Reliability Analysis for Component-based Software System in Open Distributed Environments

Haiyang Hu†

† College of Computer and Information Engineering,
Zhejiang Gongshang University, Hangzhou, China, 310018

Summary
Internet provides an open, dynamic, and uncertain environment. Component-based software development in this environment faces more challenges with built upon a set of heterogeneous, autonomous software components distributed in the open network. Making analysis on the reliability of component-based software system in this environment has important meanings. However, current approaches to software reliability are not very applicable to this open environment. This paper presents a new approach to evaluate the reliability of the component-based software system in this open distributed environment by analyzing the reliabilities of the components in different application domains, the reliabilities of the connections to these components and the architecture style of their composition. Sensitivity analysis on the elements in the software system is also presented and we make experiments on an example to show the approach's characteristics.

Key words
software components, component-based software system, reliability analysis, sensitivity analysis

1. Introduction

The Internet provides a global open infrastructure for exchanging and sharing of various resources for the people all over the world. The rapid development and wide application of the Internet makes it become a new mainstream platform for software to be used, developed, deployed and executed. The Internet platform has such characteristics different from traditional platforms as: 1)Entities are heterogeneous, dynamic and unpredictable; 2)Connections of nodes are manifold: wire or wireless, fixed or mobile;3)User's requirements are more personalized and flexible [1], [2], [3]. Thus, how to analyze the reliability of component-based software system in this open distributed environment have important meanings.

At present, there are several models of reliability analysis on component-based software system, such as [6], [7], [8], [9], [12]. However, these works seldom analyze the reliability of the connections to the components apparently, and often they assume that the component is just used in single application domain so that the reliability it shows is always the same. As a result, the above models do not adapt to the analysis for component-based software system in open distributed environments quite well. This paper presents a new approach to reliability analysis on component-based software system in open distributed environments, which evaluates the different reliabilities individual component shows in its different application domains, the reliability of the connections to these components, and the architecture style of their composition, to give evidences for assessing the overall reliability of the software system.

In the following sections of this paper, related works are discussed in section 2. Reliability analysis for the component-based software system in open distributed environments is presented in section 3, and sensitivity analysis on the elements of the system is also given in this section. We make experiments on an example to show its characteristics in section 4. Finally, section 5 concludes this paper.

2. Related Works

Early approaches to reliability analysis for a component-based software system often consider the whole system as a black box, i.e., only its interactions with the outside world are modeled while without considering its internal structure. The class of these approaches [3], [4] are suited to capture the behavior of largely custom applications. With the widespread use of object-oriented technology and web-based development, component-based software development has become a hotspot in the area of software engineering. As the software component can be

Manuscript received May 5, 2007.
Manuscript revised May 20, 2007.

commercially available off the shelf or developed contractually, the whole application can be developed with different heterogeneous components. Without taking the system architecture into account, traditional approaches are not appropriate to model these systems. At present, there are several new models for reliability analysis on the component-based software system, as shown in [7], [8], [9], [12].

In [7], the authors use a component-dependency graph (CDG) to represent the interactions among components. CDG is a direct graph, which identifiers the individual component reliability, the interface reliability, the connection reliability between components, the control transition, and the transition probability. However, [7] doesn't consider the situation that several primary components can be composed into a composite component according to a certain architecture, and it just regards the connection reliability and component reliability as parameters with fixed values while without making a further analysis on them.

In [8], the authors use a path-based model to analyze the system reliability. It considers three architecture styles: single-input/single-out system, single-input/ multiple- output system, multiple-input /multiple-out system. The execution frequency of individual component is obtained by computing the transition probabilities among components. This paper also makes sensitivity analysis on the different parts in the system, based on the reliabilities of individual components and the probabilities of transitions.

In [9], the authors analyze the reliability of a component through its interface. It evaluates the component's reliability based on its parameterized contractual specifications and the state machines on the interface between the provided component and the required component.

In [12], the authors present an analytical model for estimating architecture-based software reliability, according to the reliability of each component, the operational profile and the architecture of software. The model can be utilized to estimate the reliability of a heterogeneous architecture consisting of batch-sequential/pipeline, call-and-return, parallel/pipe-filter, and fault tolerant styles.

Without taking the component's different application domains into account, in these related works, the reliability of each component is just regarded as a fixed value. And also these works seldom analyze the reliabilities of the connections to these components and their effects on the reliability of the whole architecture.

3. Reliability analysis

A component-based software system concerning reliability analysis can be described formally as follows:

Definition 1 A component-based software system can be defined as such a tuple: < SC, SL, C_{Init}, C_f, SP>:

- SC represents the set of components in the system, SC ={ $C_1, C_2, \ldots, C_n$};
- SL represents the set of connections to these components, multiple components can be composed into a system of a certain architecture style with these connections; Here, SL ={ $l_1, l_2, \ldots, l_n$}.
- C_{Init} is the component executed first by the application;
- C_f is the component executed at last by the application;
- SP represents a set of transition probabilities: SP ={ $PT_{1\to 1}$,..., $PT_{i\to j}$ ..., $PT_{n\to n}$ }, here $PT_{i\to j}$ represents the probability that the application may execute component j, after it has executed component i.

For a component i, it can show different reliabilities in its different application domains [9], [16]. Such a component can be defined as follows:

Definition 2 A component can be defined as such a tuple < F, P, D, M> :

- F is a set of functional interfaces the component provides;
- P is the component's behavioral protocol of interactions with other components;
- D is a set of application domains that the component has. D: $C\times C\to F$, which describes the situation that components interacts with each other through an interface it provides. A component can show different reliabilities in its different application domains through its interfaces.
- M: $D \to$ [0..1], which denotes the reliability that a component shows in a certain application domain.

The connection to the component is defined as follows:

Definition 3 $l_{i\to j} = < PC_{i\to j}, LT_{i\to j}, RL_{i\to j}^{\to j} >$, here:

- $PC_{i\to j} = (C_i, C_j)$ and it represents a pair of components in interaction;
- $LT_{i\to j}$ is the type of this connection. It can be a Client-Server(C-S) mode, or a mobile -agent mode or others;
- $RL_{i\to j}^{\to j} \in [0..1]$, which represents the reliability of the connection when the application begins to use it to call component *j*, after it has executed component *i*.

3.1 Reliability analysis for components

Components interact with each other through their interfaces, and also the application calls the component through the interface it provides. A component may provide several interfaces with each including a set of operations. Calling the component through one of its interfaces has formed one of its application domains [9], [16].

We give a component's behavioral transition model on its interface as follows:

Definition 4 $STM(d) = < A_d, a_I, a_F, f_{RA}, f_{PA}, f_{TA} >$ represents a behavioral transition model in the application domain d of a component, here:

- A_d represent a set of operations included in the application domain d;
- a_I is the operation executed first in the domain d;
- a_F is the operation executed finally in the domain d;
- $f_{RA}: A_d \to [0..1]$ represents the reliability of executing an operation;
- $f_{PA}: A_d \times A_d \to [0..1]$ represents the probability of a transition between operations in execution;
- $f_{TA}: A_d \to R^+$ represents the execution time of an operation.

For any two operations a_i and a_j, if there is no such behavior that application may begin to execute a_j after it has executed a_i, then $f_{PA}(a_i, a_j) = 0$; $\forall a_i \in A_k$, $\sum_{a_j \in A_k} f_{PA}(a_i, a_j) = 1$. Suppose that the application begins to execute operation a_m after it has executed a_i, then the application goes into such a state that it will begin to execute a_j next. The reliability of this execution path is $f_{RA}(a_m) f_{RA}(a_j)$, and the probability for the application running along with this execution path is $f_{PA}(a_i, a_m) \cdot f_{PA}(a_m, a_j)$. So the average reliability of transiting from the state of finishing executing a_i to the state of executing a_j is

$$\sum_m f_{RA}(a_m) \cdot f_{PA}(a_i, a_m) \cdot f_{RA}(a_j) \cdot f_{PA}(a_m, a_j).$$

Based on the above analysis, the transition matrix of a component C concerning software reliability in one of its application domains d can be given as follows:

$$M_C(d) = (f_{RA}(a_j) \cdot f_{PA}(a_i, a_j))_{ij}, \quad 1 \le i, j \le |A_d| \quad (1)$$

Let $M_C^k(d) = M_C^{k-1}(d) \cdot M_C(d)$, and assume that the operation executed finally is a_n. Then, from Cheung model [10], the average reliability of component C in one of its application domains d is:

$$r_C(d) = f_{RA}(a_1) \cdot (I_{|A_d|} - M_C(d))(1, n), \quad (2)$$

Here $I_{|A_d|}$ is an identity matrix

For some "black-box" components, sometimes it's very difficult to obtain the reliability of the component's operation directly. We can assume that the component's failure rate λ_c obeys poisson distribution [14]. Then we can assess the component's approximate reliability in the application domain d from its average executing time in d. For an operation $a_i \in A_d$, if its execution time is $f_{TA}(a_i)$, then the reliability of executing this operation is $f_{RA}(a_i) = e^{-\lambda_c f_{TA}(a_i)}$. So formula (1) can be rewritten as follows:

$$M_C(d) = (f_{PA}(a_i, a_j) \cdot e^{-\lambda_c f_{TA}(a_i)})_{ij} \quad (3)$$

The average execution time of component in this application domain can be computed as follows: Construct the matrix of state transition $M_t(D) = ((f_{PA}(a_i, a_j)))_{ij}$, let $Q_t = \sum_{k=0}^{\infty} M_t^k$, then $Q_t = (I - M_t)^{-1}$, so the average execution time of the component in application domain d is:

$$t_d = \sum_{i \in |A_d|} Q_t(1, i) f_{TA}(a_i).$$

For $\forall a_i, a_j$, if $f_{PA}(a_i, a_j) = 1$ or 0, then we can get the application's exact execution path. Suppose the path is $< a_1, a_2, \ldots, a_n >$, then the execution time with this path is $t = \sum_{1 \le i \le n} f_{TA}(a_i)$. If the failure rate of component C is λ_C, then the reliability of this execution path is

$$r_p = \prod_{1\le i\le n} f_{RA}(a_i)\,, \qquad (4)$$

if $f_{RA}(a_i) = e^{-\lambda_c f_{TA}(a_i)}$, then $r_p = e^{-\lambda_c \sum_i f_{TA}(a_i)}$

3.2 Reliability analysis on the connections

For a software system built upon the components distributed in open environments, we have to call them based on certain connection mechanisms such as Client /Server mechanism (i.e. RPC, RMI), mobile-agent mechanism, or others. In this paper, we just discuss the client-server, and mobile-agent mechanisms. In the client-server mechanism, each time the calling to the component will traverse through the network. While in the mobile-agent mechanism, the mobile agent can migrate to the physical node where the component resides and calls it locally. So in mobile-agent mechanism, the remote calls to the component are translated into local calls between the agent and the component, and the mechanism can work well even when network has been disrupted by some unknown factors.

We discuss the reliabilities of the connections to components in the two different mechanisms above. Let B represent the bandwidth of the network, λ_N represent the failure rate of network, D_r represent the data that needs to be transported over the network when the application calls the component using the mobile-agent mechanism, D_{ag} represent the data of a mobile agent itself that needs to be transported over the network when the agent migrates to another node, N represent the total times for calling the component, and D_i represent the data that needs to be transported over the network for the ith time when the application calls the component using the client-server mechanism. Then the reliability of the connection to calling the component using the client-server mechanism is

$$RL(CS) = \prod_{1\le i\le N} e^{-\lambda_N \cdot D_i / B} \qquad (5)$$

And the reliability of the connection to calling call the component using the mobile-agent mechanism to is

$$RL(Ag) = e^{-\lambda_N (D_{ag}+D_r)/B} \qquad (6)$$

Let $q = RL(CS)/RL(Ag) = e^{\lambda_N((D_r+D_{ag})/B - \sum_{1\le i\le N} D_i/B)}$. If $D_r + D_{ag} < \sum_{1\le i\le N} D_i$, then $q < 1$. And it shows that if the total times for calling the component are large, then the connection using mobile-agent mechanism can be more reliable. If $D_r + D_{ag} > \sum_{1\le i\le N} D_i$, then $q > 1$. And it shows that if D_{ag} is large, then using client-server mechanism will be more reliable.

3.3 Reliability analysis on architecture styles

The reliability of component-based software system depends not only on the reliability of each component, the reliabilities of the connections to these components, but also the reliability of the architecture style.

Based on works [12], [16] and [15], our reliability analysis on architecture styles are shown in following:

(1) Sequence style. Suppose that two components C_1 and C_2 are composed into this style, then it can be denoted as $C_1;C_2$. In this style, C_1 will be executed first and C_2 will be executed next. Let $RL_{\to C_2}$ represent the reliability of the connection to C_2, then the reliability of this style is $r_{C_1;C_2} = r_{C_1} RL_{\to C_2} r_{C_2}$. Here, r_{C_1} is the reliability that C_1 shows in this application domain, and r_{C_2} is the reliability that C_2 shows in this application domain.

(2) Loop style. In this style C_1 will be executed repeatedly for several times, and the style is denoted as μC_1. Let μ represent an iteration operator, and suppose that the total times for executing C_1 are n, then the reliability of this style is $r_{\mu C_1} = (RL(CS) r_{C_1})^n$ (the connection to C_1 using the client-server mechanism), or $r_{\mu C_1} = RL(Ag)(r_{C_1})^n$ (the connection to C_1 using the mobile-agent mechanism).

(3) Concurrency style. This style is denoted as $C_1 \|_A C_2$. It represents that the components C_1 and C_2 are performed independently from each other with possibilities of communication over the set A. Let $RL_{C_1 \leftrightarrow C_2}(A)$ represent the reliability of the connection between C_1 and C_2. Then the reliability of this style is $r_{C_1\|_A C_2} = r_{C_1} RL_{C_1\leftrightarrow C_2}(A) r_{C_2}$.

(4) Fault-tolerant style. The style can be denoted as $C_1 | C_2$. It means that C_1, C_2 are performed in parallel to provide the same service function. If any one of them can complete successfully, then the execution of the composition can be completed. Let $RL_{\to C_1}$ represent the reliability of the connection

to C_1 , and $RL_{\to C_2}$ represent the reliability of the connection to C_2 , then the reliability of the architecture style is $r_{C_1|C_2}$ =1 − (1 − $RL_{\to C_1} r_{C_1}$)(1 − $RL_{\to C_2} r_{C_2}$).

(5) Refinement style. This style can be denoted as $ref(C_1, a, C_2)$. It means that the composition will behave as C_1 except that execution of the operation a in C_1 will be replaced by execution of the component C_2 . Let $RL_{C_1 \leftrightarrow C_2}$ be the reliability of the connection between C_1 and C_2 . Then the reliability of this style is $r_{ref(C_1,a,C_2)} = (I_{|N|} - M_{C_1}'(1, k))$, here $I_{|N|}$ is an identity matrix, and $N = |A_{C_1}(d)|$. $|A_{C_1}(d)|$ is the number of operations of C_1 included in this application domain d.

$M_{C_1}' = (m_{ij}')_{ij}$ and

$$m_{ij}' = \begin{cases} f_{PA}(a_i, a_j) f_{RA}(a_j), if \cdot a_j \neq a \\ f_{PA}(a_i, a_j) RL_{C_1 \leftrightarrow C_2} r_{C_2}, if \cdot a_j = a \end{cases} i, j \leq |S|$$

The above five basic style can also be further composed into some more complex styles.

3.4 Reliability analysis on the software system

Based the above analysis, we can present the approach to reliability analysis on the overall software system:

1) Construct the transition models of each component through their interfaces interacting with others in the application.

2) Establish the transition probabilities among the components in the application.

3)Establish the reliabilities of the connections to each component in the application.

4) For the components composed together with a certain architecture style, analyze the reliability of the style and regard the architecture as a composite component.

5) Construct the control transition matrix of the software system, and count the reliability of the whole system.

Suppose the transition matrix of the system is:

$$M_S = ((m_{ij}))_{ij} \quad (9)$$

and:

$$m_{ij} = \begin{cases} 0, C_j \cdot will \cdot not \cdot be \cdot executed \cdot after \cdot executing \cdot C_i \\ PT_{i \to j} RL_{i \to j}^{\to j} r_{i \to j}^{j}, C_j \cdot may \cdot be \cdot executed \cdot after \cdot executing \cdot C_j \end{cases}$$

Here, $PT_{i \to j}$ represents the probability that application may execute C_j after executing C_i . For any component C_i , if $C_i \neq C_f$, then $\sum_{C_j \in SC} PT_{i \to j} = 1$. $RL_{i \to j}^{\to j}$ is the reliability of the connection to C_j in the situation that C_j will be executed after C_i is executed. $r_{i \to j}^{j}$ is the reliability C_j shows in this application domain when C_j will be executed after C_i is executed.

Suppose the number of components in the application system is n, C_1 is the first component executed and C_k is the final component executed. The reliability of the system is:

$$R_S = r_{C_1} (I_{|n|} - M_S)^{-1}(1, k) \quad (10)$$

If application uses a client-server mechanism to call these components, client application will make remote communication connections to the object nodes where the components reside to call them one by one. In this scenario, the control-transition matrix of the system is: $M_S = ((m_{ij}))_{ij}$,

and

$$m_{ij} = \begin{cases} 0, C_j \cdot will \cdot not \cdot be \cdot executed \cdot after \cdot executing \cdot C_i \\ PT_{i \to j} RL_{i \to j}^{0 \to j}(CS) r_{i \to j}^{j}, or \cdot others \end{cases} \quad (11)$$

Here $RL_{i \to j}^{0 \to j}(CS)$ is the reliability of the connection to component C_j from physical node 0 (suppose the client application is on the physical node 0). Then the reliability of the component-based software system is as follows:

$$R_S = RL_{0 \to 1}^{0 \to 1}(CS) r_{C_1} (I_{|n|} - M_S)^{-1}(1, k) \quad (12)$$

If application uses the mobile-agent mechanism to call these components, client application will send out a mobile agent to the remote object nodes where the components reside to call them one after one. In this scenario, the control-transition matrix of the application system is:

$M_S = ((m_{ij}))_{ij}$, and

$$m_{ij} = \begin{cases} 0, C_j \cdot will \cdot not \cdot be \cdot executed \cdot after \cdot executing \cdot C_i \\ PT_{i \to j} RL_{i \to j}^{i \to j}(Ag) r_{i \to j}^{j}, or \cdot others \end{cases} \quad (13)$$

Here $RL_{i \to j}^{i \to j}(Ag)$ represents the reliability of the connection to the component C_j using mobile-agent mechanism. And the reliability of this system can be computed as follows:

$$R_S = RL_{0 \to 1}^{0 \to 1}(Ag) r_{C_1} (I_{|n|} - M_S)^{-1}(1, k) \quad (14)$$

3.5 Sensitivity analysis

The reliability of a component-based software system will become higher through the improvement of some elements in the system. Finding out these elements and improving their reliabilities will be benefit to the whole system. Sensitivity analysis [8] presents an approach to this problem by studying the effect of changes in the reliability of the element on the expected overall reliability of the system. In this section, we make sensitivity analysis on the reliabilities of components and connections to know which element affects the reliability of the system most.

In evaluating the sensitivity of a component's reliability, current approaches [8, 17] often study the effect of changes in the component's reliability r_i on the system's reliability R_S. Obviously, the component whose r_i has more effects on R_S is more important. Nevertheless, this approach assumes that a component has the same reliability in all its application domains, which is $\forall d_j, d_k \in D_{C_i}$, $r_i(d_j) = r_i(d_k)$. As we say above, in open distributed environments, a component can show different reliabilities in its different application domains. So we need some new approach to this problem. Here, we present a new approach to sensitivity analysis on the reliability of a component based on its failure rate λ_i:

$$SE_{\lambda_i} = \frac{|R_S(\lambda_1,..,\lambda_i + \Delta\lambda_i,..) - R_S(\lambda_1,..,\lambda_i,...)| / R_S(\lambda_1,..,\lambda_i,...)}{\Delta\lambda_i / \lambda_i}$$

In this formula, the approximate reliability of the component C_i in one of its application domain depends on the average execution time in this domain and its failure rate λ_i. So the component whose λ_i affects the changes in the SE_{λ_i} most is the most important.

When evaluating the effect of a connection's reliability $RL_{i\to j}^{\to j}$ between component C_i and C_j, we present the following formula:

$$SE_{RL_{i\to j}^{\to j}} = \frac{\Delta R_S / R_S}{\Delta RL_{i\to j}^{\to j} / RL_{i\to j}^{\to j}}$$

Also improving the reliability of the connection $RL_{i\to j}^{\to j}$ that affects the changes in the $SE_{RL_{i\to j}^{\to j}}$ more is greater to the improvement of the system's reliability.

4. Experiment analysis

In this section, we make experimental analysis on the reliabilities of the connections to components, and give an example to illustrate our approach to reliability and sensitivity analysis discussed in section 3.

We analyze the effects of different parameters (B, λ_N, D_{ag}, N) on the reliability of the connections to the components using two different mechanisms and the results are shown in figure.1-3. Table 1 lists the values of input parameters:

Table1 Input parameters

Parameters	Value
Failure rate of network λ_N	(0.005, 0.2)
Bandwidth of network B	(10k/s,1000k/s)
Average data of transferring a mobile agent D_{ag}	(1KB, 100KB)
Average data of transferring a call D	(500B, 50KB)

Fig.1 and 2 show the effect of B and D_{ag} on the two different mechanisms. In the experiment, we fix D_{ag} =10K, 50K, $D_i = D =$ 0.1K, λ_N =0.5. From the figures, we can see that when the value of B is not large, using the C/S mechanism will be more reliable. The reason is that in the network with a low bandwidth, the migration of mobile agent itself will cost much time. With the increasing of bandwidth, the time for migration of mobile agent becomes low and the reliability of this mechanism increases a lot. Figure.3 shows the test on parameter N. In this test, we set D_{ag} = 5K, D = 0.1K, λ_N =0.5, and let N be 10, 20 and 30. From the figure, we can see that the reliability of C/S mechanism comes down when the value of N increases. And this is for the reason that the number of traversing through network has increased too. While the reliability of mobile-agent mechanism doesn't become low, for the reason that it migrates to the physical node and makes local calls to the component. So the unreliable factors when traversing through network have been avoided in the mobile-agent mechanism. From the tests, we can see that if B is not high and D_{ag} is large, using the C/S mechanism will be more reliable. On the other hand, if B is high and D_{ag} is large, using the mobile-agent mechanism will be more reliable.

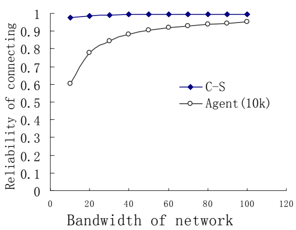

Fig.1 The effect of D_{ag} (1)

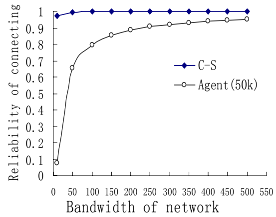

Fig.2 The effect of D_{ag} (2)

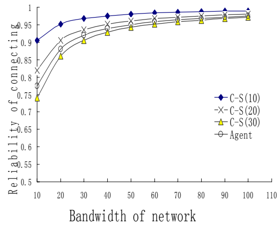

Fig.3 The effect of N

Next, we use an example adapted from [12], [17], and [18] to show the approach discussed in section 3. The component-based system consists of thirteen components, among which the components C_6, C_{61} and C_{62} are composed into a certain

C_i	1	2	3	4	5	6	7	8	9	10
r_i	0.99	$r^2_{1\to2}$:0.99 $r^2_{7\to2}$:0.98	$r^3_{1\to3}$:0.99 $r^3_{2\to3}$:0.98 $r^4_{6\to3}$:0.99	$r^4_{1\to4}$:0.97 $r^4_{8\to4}$:0.96	$r^5_{2\to5}$:0.98 $r^5_{3\to5}$:0.97 $r^5_{4\to5}$:0.99	$r^6_{4\to6}$:0.95	$r^7_{5\to7}$:0.98 $r^7_{6\to7}$:0.97	$r^8_{5\to8}$:0.96 $r^8_{6\to8}$:0.98 $r^8_{9\to8}$:0.97	$r^9_{6\to9}$:0.97 $r^9_{7\to9}$:0.98	$r^{10}_{8\to10}$:0.99 $r^{10}_{9\to10}$:0.98
λ_i		0.051	0.054	0.050	0.052	0.057	0.053	0.052	0.056	
$PT_{i\to j}$		$PT_{1\to2}$: 0.6 $PT_{7\to2}$:0.5	$PT_{1\to3}$:0.2 $PT_{2\to3}$:0.7 $PT_{6\to3}$:0.3	$PT_{1\to4}$:0.2 $T_{8\to4}$:0.95	$PT_{2\to5}$: 0.3 $PT_{3\to5}$:1.0 $PT_{4\to5}$:0.4	$PT_{4\to6}$:0.6	$PT_{5\to7}$:0.4 $PT_{6\to7}$:0.3	$PT_{5\to8}$:0.6 $PT_{6\to8}$:0.1 $PT_{9\to8}$:0.1	$PT_{6\to9}$:0.3 $PT_{7\to9}$:0.5	$PT_{8\to10}$:0.75 $PT_{9\to10}$:0.9
$RL^{\to j}_{i\to j}$		$RL^{\to2}_{1\to2}$: 0.9 $RL^{\to2}_{7\to2}$:0.99	$RL^{\to3}_{1\to3}$:0.93 $RL^{\to3}_{2\to3}$:1 $RL^{\to3}_{6\to3}$:0.99	$RL^{\to4}_{1\to4}$:0.97 $RL^{\to4}_{8\to4}$:0.95	$RL^{\to5}_{2\to5}$: 0.98 $RL^{\to5}_{3\to5}$:0.99 $RL^{\to5}_{4\to5}$:0.96	$RL^{\to6}_{4\to6}$:0.98	$RL^{\to7}_{5\to7}$:0.99 $RL^{\to7}_{6\to7}$:0.92	$RL^{\to8}_{5\to8}$:0.97 $RL^{\to8}_{6\to8}$:0.97 $RL^{\to8}_{9\to8}$:0.94	$RL^{\to9}_{6\to9}$:0.97 $RL^{\to9}_{7\to9}$:0.96	$RL^{\to10}_{8\to10}$:0.92 $RL^{\to10}_{9\to10}$:0.98

Table 2 All the values of parameters needed

architecture style as $C_6 \|_A (C_{61} | C_{62})$, which means that C_{61}, C_{62} are composed with a fault-tolerant style, and at the same time, C_6 and the assembly of C_{61} and C_{62} are composed together by communicating through the interface A. Suppose the reliabilities of the components C_{61}, C_{62}, C_6 in their application domains is 0.84, 0.86, and 0.97 respectively. The reliabilities of the connections to C_6 and C_{61}, C_{62} are 1. From the analysis presented in section 3.3, we can compute the reliability of the assembly $C_6 \|_A (C_{61} | C_{62})$ to be 0.95. Next, we regard the assembly as a composite

component C_6' in the system. In this component-based system, the reliabilities of the components in different application domains, the connections to these components, and the probabilities of transitions among the components are given in table 2.

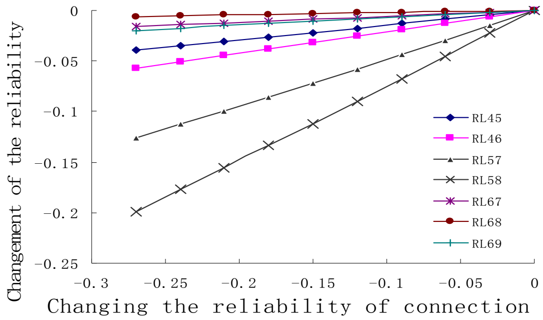

Fig.8 Sensitivity analysis on the reliabilities of the connections(2)

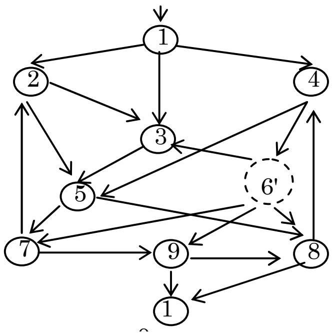

Fig.4 The component-based application system

We construct the transition matrix M_S as follows:

$$M_S = \begin{bmatrix} 0 & 0.99\cdot0.6\cdot0.9 & 0.99\cdot0.2\cdot0.93 & 0.97\cdot0.2\cdot0.97 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.98\cdot0.7\cdot1 & 0 & 0.98\cdot0.3\cdot0.98 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.97\cdot0.99 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.99\cdot0.4\cdot0.96 & 0.95\cdot0.6\cdot0.98 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.98\cdot0.4\cdot0.99 & 0.96\cdot0.6\cdot0.97 & 0 & 0 \\ 0 & 0 & 0.99\cdot0.3\cdot0.99 & 0 & 0 & 0 & 0.97\cdot0.3\cdot0.92 & 0.98\cdot0.1\cdot0.97 & 0.97\cdot0.3\cdot0.97 & 0 \\ 0 & 0.98\cdot0.5\cdot0.99 & 0 & 0 & 0 & 0 & 0 & 0 & 0.98\cdot0.5\cdot0.96 & 0 \\ 0 & 0 & 0 & 0.96\cdot0.25\cdot0.95 & 0 & 0 & 0 & 0 & 0 & 0.99\cdot0.75\cdot0.92 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.97\cdot0.1\cdot0.94 & 0 & 0.98\cdot0.9\cdot0.98 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Let $Q_S = \sum_{k=0}^{\infty} M_S^k$, so the reliability of the whole application system is $= 0.6704$.

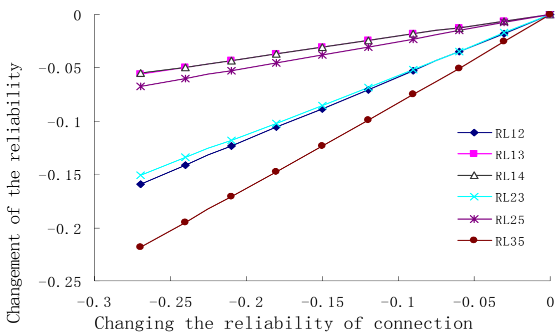

Fig.7 Sensitivity analysis on the reliabilities of the connections(1)

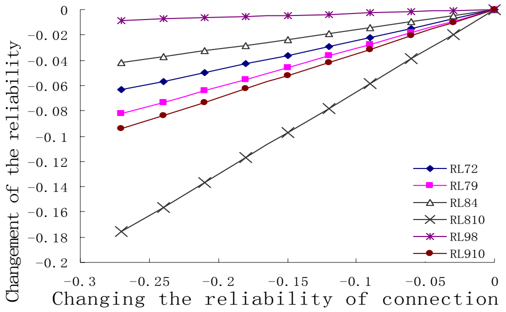

Fig.9 Sensitivity analysis on the reliabilities of the connections(3)

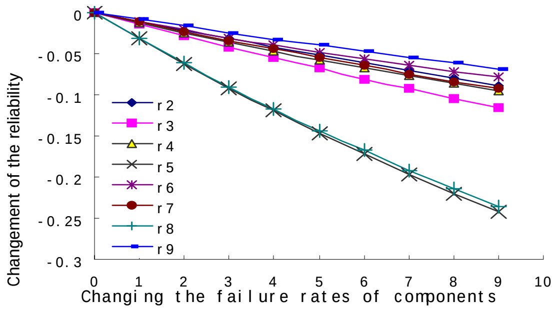

Fig.10 Sensitivity analysis on the reliabilities of the components

Fig.7-9 show our experimental sensitivity analysis on the reliabilities of the connections and components. As shown in fig.7-9, we can find that $RL_{3\to5}^{\to5}$, $RL_{5\to8}^{\to8}$ and $RL_{8\to10}^{\to10}$ have more effects on the reliability of the system than other connections. As for the connection $L_{3\to5}^{\to5}$, when its reliability $RL_{3\to5}^{\to5}$ decreases 3%, the reliability of the system decreases about 2.4%. Thus how to increase the reliabilities of these connections is more important. While for the connection $L_{6\to7}^{\to7}$, when its reliability $RL_{6\to7}^{\to7}$ decreases 20%, the reliability of the overall system decreases only 0.47%. We can also see that the sensitivity of $RL_{i\to j}^{\to j}$ is influenced by the sensitivity of $r_{i\to j}^{j}$ and the probability $PT_{i\to j}$: 1) As for $RL_{i\to j}^{\to j}$, if the reliabilities of the components *i*, *j* are more sensitive, then the reliability of this connection will be more sensitive (such as $RL_{5\to8}^{\to8}$); 2) If the $PT_{i\to j}$ relative to this connection is high, this connection will also be more sensitive. In fig.9, we make sensitivity experiments on the reliabilities of the components 2,3,4,5,6,7,8 and 9. As for the components having different application domains, we use the failure rate λ_i of the component to illustrate its sensitivity on the reliability of the system. All the failure rates of the components are given in table 2. From the figure, we can see that the components 5 and 8 are more sensitive to the reliability of the system, so improving the reliabilities of these two components are more important to the reliability of the system.

5. Conclusion

In the open, dynamic and uncertain environment, component-based software system may consist of self- contained, autonomous entities situated in distributed nodes of the Internet and coordinators connecting these entities statically and dynamically in various kinds of interaction styles (passively and actively). Making reliability analysis on this kind of component-based software system has important meanings. This paper presents a new approach to analyze the reliability of the software system in open distributed environments, based on the reliabilities of the individual components in different application domains, the connections to the components and the architecture styles of their composition. It will be applicable to developing a more reliable software system built on the components in Internet.

References

[1] FuQing Yang. Thinking on the development of software engineering technology. *Journal of Software*, 2005,16(1): 1~7(in Chinese with English abstract).

[2] FuQing Yang, Hong Mei, Jian Lv, Zhi Jin. Some discussion on the development of software technology. *Acta Electronica Sinica*, 2002, 30(12A): 1901-1906(in Chinese with English abstract).

[3] Jian Lu, XianPing Tao, XiaoXing Ma, Hao Hu, Feng Xu, Chun Cao. On Agent-Based Software Model for Internetware. *Science in China*(Series F), 2005, 35(12):1~12.

[4] Goseva-Popstojanova K, Trivedi K. Architecture-Based approach to reliability

assessment of software systems. *Performance Evaluation*, 2001, 45(2-3):179~204.

[5] Ledoux J. Availability modeling of modular software. *IEEE Transactions on Reliability*, 1999,48(2): 159~168.

[6] Gokhale SS, Lyu MR. A simulation approach to structured –based software reliability analysis. *IEEE transactions on software engineering*. 2005, 31(8): 643~656.

[7] Yacoub SM, Ammar HH. A methodology for architecture-level reliability risk analysis. *IEEE Transactions on Software engineering*. 2002,28(6):529~547.

[8] Lo JH, Huang CY, , Chen IY, Kuo SY, Lyu MR. Reliability assessment and sensitivity analysis of software reliability growth modeling based on software module structure. *Journal of systems and software*, 2005, 76:3~13.

[9] Reussner RH, Schmidt HW, Poernomo IH. Reliability prediction for component-based software architectures. *Journal of systems and software*, 2003(66): 241~252.

[10] Cheung RC. A user-oriented software reliability model. *IEEE Transactions on software Engineering*. 1980,6(2):118-125.

[11] Hamlet D. Are we testing for true reliability? *IEEE Software*, 1992,13(4)21-27.

[12] Yang WL, Wu Y, Chen MH. An architecture-based software reliability model. Proceedings of Pacific Rim International Symposium on Dependable Computing. Hong Kong, China, 1999, IEEE.

[13] Cheung RC. A user-oriented software reliability model. *IEEE Transactions on Software Engineering*, 6(2):118–125, March 1980. Special collection from COMPSAC '78. Probability and statistics with reliability, queuing, and computer science applications. John Wiley & Sons, INC, New York, 2002.

[14] Hamadi R, Benatallah. A Petri Net-based model for Web services composition. 14th Australasian Database conference, Adelaide, Australia. 2003.

[15] Hamlet D, Mason D, Woit D. Theory of software reliability based on components. 3th international workshop on component-based software engineering. Toronto. IEEE computer society. 2001. 361-370

[16] Lo JH, Huang CY, Kuo SY, Lyu MR. Sensitivity analysis of software reliability for component-based software applications. Proceedings 27th Annual International Computer Software and Applications Conference (COMPSAC '2003), Dallas, Texas, November 3-6 2003, pp. 500-505.

[17] Gokhale SS, Trivedi KS. Reliability prediction and sensitivity analysis based on software Architecture. 13th International symposium on software reliability engineering. 2002.

[18] XiaoGuang Mao, YongJin Deng. A general model for component-based software reliability. *Journal of software*, 2004,15(1):27-32.(in Chinese with English abstract).

HaiYang Hu received the B.E. and M.E. degrees, from NanJing Univ. in 2000 and 2003, respectively. He received the Dr. Eng. degree from NanJing Univ. in 2006. He has been working as an assistant professor in the Dept. of Computer and Information Engineering, ZheJiang Gongsang Univ., from 2006. His research interest includes Object-oriented technology, Network computing.