resources, software deployment and architectural styles. The selection of an appropriate architectural style has a significant impact on system reliability of the target system. Therefore, we propose a novel software architecture (SA) based reliability estimation model incorporating software deployment and architectural style. On the basis of two architectural styles, we design influence factors and present a new approach to calculate system reliability. Experimental results show that influence factors provide an accurate and simple method of reflecting architectural styles and software deployment on system reliability. It is important for considering the influence of other architectural styles on system reliability in large scale deployment environment.

Key words: software architecture (SA), software deployment, reliability, architectural style, component

0 Introduction

The past few decades 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. Previous studies have shown that a promising approach resolving the challenges of developing large-scale software systems is to employ the principles of software architectures[1]. Software architecture (SA) provides abstractions for representing the structure, behavior, and key properties of a software system[2]. They are described in terms of software components (computational elements), connectors (interaction elements), and their configurations.

In the domain of pervasive systems, system deployment architecture is a specific facet of software system. System deployment architecture is the allocation of the system software components (and connectors) on its hardware host nodes. Deployment architecture is particularly important in pervasive environments, because a system is typically comprised of many different, heterogeneous, mobile, and possibly mutable execution platforms during its lifetime[3].

Several recent approaches have begun to quantify software reliability at the level of architectural models, or at least in terms of high-level system structure[4-8]. All of these efforts focus on the system-level reliability prediction. However, these reliability estimation approaches are typically limited to certain classes or exclusively concentrate on software reliability, neglecting the influence of hardware resources, system deployment architecture and architectural styles. Software architectural styles further codify structural, behavioral, interaction, and composition guidelines that are likely to result in software systems with desired properties[9]. Therefore, in this paper, we predict system reliability at architecture-level incorporating software deployment and architectural styles.

The rest of this paper is organized as follows. Section 1 provides a brief description of the basic concepts and related work. Section 2 describes system deploy-

① Supported by the National High Technology Research and Development Programme of China (No. 2008AA01A201; 2006AA01A103; 2009AA01A404).

② To whom correspondence should be addressed. E-mail: suxihong07@126.com

Received on Aug. 18, 2010

ment architecture, and proposes the approaches of calculating conditional failure probabilities of components and system reliability with two different architectural styles. The two architecture styles are CS and LCS. The experiments are given in Section 3. Finally, the Section 4 concludes the paper.

1 Terminology and related work

In this section, we present the basic concepts: software deployment and architectural style. We investigate the approaches of software deployment and different architectural styles.

1.1 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[10].

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 offering and requiring services. Such components are often deployed into embedded, pervasive environments adding to the complexity of software deployment[11].

1.2 Architectural style

Since an architecture embodies both functional and non-functional properties, it would be difficult to compare architectures directly for different types of systems, or for even the same type of system set in different environments. Styles are a mechanism for categorizing architectures and for defining their common characteristics.

An architectural style is a coordinated set of architectural constraints that restricts the roles/features of architectural elements and the allowed relationships among those elements within any architecture that conforms to that style[12].

1.3 Approaches of software deployment

(1) Just-in-time software deployment

Just-in-time software deployment is one in which installation and activation are performed at the possibly latest time when users access the software.

(2) Component-based software deployment

A component-based architecture of software deployment on mobile devices was developed by Fjellheim, Taconet and Kwan. This architecture uses a functionally adaptation technique based on current context. The technique involves changing the way that software carries out its functionality. This approach supports adaptation to user's context and device capability[13].

(3) Middleware-based software deployment

Almost all approaches of software deployment have employed middleware architecture. The architecture often consists of two modules. A module at the server is to handle users' requests, find correct software, and deliver software to devices. The other module at the client, the middleware is to download, install, and update software.

(4) Context-aware software deployment

Context-aware software deployment employs context to determine functionalities or components being deployed to mobile devices.

(5) Pull model of software deployment

Pull model of software deployment allows users to discover software by themselves. The pull model is suitable for software deployment on user demand.

(6) Push model of software deployment

Push model or provider-initiation of software deployment allows providers to initiate a deployment process if they want. The model has been popularly applied for software deployment on desktop computers[13].

1.4 Different architectural styles

(1) Pipe and filter (PF)

In a pipe and filter style, each component (filter) has a set of inputs and a set of outputs. A component reads streams of data on its inputs and produces streams of data on its outputs, delivering a complete instance of the result in a standard order[14].

(2) Replicated repository (RR)

Systems based on the replicated repository style improve the accessibility of data and scalability of services by having more than one process and provide the same service[12].

(3) Client-server (CS)

The client-server style is used most frequently in the architectural styles for network-based applications. A server component, offering a set of services, listens to the requests upon those services. A client component, desiring that a service be performed, sends a request to the server via a connector. The server either rejects or performs the request and sends a response back to the client[14].

(4) Layered-client-server (LCS)

A layered system is organized hierarchically, and each layer provides services to the layer above it and uses services of the layer below it. Although a layered system is considered as a "pure" style, its use within network-based systems is limited to its combination with the client-server style to provide layered-client-server. Layered-client-server adds proxy and gateway components to the client-server style. Architectures based on the layered-client-server are referred to as two-tiered, three-tiered, or multi-tiered architectures in the information systems.

(5) Event-based integration (EBI)

The event-based integration style, also known as the implicit invocation or event system style, reduces coupling between components by removing the need for identification on the connector interface.

(6) C2

The C2 architectural style is directed at supporting large grain reuse and flexible composition of system components by enforcing substrate independence. It does so by combining event-based integration with layered-client-server.

(7) Hybrid style

Since architectural styles may address aspects of software architecture, a given architecture may be composed of multiple styles. Likewise, a hybrid style can be formed by combining multiple basic styles into a single coordinated style.

2 System reliability

In order to estimate the system reliability, we may meaningfully combine different approaches of software deployment with different architectural styles. In this section, a component-based approach for software deployment is the basis of SA based reliability estimation model. This model includes two architectural styles: CS and LCS.

2.1 System deployment architecture

The basic entities of SA based software deployment reliability estimation model include host nodes, components and services. These entities seem appropriate to model component deployment in embedded and pervasive systems. In details, a model consists of

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

(2) a set of components, $C = \{C_1, C_2, \cdots, C_N\}$, which represents the components of a system.

(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.

The system deployment architecture is important to system reliability in the face of component failure and host node failure. We use matrix $\boldsymbol{HC}$ to describe the deployment relationship between components and host nodes.

$$\boldsymbol{HC} = \begin{array}{c} \\ H_1 \\ H_2 \\ \vdots \\ H_M \end{array} \begin{array}{c} \begin{array}{cccc} C_1 & C_2 & \cdots & C_N \end{array} \\ \begin{bmatrix} HC_{1,1} & HC_{1,2} & \cdots & HC_{1,N} \\ HC_{2,1} & HC_{2,2} & \cdots & HC_{2,N} \\ \vdots & \vdots & \vdots & \vdots \\ HC_{M,1} & HC_{M,2} & \cdots & HC_{M,N} \end{bmatrix} \end{array}$$

The value of $HC_{i,j}$ in matrix HC may be 1 or 0, as shown in Eq. (1).

$$HC_{i,j} = \begin{cases} 1, & \text{if component } C_j \text{ is deployed on } H_i \\ 0, & \text{if component } C_j \text{ is not deployed on } H_i \end{cases} \quad (1)$$

2.2 Architectural style

To illustrate the impact of architectural styles on system reliability, we adopt two architectural styles (CS and LCS) in our reliability estimation model.

2.2.1 CS

In this subsection, we describe three research hypotheses:

(1) there are two types of components: client component and server component;

(2) a client component sends requests to and receives responses from a server component;

(3) a server component receives requests from and sends responses to a client component.

We consider a system consisting of a set of C components, $C = \{C_1, \cdots, C_N\}$ and $|C| = N$. In a deployment architecture, if a set of SC server components include $C_l, \cdots, C_k$, $SC = \{C_l, \cdots, C_k\}$, the other components constitute the set of SC_2 client components, $SC_2 = C - SC$. The failure impact of different types of components on system failure is different, as shown in Eqs(2) and (3). In Eq. (2), fc_i is the original conditional failure probability of components and fc'_i is the adjusted conditional failure probability of components according to component types. a, a_1 and a_3 are called influence factor. These influence factors are all real numbers $[0,1]$.

$$fc'_i = a \times fc_i \quad (2)$$

$$a = \begin{cases} a_1, & \text{if } C_i \in SC \\ a_3, & \text{if } C_i \in SC_2 \end{cases} \quad (3)$$

2.2.2 LCS

In this subsection, we describe four research hypotheses:

(1) there are three types of components: client component, middle component and server component;

(2) a client component sends requests to and receives responses from a middle component;

(3) a middle component sends requests to and receives responses from a server component;

(4) a server component receives requests from and sends responses to a middle component.

We consider a system consisting of a set C of components, $C = \{C_1, \cdots, C_N\}$ and $|C| = N$. Components of set C should be divided into three subsets: SC, SC_1, SC_2. SC is the set of server components. SC_1 is the set of middle components. SC_2 is the set of client components. That is, $C = SC \cup SC_1 \cup SC_2$, $SC \cap SC_1 = \phi$, $SC \cap SC_2 = \phi$, $SC_1 \cap SC_2 = \phi$. The failure impact of different types of components on system failure is different, as shown in Eqs(4) and (5). In Eq. (4), fc_i is the original conditional failure probability of components and fc'_i is the adjusted conditional failure probability of components according to component types. a, a_1, a_2 and a_3 are also called influence factor. These influence factors are all real numbers $[0,1]$.

$$fc'_i = a \times fc_i \tag{4}$$

$$a = \begin{cases} a_1, \text{ if } C_i \in SC \\ a_2, \text{ if } C_i \in SC_1 \\ a_3, \text{ if } C_i \in SC_2 \end{cases} \tag{5}$$

2.3 Calculating conditional failure probability of components

2.3.1 CS

(1) Conditional failure probability of server components

If server component C_s is deployed on host node H_j, the original conditional failure probability fc_s of C_s is calculated in Eq. (6). fc'_s is calculated in Eq. (7). ph_j is the failure probability of host node H_j. pc_s is failure probability of component C_s.

$$fc_s = 1 - (1 - ph_j) \times (1 - pc_s) \tag{6}$$

$$fc'_s = a_1 \times fc_s \tag{7}$$

(2) Conditional failure probability of client components

Client component C_i needs to send requests to many server components, for example $C_l, \cdots C_h$. These server components have been deployed on many host nodes, for example $H_p, \cdots, H_q$. $CSCH$ represents the set of these host nodes, $CSCH = \{H_p, \cdots, H_q\}$, $|CSCH| = k$. That is, these server components $C_l, \cdots, C_h$ should be divided into k component subsets, $CSCR_1, \cdots, CSCR_k$. Components of each subset should be deployed on one host node. We suppose that components of set $CSCR_1$ are deployed on host node H_p and components of set $CSCR_k$ are deployed on host node H_q and so on.

If client component C_i has been deployed on host node H_l, $H_l \in CSCH$. The original conditional failure probability fc_i of C_i is expressed in Eq. (8). S_j describes the probability of normal function of all server components of one component subset.

$$fc_i = 1 - (1 - pc_i) \times \prod_{j=1}^{k} S_j \tag{8}$$

$$\begin{aligned} S_1 &= (1 - ph_p) \times \prod_{\forall C_i \in CSCR_1} (1 - pc_i) \\ &\vdots \\ S_k &= (1 - ph_q) \times \prod_{\forall C_j \in CSCR_k} (1 - pc_j) \end{aligned} \tag{9}$$

If client component C_i has been deployed on host node H_l, $H_l \notin CSCH$. The original conditional failure probability fc_i of C_i is expressed in Eq. (10). S_j describes the probability of normal function of all server components of one component subset.

$$fc_i = 1 - (1 - ph_l) \times (1 - pc_i) \times \prod_{j=1}^{k} S_j \tag{10}$$

Because C_i is a client component, fc'_i can be calculated in Eq. (11).

$$fc'_i = a_3 \times fc_i \tag{11}$$

2.3.2 LCS

(1) Conditional failure probability of server components

If server component C_s is deployed on host node H_j, the original conditional failure probability fc_s of C_s can be calculated in Eq. (6). fc'_s can be calculated in Eq. (7).

(2) Conditional failure probability of middle components

Middle component C_m needs to send requests to many server components, for example $C_x, \cdots, C_y$. These server components have been deployed on many host nodes, for example $H_a, \cdots, H_b$. $LCSCH$ represents the set of these host nodes, $LCSCH = \{H_a, \cdots, H_b\}$, $|LCSCH| = k_S$. That is, these server components $C_x, \cdots, C_y$ need to be divided into k_S component subsets, $LCSCR_1, \cdots, LCSCR_{k_S}$. Similarly, we suppose that components of set $LCSCR_1$ are deployed on host node H_a and components of set $LCSCR_{k_S}$ are deployed on host node H_b and so on. Therefore, we can obtain fc_m and fc'_m as follows.

If middle component C_m has been deployed on host node H_d, $H_d \in LCSCH$. We calculate fc_m in Eq. (12). S_j is the probability of normal function of all server components of one component subset.

$$fc_m = 1 - (1 - pc_m) \times \prod_{j=1}^{k_S} S_j \tag{12}$$

$$\begin{aligned} S_1 &= (1 - ph_a) \times \prod_{\forall C_i \in LCSCR_1} (1 - pc_i) \\ &\vdots \\ S_{k_S} &= (1 - ph_b) \times \prod_{\forall C_j \in LCSCR_{k_S}} (1 - pc_j) \end{aligned} \tag{13}$$

If middle component C_m has been deployed on host node H_d, $H_d \notin LCSCH$. We calculate fc_m in Eq. (14). S_j is the probability of normal function of all server components of one component subset.

$$fc_m = 1 - (1 - ph_d) \times (1 - pc_m) \times \prod_{j=1}^{k_S} S_j \tag{14}$$

C_m is a middle component, fc'_m can be calculated in Eq. (15).

$$fc'_m = a_2 \times fc_m \tag{15}$$

(3) Conditional failure probability of client component

Client component C_c needs to send requests to many middle components, for example $C_e, \cdots, C_f$. These middle components have been deployed on many host nodes, for example $H_g, \cdots, H_l$. $LCSCHM$ represents the set of these host nodes, $LCSCHM = \{H_g, \cdots, H_l\}$, $|LCSCHM| = k_M$. That is, these middle components $C_e, \cdots, C_f$ should be divided into k_M component subsets, $LCSCRM_1, \cdots, LCSCRM_{k_M}$. We suppose that components of set $LCSCRM_1$ are deployed on host node H_g and components of $LCSCRM_{k_M}$ are deployed on host node H_l and so on.

$\exists LCSCRM_i (i = 1, 2, \cdots, k_M)$, $\exists C_m \in LCSCRM_i$, middle component C_m needs to send requests to many server components, for example $C_g, \cdots, C_r$. These server components should be deployed on many host nodes, for example $H_m, \cdots, H_n$. $LCSCHS$ represents the set of these host nodes. $LCSCHS = \{H_m, \cdots, H_n\}$, $|LCSCHS| = k_S$. That is, these server components $C_g, \cdots, C_r$ should be divided into k_S component subsets: $LCSCRS_1, \cdots, LCSCRS_{k_S}$. We suppose that components of set $LCSCRS_1$ are deployed on host node H_m and the components of $LCSCRS_{k_S}$ are deployed on host node H_n and so on.

We assume that component C_m is deployed on host node H_{r2} and component C_c is deployed on host node H_{r1}. $HH = LCSCHM \cup LCSCHS$. Then fc_c is generated in the following four cases.

(1) if $H_{r1} \in HH, H_{r2} \in HH$

$$fc_c = 1 - (1 - pc_c) \times (1 - pc_m) \times temp \tag{16}$$

(2) if $H_{r1} \notin HH$, $H_{r2} \in HH$

$$fc_c = 1 - (1 - pc_c) \times (1 - pc_m) \times (1 - ph_{r1}) \times temp \tag{17}$$

(3) if $H_{r1} \in HH$, $H_{r2} \notin HH$

$$fc_c = 1 - (1 - pc_c) \times (1 - pc_m) \times (1 - ph_{r2}) \times temp \tag{18}$$

(4) if $H_{r1} \notin HH$, $H_{r2} \notin HH$

$$fc_c = 1 - (1 - pc_c) \times (1 - pc_m) \times (1 - ph_{r1}) \times (1 - ph_{r2}) \times temp \tag{19}$$

where $temp$ is a temporary variable, as shown in Eq. (20). $S_{m,1}, \cdots, S_{m,k_S}$ describes the probability of normal function of server component sets for component C_m. They can be calculated in Eq. (21) for middle component C_m. If components of subset $LCSCRM_m$ are deployed on the host node H_l, ph'_m is the failure probability of host node H_l.

$$temp = \prod_{m=1}^{k_M} (((1 - ph'_m) \times \prod_{\forall C_i \in LCSCRM_m} (1 - pc_i)) \times \prod_{j=1}^{k_S} S_{m,j}) \tag{20}$$

$$\begin{aligned} S_{m,1} &= (1 - ph_m) \times (\prod_{\forall C_i \in LCSCRS_1} (1 - pc_i)) \\ &\vdots \\ S_{m,k_S} &= (1 - ph_n) \times (\prod_{\forall C_i \in LCSCRS_{k_S}} (1 - pc_i)) \end{aligned} \tag{21}$$

Since C_c is a client component, we obtain fc'_c in Eq. (22).

$$fc'_c = a_3 \times fc_c \tag{22}$$

2.4 Reliability of calculation system

System reliability estimations are often obtained based on the reliability information of subsystems, or components[15]. We suppose that a system consists of N components and M host nodes. Component failure and host node failure are the main source of system reliability decreasing. System reliability is calculated in Eq. (23). R_{system} is the system reliability. P_{system} is the failure probability of system. $ph_{1,M}$ is failure probability of host nodes $H_1, \cdots, H_M$ simultaneously. fc'_i is the adjusted conditional failure probability of C_i.

$$\begin{aligned} R_{system} &= 1 - p_{system} \\ &= 1 - \bigcup_{j=1}^{M} ph_j - (\prod_{j=1}^{M} (1 - ph_j)) \times (\bigcup_{i=1}^{N} fc'_i) \end{aligned} \tag{23}$$

$$\bigcup_{j=1}^{M} ph_j = \sum_{j=1}^{M} ph_j - \sum_{1 \le i < j \le M} ph_{i,j} + \cdots + (-1)^{M-1} ph_{1,M} \tag{24}$$

3 Experiment

In this section, the experiments are based on randomly generated inputs: failure probabilities of four host nodes, failure probabilities of fourteen components and graphs of component interaction of two architectural styles. The two architectural styles are CS and LCS. In CS style, components should be divided into two subsets: set SC of server components and set SC_2 of client components. In LCS style, components should be divided into three subsets: set SC of server components, set SC_1 of middle components and set SC_2 of client components. We deploy fourteen components on four host nodes. Then, we calculate system reliability with different architectural styles. We investigate the impact of influence factors (e. g. a_3) on system reliability.

3.1 Experiement one

In this experiment, the set C of components is $C = \{C_1, C_2, \cdots, C_{14}\}$. Components of set C can be divided into two subsets: SC and SC_2. We calculate system reliability with architectural style CS.

(1) Failure probabilities of four host nodes

Failure probabilities of four host nodes are real numbers $[0, 0.01]$, $ph_1 = 0.0013$, $ph_2 = 0.0005$, $ph_3 = 0.0009$, $ph_4 = 0.0002$.

(2) Failure probabilities of fourteen components

Failure probabilities of fourteen components are real numbers $[0, 0.01]$. $pc_1 = 0.0004$, $pc_2 = 0.0013$, $pc_3 = 0.0007$, $pc_4 = 0.0003$, $pc_5 = 0.0011$, $pc_6 = 0.0002$, $pc_7 = 0.0006$, $pc_8 = 0.0012$, $pc_9 = 0.0008$, $pc_{10} = 0.0013$, $pc_{11} = 0.0021$, $pc_{12} = 0.0012$, $pc_{13} = 0.0001$, $pc_{14} = 0.0002$.

(3) Graph of component interaction of CS style

As seen in Fig. 1, $SC = \{C_2, C_4, C_8, C_{10}, C_{14}\}$ and $SC_2 = C - SC$. Component C_{12} sends requests to C_2 in real line and component C_{12} receives responses from C_2 in dotted line.

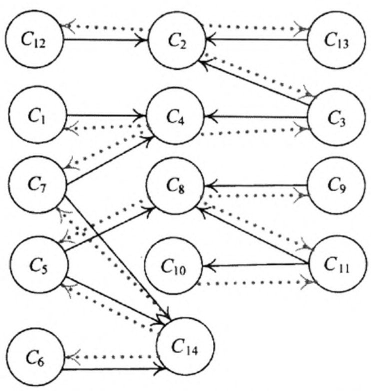

Fig. 1 Graph of component interaction with CS style

(4) Deployment architecture matrix

Matrix $\boldsymbol{HC}_1$ describes how to deploy fourteen components on four host nodes. That is, components C_1, C_2, C_3, C_4, C_{12}, C_{13} are deployed on H_1. Components C_5, C_6, C_{14} are deployed on H_2. Components C_7, C_8, C_9 are deployed on H_3. Components C_{10}, C_{11} are deployed on H_4.

$$\boldsymbol{HC}_1 = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}$$

(5) System reliability

The value of a_3 reflects the importance of client components in system reliability estimation. The function relationship between a_1 and a_3 depends on concrete system. It may be linear function, logarithmic function, exponential function and so on. We use linear function as the basis of the experiment and investigates seven cases of the relationship between a_1 and a_3. The first case is $a_1 + a_3 = 1$. The second case is $1.3a_1 + a_3 = 1$. The third case is $1.2a_1 + a_3 = 1$. The fourth case is $1.1a_1 + a_3 = 1$. The fifth case is $0.9a_1 + a_3 = 1$. The sixth case is $0.8a_1 + a_3 = 1$. The seventh case is $0.7a_1 + a_3 = 1$. The initial value of a_3 is 0.003.

Fig. 2 shows system reliability varying with the value of a_3. *FirC* is system reliability of the first case. *SecC* is system reliability of the second case. *ThiC* is the system reliability of third case. *FouC* is system reliability of the fourth case. *FifC* is system reliability of the fifth case. *SixC* is system reliability of the sixth case. *SevC* is system reliability of the seventh case. With increasing of the value of a_3, system reliability decreases. Various linear combinations of a_1 and a_3 have an important effect on system reliability.

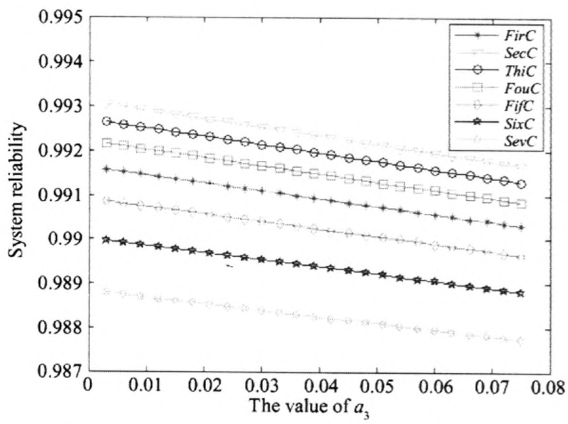

Fig. 2 System reliability with different values of a_3

3.2 Experiement two

In this experiment, the set C of components is $C = \{C_1, C_2, \cdots, C_{14}\}$. Components of set C should be divided into three subsets: SC, SC_1 an SC_2. We investigate system reliability with architectural style LCS.

(1) Failure probabilities of four host nodes

Failure probabilities of four host nodes are real numbers $[0, 0.01]$, $ph_1 = 0.0013$, $ph_2 =$

0.0005, $ph_3 = 0.0009$, $ph_4 = 0.0002$.

(2) Failure probabilities of fourteen components

Failure probabilities of fourteen components are real numbers [0, 0.01]. $pc_1 = 0.0004$, $pc_2 = 0.0013$, $pc_3 = 0.0007$, $pc_4 = 0.0003$, $pc_5 = 0.0011$, $pc_6 = 0.0002$, $pc_7 = 0.0006$, $pc_8 = 0.0012$, $pc_9 = 0.0008$, $pc_{10} = 0.0013$, $pc_{11} = 0.0021$, $pc_{12} = 0.0012$, $pc_{13} = 0.0001$, $pc_{14} = 0.0002$.

(3) Graph of component interaction with LCS style

As seen in Fig. 3, $SC = \{C_2, C_4, C_8, C_{10}\}$, $SC_1 = \{C_7, C_9, C_{14}\}$ and $SC_2 = \{C_1, C_3, C_5, C_6, C_{11}, C_{12}, C_{13}\}$. Client component C_1 sends requests to middle component C_9 in real line and component C_1 receives responses from component C_9 in dotted line. Middle component C_9 sends requests to server component C_2 in real line and component C_9 receives responses from component C_2 in dotted line.

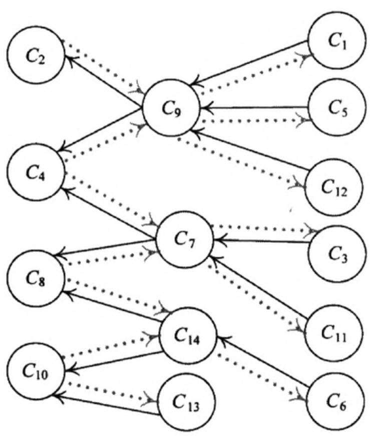

Fig. 3 Graph of component interaction with LCS style

(4) Deployment architecture matrix

Matrix $\boldsymbol{HC}_2$ describes how to deploy fourteen components on four host nodes.

$$\boldsymbol{HC}_2 = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

(5) System reliability

The value of a_3 reflects the importance of client components and the value of a_2 reflects the importance of middle components in system reliability estimation. The function relationship among a_1, a_2 and a_3 depends on concrete system. It may be linear function, logarithmic function, exponential function and so on. We also use linear function as the basis of the experiment. That is, $a_1 + a_2 + a_3 = 1$. The initial value of a_3 is 0.003. We discuss two different relationships among a_1, a_2 and a_3.

(1) First case

In the CS style, the initial value of a_3 is 0.003. $a_1 + a_3 = 1$. FCSR represents system reliability with CS style varying with the value of a_3. In LCS style, the initial value of a_3 is also 0.003. $a_1 + a_2 + a_3 = 1$, $a_1 = (1 - a_3) \times 0.9$ and $a_2 = (1 - a_3) \times 0.1$. SCSR represents system reliability with LCS style varying with the value of a_3.

As seen in Fig. 4, there exists a change point. Before the change point, system reliability with LCS is higher than system reliability with CS style.

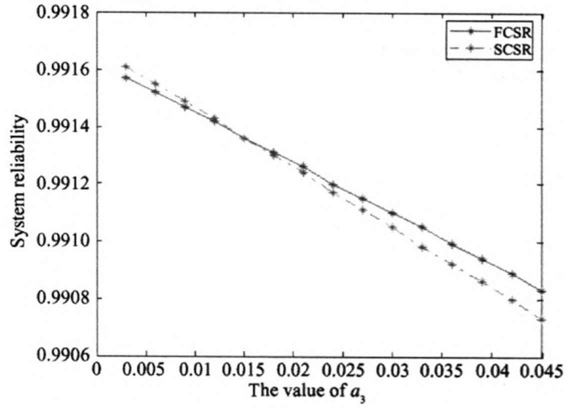

Fig. 4 Comparative graph of system reliability of different values of a_3

(2) Second case

In the CS style, the initial value of a_3 is 0.003. $a_1 + a_3 = 1$. FCSR represents system reliability with CS styles varying with the value of a_3. In the LCS style, the initial value of a_3 is 0.003. The relationship among a_1, a_2 and a_3 has changed. $a_1 = (1 - a_3) \times 0.85$, $a_2 = (1 - a_3) \times 0.15$.

As seen in Fig. 5, FCSR represents system reliability with CS style varying with the value of a_3. SCSR represents system reliability with first relationship among a_1, a_2 and a_3 in the LCS style. TCSR represents system reliability with second relationship among a_1, a_2

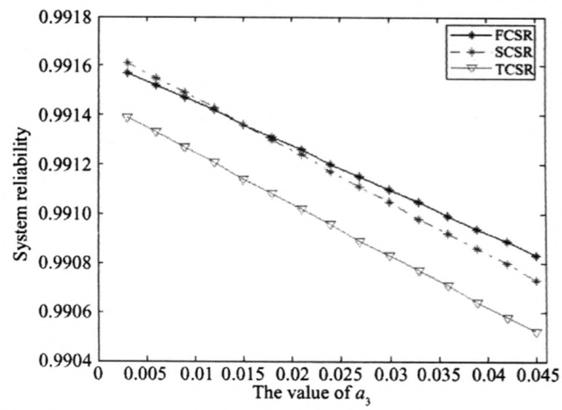

Fig. 5 Comparative graph of system reliability of different values of a_3

and a_3 in the LCS style. With the increasing value of a_3, the system reliability has decreased. Because middle components become more important in the second case, and the system reliability is lower than the one in first case.

4 Conclusions

In this paper, we present a novel system reliability estimation model at the architecture level. This model incorporates the influence of software deployment and architecture styles into system reliability estimation. There are many approaches of software deployment and different architectural styles. Our model is based on component-based software deployment and two architectural styles: CS and LCS. We also propose a new approach of calculating system reliability with different architecture styles. Simulated results show the important influence of architecture styles on system reliability. We investigate the impact of influence factors on system reliability.

References

[1] Sousa J P, Garlan D. Aura: an architectural framework for user mobility in ubiquitous computing environments. In: Proceedings of the 3rd Working IFIP/IEEE Conference on Software Architecture, Montreal, Canada, 2002. 1-18

[2] Seo C. Exploring the role of software architecture in dynamic and fault tolerant pervasive systems. In: Proceedings of 1st International Workshop on Software Engineering for Pervasive Computing Applications, Systems, and Environments, Washington, USA, 2007. 9-15

[3] Medvidovic N, Malek S. Software deployment architecture and quality-of-service in pervasive environments. In: 2007 IEEE International Workshop on the Engineering of Software Services for Pervasive Environments, Dubrovnik, Croatia, 2007. 47-51

[4] Cortellesa V, Grassi V. A modeling approach to analyze the impact of error propagation on reliability of component-based systems. In: Proceedings of the 10th International Symposium on Component-Based Software Engineering, Medford, USA, 2007. 140-156

[5] Goseva-Popstojanova K, Trivedi K S. Architecture-based approaches to software reliability prediction. *Journal of Computers & Mathematics with Applications*, 2003, 46 (7): 1023-1036

[6] Reussner R H, Schmidt H W, Poernomo I H. Reliability prediction for component-based software architectures. *Journal of Systems and Software*, 2003, 66(3): 241-252

[7] Wang W, Wu Y, Chen M. An architecture-based software reliability model. In: Proceedings of 6th Pacific Rim International Symposium on Dependable Computing, Hongkong, China, 1999. 143-150

[8] Yacoub S M, Cukic B, Ammar H H. Scenario-based reliability analysis of component-based software. In: Proceedings of 10th International Symposium on Software Reliability Engineering, Boca Raton, USA, 1999. 125-130

[9] Malek S. Effective realization of software architectural styles with aspects. In: 2008 IEEE International Conference on Software Architecture, Washington DC, USA, 2008. 313-316

[10] Dearie A. Software deployment, past, present and future. In: 2007 IEEE International Symposium on Future of Software Engineering, Washington DC, USA, 2007. 269-284

[11] Wand P, Jin T. Complex systems reliability estimation considering uncertain component lifetime distributions. In: Proceedings of the 8th International Conference on Reliability, Maintainability and Safety, Chengdu, China, 2009. 395-398

[12] Fielding R T. Architectural Styles and the Design of Network-Based Software Architectures: [Ph. D. dissertation]. Irvine, USA: University of California, 2000. 69-77

[13] Vo C C, Torabi T. A framework for over the air provider-initiated software deployment on mobile devices. In: Proceedings of 19th Australian Conference on Software Engineering, Perth, Australia, 2008. 633-638

[14] Majidi E, Alemi M, Rashidi H. Software architecture: a survey and classification. In: Proceedings of the 2nd International Conference On Communication Software and Networks, Singapore, 2010. 454-460

[15] Jin T. Hierarchical variance decomposition of system reliability estimates with duplicated components. *IEEE Transactions on Reliability*, 2008, 57(4): 564-573

Su Xihong, born in 1981. She is currently pursuing the Ph. D. degree at School of Computer Science and Technology of Harbin Institute of Technology. She received the M. S. degrees from Harbin Institute of Technology in 2007. Her research interests include software architecture analysis, architectural styles and system reliability estimation.