A Reliability-oriented Evolution Method of Software Architecture Based on Contribution Degree of Component

Jun Wang
Shenyang University of Chemical Technology, Shenyang, 110142, China
Email: wj_software@hotmail.com

WeiRu Chen
Shenyang University of Chemical Technology, Shenyang, 110142, China
Email: wangjundmu2010@gmail.com

***Abstract*—Owing to the low cost, high profit and disaster-free, the reliability-oriented evolution method is proposed in this paper. The method improves the reliability of software architecture by analysis contribution degree of component. Because the various components will play different roles in the reliability-oriented evolution of software architecture, the contribution degree of component is first put forward. It will be used for the reliability-oriented evolution in this paper. At the same time, the reliability-oriented evolution method of software architecture based on contribution degree of component is applied to an ATM system. The evolution process is illustrated by the method, and it shows that which component is playing an important or crucial role in the process of reliability-oriented evolution of software architecture. The reliability of software architecture will can be improved effectively and fleetly by the result.**

***Index Terms*—contribution degree, software architecture, reliability, evolution**

I. Introduction

Software evolution is a set of activities which improve software after it is delivered to customer, it is a process which generates a new software version from an earlier version, and it is one important area of software engineering. A failure operation of software can lead to economic loss and even cause loss of human lives. Therefore, unreliable software is not acceptable and should be improved [1-3]. For improving overall system reliability, the component and the architecture of system will be evolved. When the reliability of some components can not be increased, or when you require very high expenses to increase reliability a little, the reliability-oriented evolution of software architecture maybe another solution. This is the reliability-oriented evolution of software architecture.

How to improve the efficiency of reliability-oriented evolution? Some scholars proposed a lot of methods based on project management. Others researched that how to enhance the reliability of each component. Another one set up some models to calculate reliability of software architecture, and improve reliability by changing architecture. We combine the latter two results and propose a reliability-oriented evolution method in this paper. The method improves the reliability of software architecture by analysis contribution degree of component. The method is more effective and faster than some methods that consider only one aspect [4-7].

The discussion of this topic takes the following structure. Section II describes the our reliability model based on software architecture. In section III, we compare the model with the classic model. Section IV proposes the decision rule on contribution degree of component in the process of reliability-oriented evolution of software architecture. In section V, the contribution degree of component is analyzed, and a process of evolution is completed and showed. Conclusions are provided in section VI.

II. Reliability Model Based on Software Architecture

Currently, most of the reliability models based on software architecture use the idea proposed by Cheung et al [8-10]. These models estimate reliability by the definition of software reliability. It is based on properties of Markov chain, and the conversion from software architecture view to state transition view. According to state transition view, the state transition matrix is constructed. Of course, the reliability of each component and the operational profile will be considered too.

A. *Reliability Calculation Steps*

The reliability model based on software architecture usually utilizes Markov chain to compute reliability of system [11]. On the basis of properties of Markov chain, the state transition is assumed as a Markov process [12-13].

(a)The state transition diagram is usually used to depict system behavior. The node S_i represents system state i, and a directed edge (S_i, S_j) represents the state transition from S_i to S_j. The Fig.1 shows the state transition diagram.

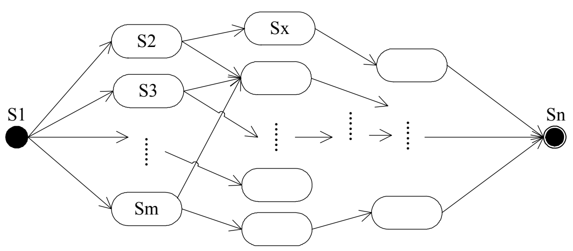

Figure 1. The state transition diagram

(b)The value of *M (i, j)* indicates the possibility of transition from state S_i to state S_j. A state transition matrix *M* can be defined by the state transition diagram. Therefore, the reliability *R* is represented as follow:

$$R = M(i,j) \times R_j \tag{1}$$

If the value of *M (i, j)* is computed and gained, the reliability of software architecture will be achieved too. Let array *T*, $T=\{R_{1\sim n}, R_{2\sim n}, R_{3\sim n},\ldots\ldots, R_{n-1\sim n},1\}$, be a set of reliability in software architecture. The $R_{i\sim j}$ represents reliability from state S_i to state S_j. According to matrix theory and reliability definition of software architecture, the equation, $T^T = M \times T^T$, will come into existence.

(c)The $R_{1\sim n}$ is the first vector in the array *T*. The value of $R_{1\sim n}$ equals to reliability of software architecture.

B. Constructing Transition Matrix

How to compute the value of *M (i, j),* it is important to obtain reliability of software architecture. In consideration of the reliability of components, connectors and different architectural styles, the transition matrix *M* can be obtained as follow [9]:

The l_{ij} denotes the reliability of connector from state S_i to state S_j. The l_{nij} denotes the reliability of connector from state S_i to state S_j , and there have *n* components in state S_i. The l'_{nij} denotes the reliability of connector from state S_i to state S_j , and there have *n* components in state S_j. The P_{ij} represents the transition probability from state S_i to state S_j. The *n* denotes the number of components in a state.

(a) Sequence style: Components are executed in a sequential order.

When state S_i can arrive at state S_j directly:

$$M(i,j) = R_i l_{ij} P_{ij} \tag{2}$$

When state S_i can not arrive at state S_j directly:

$$M(i,j) = 0 \tag{3}$$

(b) Parallel style: Components are commonly running simultaneously to improve performance.

When state S_i contains components of running simultaneously:

$$M(i,j) = \left(\prod_{n=1}^{N} (R_n l_{nij})\right) P_{ij} \tag{4}$$

When state S_j contains components of running simultaneously:

$$M(i,j) = R_i P_{ij} \left(\prod_{n=1}^{N} l'_{nij}\right) \tag{5}$$

(c) Redundancy style: It has a set of components. Some of components will start to work after the primary component fails. When the primary component fails, the first backup component will take over the responsibility and become a new primary component. If the new one fails too, another one will take over its responsibility. *M (i, j)* can be constructed as follows:

When state S_i contains a set of backup components:

$$M(i,j) = \left(R_i l_{1ij} + \sum_{q=2}^{N} \left(\left(\prod_{m=2}^{q} (1 - R_i l_{mij})\right) R_i l_{qij}\right)\right) P_{ij} \tag{6}$$

When state S_j contains a set of backup components:

$$M(i,j) = R_i P_{ij} \left(l'_{1ij} + \sum_{q=2}^{N} \left(\left(\prod_{m=2}^{q} (1 - l'_{mij})\right) l'_{qij}\right)\right) \tag{7}$$

III. Comparison with Classic Model

We respectively used our model and classic model to estimate the reliability of an ATM banking system, which is provided by W.L. Wang [14-15]. The result of classic model is compared with our reliability model based on software architecture in this section.

In this ATM banking system, two architectural types are defined, including sequence style and redundancy style. Fig.2 shows the architecture of system. The component *Start* is the initial component, and the component *End* is an end component. In addition to the following conditions, the system is basically run by sequence. Component *DBMS1* and component *DBMS2* are defined as redundancy style. The component *DBMS2* is backup component of *DBMS1*. It is used to increase fault tolerance.

The reliability of components in single state:

R_1=1.0;
R_2=0.982;
R_4=1.0;
R_5=0.996;
R_6=1.0;
R_7=0.8999;
R_8=0.99;
R_9=1.0;
R_{10}=1.0;

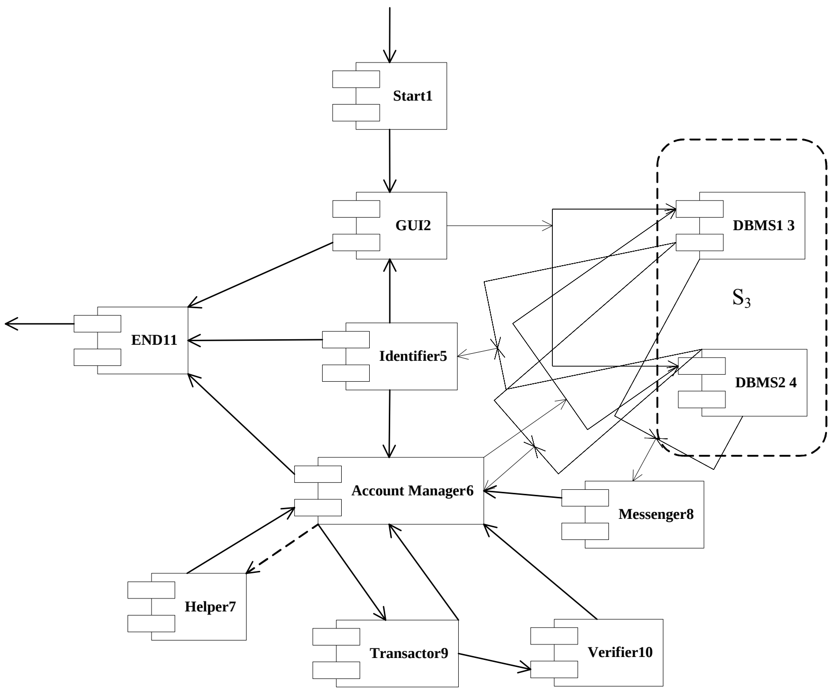

Figure 2. The architecture of system

The reliability of components in state S_3 of redundancy style:

r_{31}=0.97;
r_{32}=0.96;

The reliability of all connectors:

l_{135} =0.97;
l_{139}=0.97;
l_{235}=0.96;
l_{239}=0.96;
l'_{123}=0.97;
l'_{223}=0.96;

Others are "1".

The transition probability:

P_{12}=1.0;
P_{23}=0.999;
$P_{2\,10}$=0.001;
P_{39}=0.227;
P_{35}=0.669;
P_{34}=0.104;
P_{92}=0.048;
P_{95}=0.951;
$P_{9\,10}$=0.001;
P_{53}=0.4239;
P_{58}=0.1;
P_{56}=0.4149;
$P_{5\,10}$=0.0612;
P_{85}=1.0; P_{45}=1.0;

Others are zero.

The transition matrix M can be constructed by our model as follow:

$$M=\left\{\begin{matrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0.9798 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0001 \\
0 & 0 & 0 & 0.2259 & 0.6659 & 0 & 0.1035 & 0 & 0 & 0 \\
0 & 0.0480 & 0 & 0 & 0.9510 & 0 & 0 & 0 & 0 & 0.0010 \\
0 & 0 & 0.4217 & 0 & 0 & 0.1000 & 0 & 0.4132 & 0 & 0.0609 \\
0 & 0 & 0 & 0 & 0.9900 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0.0100 & 0 & 0 & 0 & 0.9900 & 0 \\
0 & 0 & 0 & 0 & 0.8999 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{matrix}\right\}$$

The transition matrix M' can be constructed by classic model as follow [14]:

$$M'=\left\{\begin{matrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0.981018 & 0 & 0 & 0 & 0 & 0 & 0 & 0.000982 \\
0 & 0 & 0 & 0.2267 & 0.6682 & 0 & 0.1039 & 0 & 0 & 0 \\
0 & 0.0480 & 0 & 0 & 0.9510 & 0 & 0.104 & 0 & 0 & 0.0010 \\
0 & 0 & 0.4222 & 0 & 0 & 0.1000 & 0 & 0.4132 & 0 & 0.061 \\
0 & 0 & 0 & 0 & 0.9900 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0.0100 & 0 & 0 & 0 & 0.9900 & 0 \\
0 & 0 & 0 & 0 & 0.8999 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{matrix}\right\}$$

We respectively used above two transition matrix M to estimate the reliability of an ATM banking system. The result is shown in Table 1.

TABLE I
THE RESULT

	Classic model	Improved model	Actual value
Reliability	0.5590	0.5451	0.526

VI. THE DECISION RULE ON CONTRIBUTION DEGREE OF COMPONENT

In the process of reliability-oriented evolution of software architecture, the decision rules on contribution degree of component are as follow:

(a) If the out-degree of component is larger than others, then the component is more important in the process of reliability-oriented evolution of software architecture, and its contribution degree is higher than others.

(b) The contribution degree of component may be different in different location of software architecture.

(c) The transition probability of component is one of crucial factors of their contribution degree in software architecture.

The above is qualitative analysis on contribution degree of component, but it is necessary. Because the contribution degree of all components will be analyzed, this will be a complex and huge projects.

If you want to know the contribution degree of every component, the relationship function between components and system must be calculated. Using the *Matlab*, we may simulate the variation curve of component reliability. Of course, we can also obtain the variation curve of system reliability when the reliability of other components is unchanged. Finally, contribution degree of every component will be obtained. In the process of reliability-oriented evolution of software architecture, we will find that which component is the biggest evolutionary relationship with software architecture by their contribution degree.

V. Analysis of Contribution Degree

The transition matrix M is constructed by the above method. The relationship function between components and system may be calculated by the definition of reliability. The variation curve of component reliability is obtained by *Matlab* when the reliability of others remains stabilizing. At the same time, contribution degree of component is analyzed by the curve in this section.

A. *The Process of Reliability-oriented Evolution of Software Architecture in an ATM Banking System*

Given $T= \{R_{1\sim10}, R_{2\sim10}, R_{3\sim10}, R_{4\sim10}, R_{5\sim10}, R_{6\sim10}, R_{7\sim10}, R_{8\sim10}, R_{9\sim10}, 1\}$, the reliability R of software architecture can be obtained by equation, $T^T = M \times T^T$. We analyze contribution degree of every component in the process of reliability-oriented evolution of software architecture. It is helpful to improve efficiency in reliability-oriented evolution of software architecture.

(a) Analysis of contribution degree by means of the out-degree and transition probability

We find that out-degree and transition probability of component 6 (*Account Manager*) are the largest by statistics. The variation curve of component (*Account Manager*) reliability is established, and the image is shown in Fig. 3. The larger curvature shows the contribution degree is higher in the process of reliability-oriented evolution of software architecture.

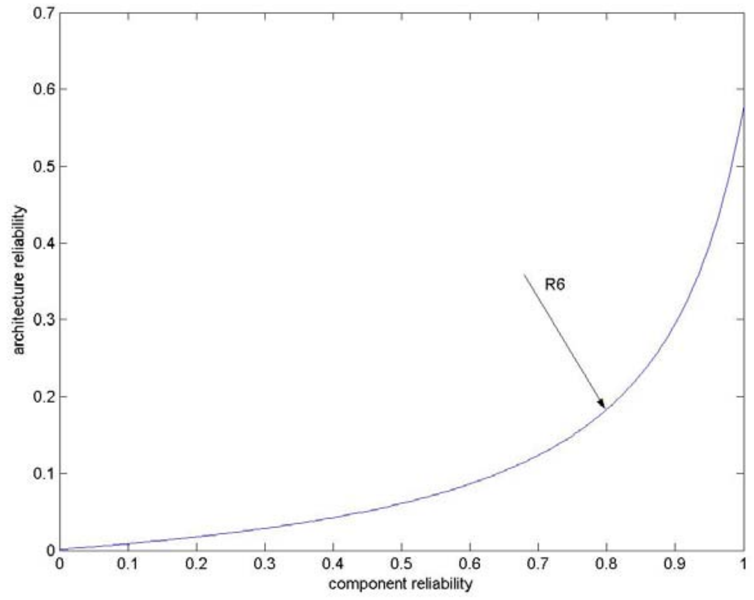

Figure 3. The evolution relationship between component 6 and the overall system

The system has a redundancy style. Through the reliability model, the variation curves of components (*6 and 7*) reliability are derived in Fig. 4, the smaller curvature shows the contribution degree of component *7* is lower than component *6* in the process of reliability-oriented evolution of software architecture.

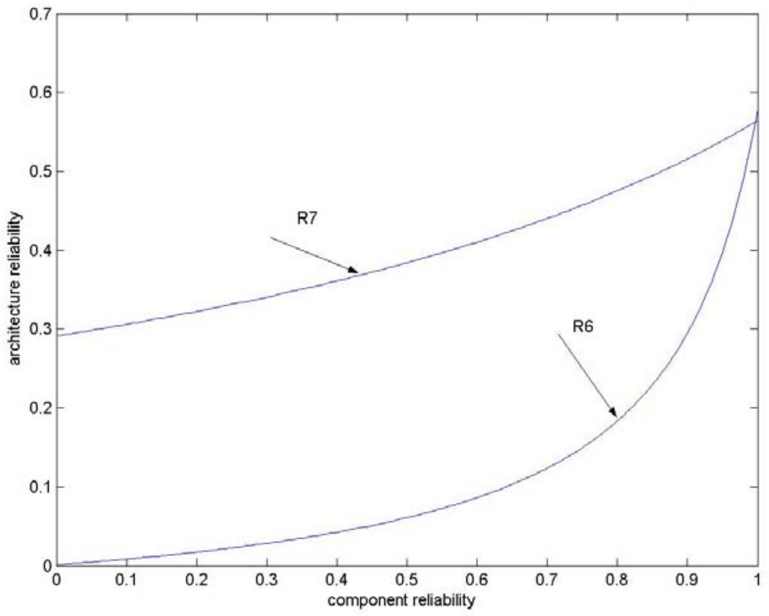

Figure 4. The evolution relationship between components 6, 7 and the overall system

(b) Analysis of contribution degree of every component

On the basis of the reliability model, the variation curves of all component reliability are derived in Fig. 5. It shows that component 6 and component 10 have larger curvature than others. So the contribution degree of component (*6 and 10*) is higher in the process of reliability-oriented evolution of software architecture. In addition, Fig. 5 shows that the reliability of system will be improved promptly when the reliability of component 10 is increased. Therefore, we analyze that the contribution degree of component 10 is the highest than others in the process of reliability-oriented evolution of software architecture.

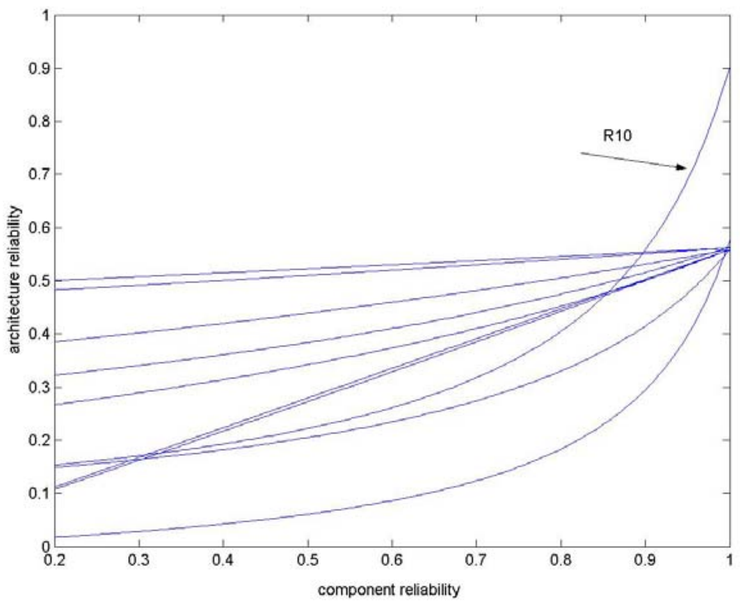

Figure 5. The evolution relationship between every components and the overall system

B. *The Reliability-oriented Evolution of Software Architecture based on Contribution Degree of Component*

The contribution degree of component 10 is the highest, and it will be a bottleneck. It will affect on reliability-oriented evolution of software architecture. Architecture (style) evolution is used in the paper. We change a single component 10 into a new component 10. It becomes a redundancy style. It is shown in Fig. 6.

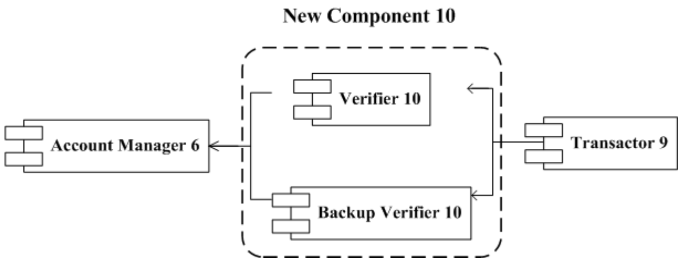

Figure 6. The new component 10

By the reliability model based on software architecture, we calculate the reliability of evolved software architecture, and it is 0.8512. The 0.5451 is the reliability of original software architecture. The reliability of system increase by 56% after the architecture is evolved.

VI CONCLUSION

We have studied architecture-based software reliability for last 6 years, and published a number of papers. We use these results for reliability-oriented evolution of software architecture today. It is a good ideal that we decide to how to evolve reliability of software architecture according to contribution degree of component. It is an old technology and is used in a new area. The method can find some bottlenecks in the process of reliability-oriented evolution of software architecture. It is an effective method of reliability-oriented evolution for software architecture.

ACKNOWLEDGMENT

This work was supported by Natural Science Foundation of Liaoning Provincial (No.20102178) and Liaoning Provincial Office of Education Fund of China (No. L2010439 and No. 2009B149).

REFERENCES

[1] M. Tom, “Guest editors' introduction: Software evolution”, IEEE Software, vol.27, No.4, 2011, pp.22-25.
[2] O. Hryniewicz, “An evaluation of the reliability of complex systems using shadowed sets and fuzzy lifetime data”, International Journal of Automation and Computing, vol.3, No.3, 2006, pp.145-150.
[3] B. Kwiatuszewska-Sarnecka. “Reliability improvement of large multi-state series-parallel systems”, International Journal of Automation and Computing, vol.3, No.2, 2006, pp.157-164.
[4] S. Gokhale, B. Mendiratta, “Architecture-Based Assessment of Software Reliability”, IEEE proceedings on QSIC, 2008, pp.444 -448.
[5] P. Nicolas, “A component-based and aspect-oriented model for software evolution”, International Journal of Computer Applications in Technology, vol.32, No.1, 2008, pp.94-105.
[6] S. Vibha, “Modeling software evolution with game theory”, Lecture Notes in Computer Science, vol. 5543, 2009, pp.354-365.
[7] P. K. Kapur, A. Gupta, P. C. Jha, “Reliability Growth Modeling and Optimal Release Policy under Fuzzy Environment of an N-version Programming System Incorporating the Effect of Fault Removal Efficiency”, nternational Journal of Automation and Computing, vol.4, No.4,2007, pp.369-379.
[8] R. C. Cheung, “A User-Oriented Software Reliability Model”, IEEE transactions on Software Engineering, 1980, pp.59-65.
[9] J. Wang, J. Liu,W.R. Chen, “Research of software reliability based on synthetic architecture”, International Conference on Computational Intelligence and Security, 2007, pp.785-788.
[10] W.L.Wang, M.H. Chen,“Heterogeneous Software Reliability Modeling” Software Reliability Engineering, Nov. 2002, pp.41-52.
[11] B. Tekinerdogan, H. Sozer, “Software Architecture Reliability Analysis Using Failure Scenarios”, IEEE proceedings on WICSA, 2005, pp.203-204.
[12] Z. Yefei, Y. Zongyuan, X. Jinkui, “Performance Analysis of System Model Based on UML State Diagrams and Continuous-time Markov Chains”, Journal of Software, vol.5, No.9, 2010, pp. 974-981.
[13] H. Pham, “Improving Energy and Power Efficiency Using NComputing and Approaches for Predicting Reliability of Complex Computing Systems”, International Journal of Automation and Computing, vol.7, No.2, 2010, pp.153-159.
[14] W.L.Wang, Y. Wu ,M. H. Chen, “Architecture-Based Software Reliability Analysis”, Proceedings of Pacific Rim International Symposium on Dependable Computing, 1999, pp.16-17.
[15] J. Wang, W.R. Chen, J. Liu, “A Modeling of Software Architecture Reliability”, IFIP International Conference on Network and Parallel Computing, 2007, pp.983-986.

Jun Wang received the B.Sc. and M.Sc. degrees in computer science from the Shenyang Institute of Chemical Technology, PRC in 2001 and 2005, respectively, and the Ph.D. degree from Shenyang Institute of Automation of CAS, PRC in 2009. Currently, he is an associate professor at Shenyang University of Chemical Technology, PRC and leads the Network Engineering Teaching and Research Group in the Department of Computer Science and Technology. Since January 2010, he has been invited as an Academic Visitor (including post-doctoral project as a post-doctor) at De Montfort University, UK.

His research interests include wireless network, software reliability in distributed computing systems and the Internet of things.

Wei-Ru Chen received the B.Sc. and M.Sc. degrees from Northeast University, PRC in 1985 and 1988, respectively. He is currently the head of the Department of Computer Science and Technology Shenyang University of Chemical Technology, PRC.

His research interests include software architecture, software reliability, and data mining.