Research on the software trustworthiness extended measurement based on IMC

Feng He[1], Haican Peng[2*], Kun Yao[3]

[1]*College of Mathematics, Physics and Information Engineering, Jiaxing University, China*

[2]*Computer Science and Engineering, Beifang University of Nationalities, China*

[3]*Hanzhong Branch Company of Shanxi Province of China Telecom China*

Received 13 May 2014, www.cmnt.lv

Abstract

This paper introduces two methods of extension measure based on model checking algorithm of interactive Markov chains (IMC) to decide the software trustworthiness. The first extended measurement is to establish multiple corresponding temporal logic relations for each software trustworthy attribute that affecting software trustworthiness, also is to use multiple temporal logic to describe a software trustworthy attribute, which is aim to measure the software trustworthiness on the multi-level and fine-grained. Then the paper will determine the measurement ultimately. The second extended measurement is to locate for the untrusted states, then find out the detail path and detail parameters of the path. Next, we will get the location that not trusted through further analysis. Eventually meet people's expectations by improving.

Keywords: software trustworthiness, model checking, finite state machine model, trustworthy attribute

1 Introduction

Trust is essential to most human transactions [1]. Numerous research papers have addressed trust and software trustworthiness from many kinds of different perspectives in recent years [2]. However, at present, the existing researches mainly focus on two aspects, which are software reliability metrics and safety assessment [3]. M. Ohba divided the software reliability model into two categories: static model and dynamic model according to the modelling object [4]. Among them, the dynamic model becomes popular and has the most researchers. It models with some data or information related to the running time. This type of dynamic model utilizes software-testing process to obtain the failure time or software failure frequency over a period of time to estimate the number of failures of the entire software and time of failure occurrence or some other data involved with software failure. This typical model is Markov Process Model [5], Non-homogeneous Poisson Process (NHPP) [6] and Bayesian Model [7]. Certainly, there are also other extended models.

In addition, software interactivity cannot be neglected any longer because of that, too many safety issues are introduced through interaction. Nonetheless, people still do not keep a watchful eye on the measurement of the software interactivity. Therefore researching on software interactive security measure is a necessary complement for software reliability measure research and also a new development. This paper will model the software interaction as the state model and utilize the model-checking algorithm, which is a formal verification by exhaustively searching the finite state automata. And then convert the verification of properties to the corresponding temporal logic, using the model checking tool to traverse system model automatically, at last, it will check whether the system meets the corresponding properties or not. Compared with ordinary artificial validation method, model checking is of speed and high accuracy and is very useful for realizing the automation. The most important is that this model-checking algorithm not merely can reflect the behaviours of the software from the angle of function layer, but also further measure the software credibility from a performance perspective. Hence, this paper selects the model-checking algorithm of the IMC model to give two extended measurement to determine the software trustworthiness.

The scope of this paper is organized as follows: Section 2 introduces related work. In Section 3 presents the model checking in detail, especially the two extended measurement methods based on IMC and will utilize the two extended methods to decide the software trustworthiness. At the end of this chapter we verify the feasibility and effectiveness of these methods by experiments. Conclusions and some directions for future research are given in Section 4. Section 5 is the acknowledgements.

* ***Corresponding author*** e-mail: phc0409@126.com

2 Related work

Many reliability models and measurement methods have been proposed to estimate the software trustworthiness up to now. However, due to the software is more complex, so now there is no authority measurement in the world. At present, the most popular methods of software reliability analysis and evaluation include based on the development model, based on the informal method, based on the software behaviour, based on the formal method and based on the model checking measurement.

Software reliability analysis and measurement based on the development model usually makes full use of various development model to guarantee the reliability of the software. [8] proposed a trusted software design and development process based on model-driven architecture (MDA) which combines the executable formal specification language with UML description method to realize the executable formal specification description in the whole software development life cycle and guarantee the credibility of the software. This method can effectively detect the software behaviour to identify whether is trustable or not. Nonetheless, there is no detailed implement process and clear instructions. [9] described the software architecture applying AC2-ADL (Architectural Description of Aspect-Oriented Systems) and proposed a kind of trusted software architecture design method supporting run-time monitoring. This way can effectively achieves the trusted software system development process, but still need to further improve and research on software credible guarantee mechanism. [10] extended the trusted chain suggested by TCG (Trusted Computing Group) based on trusted computing platform. Through the description of irregular track, it inserted the corresponding check sensor into the key code that needed to be checked to implement dynamic reliable detection at the runtime. It is based on trusted computing and has the characteristics of high formalization, but the applicable scope is small.

Based on the formalization of software reliability mainly uses artificial method to analyse software and obtain the corresponding measure matrix to evaluate software reliability. [11] puts forward to extract different attribute benchmark index to evaluate the trusted degree based on the layered mechanism. This method is applicable to large modular software system. Nevertheless, the process of classifying the software reliable properties and obtaining the corresponding indicators is not fully automated. [12] proposed a trusted software process assessment method based on the evidence. This way picks the objective data as the evaluation data. However, the corresponding metrics and algorithm still need further improvement.

In information security, the research on the behaviour of the software has always been used in intrusion detection. The theory of based on the software behaviour has been increasingly used in the dynamic measurement of trusted computing. [13] introduced a dynamic credible measurement based on software behaviour. At the same time, it puts forward an authentication mechanism based on expanded behaviour trace and behaviour measurement information. [14] used the behaviour track and checkpoints scenario to describe the dynamic characteristics, its aim is to detect the attacks. The software will stop running as long as finding any behaviour that deviation from the original expected track. Based on dynamic credible measurement, the measurements are divided into trusted or untrusted, but the credibility of software cannot be simply represented by trusted or untrusted.

Compared with the general software reliability analysis methods, the formal method based on strict mathematical foundation can carry on the formal descriptions or verification accurately and is suitable for reliability analysis and evaluation of the software.

Model checking is a kind of effective formal verification method. With the increasing development of model test technology, more and more researchers will apply the model checking technique to property verification of the code. [15, 16] both adopted the model checking method to validate the software trustworthiness. However, the current study is centred around the UML diagram of the early stage of the software development phase, for this reason, it does not go deep into the interaction level and also cannot verify the complex software trustworthiness in the operation phase. In this paper, we utilize the model-checking algorithm of IMC to extend the measurement of software trustworthiness.

3 Model checking

In this section, we will introduce the model-checking algorithm in detail. Next, the experiment, analysis and measurement will be given.

The structure of the model is roughly divided into three parts: modelling phase, running phase, analysis phase. The overall structure framework is shown in Figure 1 below.

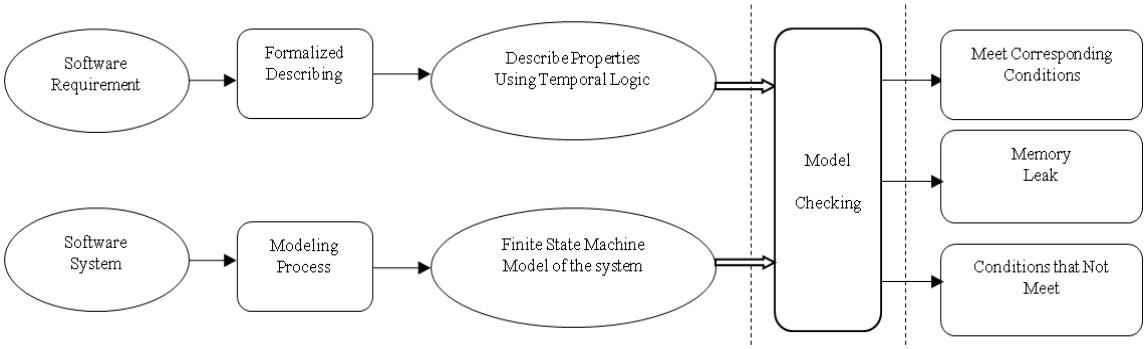

FIGURE 1 Overall structure framework

3.1 MODEL CHECKING ALGORITHM

The key algorithm is the two numerical iterative algorithm $F(s,t)$ and $G(s,t)$.

Theorem one: for $\varphi = \Phi_1 {}_A U^{<t} \Phi_2$.

1. If $s \models \Phi_2$, $\mathrm{Prob}(s, \Phi_1 {}_A U^{<t} \Phi_2) = 1$.

2. If $(s \models \Phi_1 \wedge \neg\Phi_2) \wedge (s \in PS)$,

$$\mathrm{Prob}(s, \Phi_1 {}_A U^{<t} \Phi_2) = \sum_{s' \in S} \int_0^t R(s,s') e^{-E(s)x} \mathrm{Prob}(s', \Phi_1 {}_A U^{<t-x} \Phi_2) dx .$$

3. If $(s \models \Phi_1 \wedge \neg\Phi_2) \wedge (s \in NS)$,

$$\mathrm{Prob}(s, \Phi_1 {}_A U^{<t} \Phi_2) = \sum_{R(s,s')\geq 0} \int_0^{\delta_A(s)} R(s,s') e^{-E(s)x} \mathrm{Prob}(s', \Phi_1 {}_A U^{<t-x} \Phi_2) dx + \sum_{I(s,s') \in A} \mathrm{Prob}(s', \Phi_1 {}_A U^{<t-\delta_A(s)} \Phi_2).$$

4. $\mathrm{Prob}(s, \Phi_1 {}_A U^{<t} \Phi_2) = 1$.

Theorem two: for $\varphi = \Phi_1 {}_A U^{<t} {}_B \Phi_2$.

5. If $(s \models \Phi_1) \wedge (\exists s' ((s' \models \Phi_2) \wedge (\delta_B(s) \leq \delta_{A \setminus B}(s) < t)))$,

$\mathrm{Prob}(s, \Phi_1 {}_A U^{<t} {}_B \Phi_2) = 1$.

6. If $(s \models \Phi_1) \wedge (s \in PS)$,

$$\mathrm{Prob}(s, \Phi_1 {}_A U^{<t} {}_B \Phi_2) = \sum_{s \in S} \int_0^t R(s,s') e^{-E(s)x} \mathrm{Prob}(s', \Phi_1 {}_A U^{<t-x} {}_B \Phi_2) dx .$$

7. If $(s \models \Phi_1) \wedge (s \in NS) \wedge (\delta_{A \setminus B}(s) < \delta_B(s) < t)$

$$\mathrm{Prob}(s, \Phi_1 {}_A U^{<t} {}_B \Phi_2) = \sum_{R(s,s')\geq 0} \int_0^{\delta_{A \setminus B}(s)} R(s,s') e^{-E(s)x} \mathrm{Prob}(s', \Phi_1 {}_A U^{<t-x} {}_B \Phi_2) dx + \sum_{I(s,s') \in A \setminus B} \mathrm{Prob}(s', \Phi_1 {}_A U^{<t-\delta_{A \setminus B}(s)} {}_B \Phi_2).$$

8. $\mathrm{Prob}(s, \Phi_1 {}_A U^{<t} {}_B \Phi_2) = 1$.

3.2 THE TWO EXTENDED MEASUREMENT METHODS

The model-checking algorithm can only assess the performance of the system. However, most users hope to know whether the software system is trusted or not and the measurement value. In addition, state transition as well as the parameters in the model is also important. In this part, we will give the two extended methods to solve the above problems.

3.2.1 The first extension measure

The first extended measure is to establish multiple corresponding temporal logic relations for each software trustworthy attribute that affecting software trustworthiness, that is to say that using multiple temporal logic formulas to describe a software trustworthy attribute, which is aim to measure the software trustworthiness on the multi-level and fine-grained. Then the paper will determine the measurement ultimately.

Here, we have to explain the trustworthy attribute, which generally refers to the functionality, the maintainability, the reliability, the survivability and the controllability of the software. Utilizing these attributes to describe the software trustworthiness. However, each attribute is expressed by multiple temporal logic formulas. The concrete practices are as follows.

The main algorithm of model checking just checks whether each state meet the path formula that is given. If meet the formula, it will return yes, whereas return no. In our first extension measure, we still adopt it. In addition, each temporal logic formula that corresponding to each trustworthy attribute will be taken into account. Then establish the corresponding relations between the important states and the results of these states whether meet each temporal logic formula, as shown below:

$$\{s_0, s_1, s_2, \ldots, s_n\} \underset{corresponds}{\rightarrow} \begin{bmatrix} \Phi_{00} & \Phi_{01} & \ldots & \Phi_{0m} \\ \Phi_{10} & \Phi_{11} & \ldots & \Phi_{1m} \\ \ldots & \ldots & \ldots & \ldots \\ \Phi_{n0} & \Phi_{n1} & \ldots & \Phi_{nm} \end{bmatrix},$$

where $\Phi_{ij}(i \in n; j \in m)$ represents that the state s_i whether meet the j^{th} temporal logic formula or not. If met, then: $\Phi_{ij}(i \in n; j \in m) = 1$ or $\Phi_{ij}(i \in n; j \in m) = 0$. It is easy to find that $\forall i, \forall j, \Phi_{ij}(i \in n; j \in m) = 1$ is the best condition.

Now assume that: $S = [s_1, s_2, ..., s_n]$, $s_i (i \in n)$ indicates the weight of state s_i. $L = [l_1, l_2, ..., l_m]$, $l_j (j \in n)$ indicates the weight of the j^{th} temporal logic formula.

Then the last measurement of the whole software can be simply represented by $M_k = S_{1n} \cdot \Phi_{nm} \cdot L_{1m}^{\ T}$, M_k indicates the k^{th} trustworthy attribute that affecting the software trustworthiness. After all are calculated, we can continue to choose the weighted average method to calculate the system reliability value. During this process, the most important step is to determine the corresponding temporal logic formulas for each trustworthy attribute, because only then can we really reflect the software system in detail. The experiment will be given in Section 3.3.

3.2.2 The second extension measure

The second extension measure is to locate the untrusted states, then find out the detail path and detail parameters of the path. Next, we will get the location that not trusted through further analysis. Eventually meet people's expectations by improving.

In the model of IMC, state transition is used to describe the path parameters. And the crucial factor of state transition is the occurrence time of acts and the state of residence time. Suppose we get the times, then we can clearly depict the system. Hence, the paper obtains the runtime parameters as follows:

$$\sigma = s_i < con_k, \delta(s_i), \delta_{act}(s_i) > s_j ... < ... > s_n,$$

where con_k denotes the jump from state s_i to state s_j belongs to the k^{th} condition, $\delta(s_i)$ denotes the residence time of the state s_i, $\delta_{act}(s_i)$ denotes the occurrence time of the *act* from the state s_i.

The experiment will also be shown at the next section.

3.3 EXPERIMENT AND MEASUREMENT

In this part, we will give the related experimental data and analysis for the two extension measure.

3.3.1 The experiment of the first extension measure

The experiment example is the example of 6.5.1 in [17]. Figure 2 is the IMC model diagram, as follows:

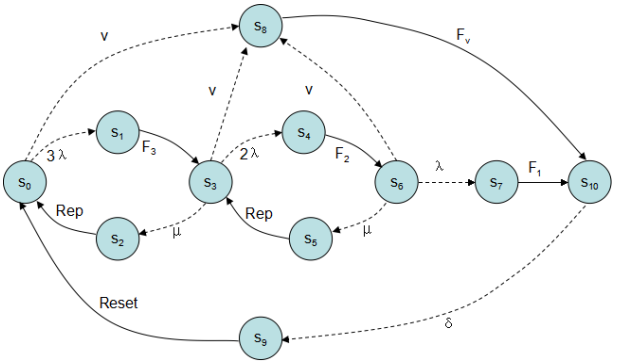

FIGURE 2 IMC model of a fault-tolerant system

Act is the set of acts: $Act = \{F_1, F_2, F_3, F_v, \text{Rep}, \text{Reset}\}$, F_i shows the i^{th} processor is not work, F_v shows the vector cannot work normally Rep represents the fix act, the act of Reset can reset the system.

$\lambda = 0.01$, $\mu = 0.02$, $\nu = 0.001$, $\delta = 0.2$, $\delta_{F_2} = 0.2$, $\delta_{F_3} = 0.1$, $\delta_{F_v} = 0.4$, $\delta_{\text{Rep}} = 0.1$, $\delta_{\text{Reset}} = 0.1$. As shown in Figure 2, there is no doubt that the most important question is the fault tolerance in a fault-tolerant system of IMC model. However, the fault tolerance belongs to reliability. Then we select the reliability to describe the trustworthiness of the system. Temporarily we ignore other properties in this example. Next, we can use several temporal logic equation to describe the fault-tolerant system for fine-grained, as follows:

$$\Phi_1 = P_{<0.02}(true_{\{F_1, F_v\}} U^{<12}{}_{\{F_1, F_v\}} true),$$

$$\Phi_2 = P_{>0.2}(ture_{\varphi} U^{<12}{}_{\{F_1, F_2, F_3, \text{Re}\, p\}} true),$$

$$\Phi_3 = P_{>0.5}(true_{Act} U^{<12} \Phi_2).$$

The fault tolerance is depicted using the three temporal logic formulas. The first step is to build a relationship according the result as follows:

$$\{s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8, s_9, s_{10}\} \underset{corrensponds}{\rightarrow}$$

$$\begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}.$$

The second step is to determine the weight set of each state and the temporal logic formula respectively.

$$S = \{s_0, s_1, ..., s_{10}\} =$$
$$\{0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.06, 0.1, 0.12, 0.02\},$$

$$L = [l_1, l_2, l_3] = [0.3, 0.3, 0.4].$$

The reliability metrics is finally determined using the following formula:

$$M_k = S_{1n} \cdot \Phi_{nm} \cdot L_{1m}^{\ T} (n = 11,\ m = 3) = 0.6380.$$

For this particular system, the measurement of the fault tolerant is equal to the measurement of the reliability. So the trustworthiness measurement value of the fault-tolerant system is 0.6380. However, in general, the software trustworthiness is depicted by many trustworthy attributes. At this point, we should apply different method to synthesize according to the different system and situation.

We can conclude that temporal logic formulas data in table 1 and the corresponding results in table 2 according to the first extended method and algorithm procedures.

TABLE 1 Temporal logic formulas data

Prob(s_i,φ)	Φ_1	Φ_2	Φ_3
S_0	0.017	0.321	1.000
S_1	0.022	1.000	1.000
S_2	0.017	1.000	1.000
S_3	0.021	0.213	1.000
S_4	0.110	1.000	1.000
S_5	0.021	1.000	1.000
S_6	0.121	0.187	0.141
S_7	1.000	0.000	0.475
S_8	1.000	0.000	0.462
S_9	0.018	0.000	1.000
S_{10}	0.011	0.000	0.501

TABLE 2 Corresponding results

Φ_n	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}
Φ_1	1	0	1	0	1	0	1	0	0	1	1
Φ_2	1	1	1	1	1	1	0	0	0	0	0
Φ_3	1	1	1	1	1	1	0	0	0	1	1

3.3.2. The experiment of the second extension measure

The way of the second extension measure is to obtain the detail path and path parameters for the states, which cannot meet the temporal logic formulas in the first extension measure. Then locate the positions that make the reliability low and give the reasons by analysing the path and the path parameters.

Here we still choose the example above, track the path and extract the operation path parameters to the fault-tolerant. We find the state s_6 does not meet the temporal logic formula 2 according the Table 2. Now we will track the path and extract the operation path parameters to the state s_6 as follows:

S6<0.3023,Maxdouble>-S5<Maxdouble,0.1>-
S3<0.3810,Maxdouble>-S4<Maxdouble,0.2>-
S6<0.2832,Maxdouble>-S5<Maxdouble,0.1>-
S3<0.3551,Maxdouble>-S2<Maxdouble,0.1>-
S0<0.2654,Maxdouble>-S1<Maxdouble,0.1>-
S3<0.3331,Maxdouble>-S4<Maxdouble,0.2>-
S6<0.2446,Maxdouble>-S5<Maxdouble,0.1>-
S3<0.3086,Maxdouble>-S4<Maxdouble,0.2>-
S6<0.2314,Maxdouble>-S5<Maxdouble,0.1>-
S3<0.2785,Maxdouble>-S2<Maxdouble,0.1>-
S0<0.2119,Maxdouble>-S1<Maxdouble,0.1>-
S3<0.2608,Maxdouble>-S4<Maxdouble,0.2>-
S6<0.1928,Maxdouble>-S5<Maxdouble,0.1>-
S3<0.2376,Maxdouble>-S2<Maxdouble,0.1>-
S0<0.1769,Maxdouble>-S1<Maxdouble,0.1>-
S3<0.2181,Maxdouble>-S4<Maxdouble,0.2>-
S6<0.1589,Maxdouble>-S5<Maxdouble,0.1>-
S3<0.1961,Maxdouble>-S4<Maxdouble,0.2>-
S6<0.1417,Maxdouble>-S5<Maxdouble,0.1>-
S3<0.1379,Maxdouble>-S4<Maxdouble,0.2>-
S6<0.1288,Maxdouble>-S5<Maxdouble,0.1>-
S3<0.1226,Maxdouble>-S4<Maxdouble,0.2>-
S6<0.1477,Maxdouble>-S5<Maxdouble,0.1>-
S3<0.1070,Maxdouble>-S2<Maxdouble,0.1>-
S0<0.0984,Maxdouble>-S1<Maxdouble,0.1>-
S3<0.1246,Maxdouble>-S2<Maxdouble,0.1>-
S0<0.0869,Maxdouble>-S1<Maxdouble,0.1>-
S3<0.0840,Maxdouble>-S2<Maxdouble,0.1>-
S0<0.0766,Maxdouble>-S1<Maxdouble,0.1>-
S3<0.0767,Maxdouble>-S4<Maxdouble,0.2>-
S6<0.0687,Maxdouble>-S5<Maxdouble,0.1>-
S3<0.0835,Maxdouble>-S4<Maxdouble,0.2>-
S6<0.0585,Maxdouble>-S5<Maxdouble,0.1>-
S3<0.0706,Maxdouble>-S4<Maxdouble,0.2>-
S6<0.0459,Maxdouble>-S5<Maxdouble,0.1>-
S3<0.0546,Maxdouble>-S4<Maxdouble,0.2>-
S6<0.0340,Maxdouble>-S5<Maxdouble,0.1>-
S3<0.0394,Maxdouble>-S2<Maxdouble,0.1>-
S0<0.0250,Maxdouble>-S1<Maxdouble,0.1>-
S3<0.0290,Maxdouble>-S4<Maxdouble,0.2>-
S6<0.0150,Maxdouble>-S8<Maxdouble,0.4>-
S10<0.0014,Maxdouble>-S9<Maxdouble,0.1>

Among them, the *Maxdouble* represents the maximum double time. Here, if there is no Markov transfer except action transfer, we assume that the residence time of the state is *Maxdouble*. Similarly, if there is no action transfer except Markov transfer, we suppose the occurrence time of the action is *Maxdouble*.

From the above path, we can find that when the system start run from *s6* to *s9*, then the residual execution time is 0.0625 unit of time. However, the act starting from *s9* is only a Reset action operation, and the execution time of the Reset action is 0.1 unit of time. Hence, the vector fails due to the remaining time 0.0625 is less than 0.1. As a result, the state s_6 cannot meet the temporal logic formulas. Next, it affects the software trustworthiness and makes the measurement low.

4 Conclusions

This paper mainly proposed two expended measurement methods based on IMC model. The first expended method can give a final credibility value according to the result of temporal logic formulas. More than that, the result is intuitive and easy to understand to users. The second expended method can track the path and extract the operation path parameters for the important and untrusted states in accordance with the specific results of the first expended method. Of course, the intention is to analysis the cause of the result.

Software interaction is one of the most important key factors to the software reliability research. In the current open network environment, the introduction of interaction often leads to unpredictable risks, while this article on the basis of software interaction has proposed two extended methods, but there are a lot of limitations in this kind of methods based on IMC model. Moreover, the factors we considering are still not enough. So next, we want to introduce more data information on the basis of the dynamic interaction model, for example, the data or information that has nothing to do with the running time to

dynamically reflect the trustworthiness more accurately, comprehensively and truly.

The purpose to study the trusted software is to build the trusted software system that can meet the users, which requires the creditability validation before putting it into use. However, the current measurement theory, models and methods are mostly stay in theory, there are also some scholars that tried to apply various measurement methods in different kinds of industrial production, service system ,such as in [19-22], and obtained a series of research achievements, this article only carried on the analysis and verification of the experiment on a small fault tolerance system. Then the focus of next step is to try to apply the extended methods to more areas of different systems. This will make the theoretical model can be used in real life successfully and embody the research significance.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (71061001): Research on the key technology for semantic business process model validation.

References

[1] Cerf 2010 V G Trust and the Internet *IEEE Internet Computing* **14**(5) 95-6
[2] Yuyu Yuan and Qiang Han 2011 A Software Behavior Trustworthiness Measurement Method based on Data Mining, International Journal of Computational Intelligence System **4**(5) 817-25
[3] Tang Y, Liu Z 2010 Progress in Software trustworthiness metrics models *Computer Engineering and Applications* **46**(27) 12-6
[4] Ohba M 1984 Software reliability analysis models *IBM Journal of Research and Development* **28**(4) 428-43
[5] Littlewood B A 1975 Reliability model for systems with Markov structure *Applied Statistics* **24** 172-7
[6] Huang C Y, Lyu M R, Kuo S Y 2003 *IEEE Transactions on Software Engineering* **29**(3) 261-9
[7] Littlewood B, Verrall J L 1973 A Bayesian reliability growth model for computer software *Applied Statistics* **22**(3) 332-46
[8] Tang Y, Du Y, Liu W 2009 Design of Trusted Software Based on MDA and Executable Formalization *Computer Engineering* **35**(19) 138-40
[9] Wen J, Wang H, Ying S, Ni Y, Wang T 2010 Toward a Software Architectural Design Approach for Trusted Software Based on Monitoring Chinese Journal of Computers **33**(12) 2321-34 *(in Chinese)*
[10] Tian J, Li Z, Liu Y 2011 A Design Approach of Trustworthy Software and Its Trustworthiness Evaluation *Journal of Computer Research and Devolopment* **48**(8) 1447-54 *(in Chinese)*
[11] Mukherjee A, Siewiorek D P 1997 *IEEE Transactions on Software Engineering* **23**(6) 366-78
[12] Du J, Yang Y, Wang Q, Li M 2011 Evidence-Based Trustworthy Software Process Assessment Method *Journal of Frontiers of Computer Science and Technology* **6** 501-12 *(in Chinese)*
[13] Zhuang L, Cai M, Li C 2010 Software Behavior-Based Trusted Dynamic Measurement *Wuhan University (Nat Sci Ed)* **56**(2) 133-7 *(in Chinese)*
[14] Cheng L, Zhang Y 2009 A Verification Method of Security Model Based on UML and Model Checking *Chinese Journal of Computers* **32**(4) 1035-1039 *(in Chinese)*
[15] He F, Zhang H, Yan F, Yang Y, Wang H, Meng X 2010 Test of Trusted Software Stack Based on Model Checking *Wuhan University (Nat Sci Ed)* **56**(2) 129-32 *(in Chinese)*
[16] Wu J, Wu Y, Tan G 2007 Interactive Markov Chain: The Design, Verification and Evaluation of Concurrent System *Science Press: Beijing (in Chinese)*
[17] Zhuang L, Cai M, Shen C 2011 Trusted Dynamic Measurement Based on Interactive Markov Chains *Journal of Computer Research and Development* **48**(8) 1464-72 *(in Chinese)*
[18] Mohammed M H,·Lim C P,·Qureishat A 2014 A novel trust measurement method based on certified belief in strength for a multi-agent classifier system *Neural Computing and Applications* **24**(2) 421-9
[19] Huynh T D, Jennings N R, Shadbolt N R 2006 Developing an Integrated Trust and Reputation Model for Open Multi-Agent Systems *Autonomous Agents and Multi-Agent Systems* **13**(2) 119-54
[20] Saint Germain B, Valckenaers P, Van Belle J, Verstraete P, Van Brussel H 2012 Incorporating trust in networked production systems *Journal of Intelligent Manufacturing* **23**(6) 2635-46
[21] Zhan G, Shi W, Deng J 2009 Sensor Trust: A Resilient Trust Model for Wireless Sensing Systems *ACM Sensys* Ann Arbor USA 1-40

Authors

Feng He, born in November, 1964, Ningxia, China

Current position, grades: full professor of Computer Science at Computer Department, Jiaxing College, China.
University studies: M.Sc. in Mathematics (1977), PhD in Computer Sciences (2008) at Donghua University, China.
Scientific interest: Database and knowledge engineering, service-oriented computing
Experience: Lead and participated in National and Provincial Scientific research projects.

Haican Peng, born in April, 1987, Henan, China

Current position, grades: master's degree student.
University studies: Computer Science.
Scientific interest: information system analysis and integration, data mining.

Kun Yao, born in February, 1989, Shanxi, China

Current position, grades: Hanzhong Branch Company of Shanxi Province of China Telecom.
University studies: master's degree in Computer Technology (2013).
Scientific interest: computer networks and information security.