International Journal of Computational Intelligence Systems, Vol. 4, No. 5 (September, 2011), 817-825

A Software Behavior Trustworthiness Measurement Method based on Data Mining

Yuyu Yuan[1], Qiang Han[1,2]
1. School of Computer Science, Beijing University of Posts and Telecommunications
XiTuCheng No.10, HaiDian Strict, Beijing, 100876, China
{yyy1012, nxhanq}@gmail.com
www.bupt.edu.cn
2. School of Computer Science and Engineering, Beifang University of Nationalities
WenChang, XiXia Strict, Yinchuan, Ningxia, 750021, China
www.nun.edu.cn

Abstract

This paper presents a congruence measurement method by partitions to apply software trustworthiness measures in dynamic behavior feature datasets. The datasets are generated at software running time. And the method compares the datasets with the static attribute feature datasets generated at software testing time. So this method can make recommendations for users in services selection time under the environment of SaaS. The measurement method is carried out in three stages: firstly, defining the concept of trust, software trustworthiness, static and dynamic feature datasets with fundamental calculating criteria; secondly, providing a group of formulas to illustrate congruence measurement approach for comparing the two types of feature datasets; lastly, giving an architecture supported by software trustworthiness measurement algorithm to evaluate conceptualized hierarchical software trustworthiness.

Keywords: Software Behavior; Software Trustworthiness; Trust; Measurement Method; Data Mining.

1. Introduction

Trust is essential to most human transactions[1] as well as for Internet based software applications. Numerous research papers have addressed trust and software trustworthiness in recent years, but mainly from a security point of view. A decision to trust is usually associated with an explicit or implicit assessment of risk[1]. Therefore we consider trust as a subjective concept sourced from the human mind, and related to this, software trustworthiness as an objective concept, a comprehensive characteristic in Cloud Computing[2,3]. Only if software trustworthiness is consistent and match the expectation of users' trust, users will accept the services provided by software.

Testing[4] and data mining techniques[5] can be used to analyze different types of software engineering data to substantially assist in building software trustworthiness[6, 7, 8]. In this paper, we propose a measurement method for software trustworthiness based on black box testing and data mining techniques to support trustworthiness measurement for Internet-based software[9, 10].

The organization of the remainder of this paper is as follows. Section 2 discusses related work. Section 3 describes measurement concepts. Section 4 presents a group of formulas for congruence measurement of distance between behavior feature datasets and attribute feature datasets. Section 5 proposes a framework for the trustworthiness concept hierarchies, measurement architecture and measurement algorithm, including the components it is comprised of. Finally, section 6 provides a conclusion and future research directions.

2. Related Work

There has been a lot of research on trust and software trustworthiness related to evaluation and measurement. Marsh formalized trust as a computational concept[11].

Published by Atlantis Press

Limam *et al*. described a framework for reputation-aware software service selection and rating as a devise for service recommendation to provide SaaS consumers with different choices[12]. Chen *et al*. discussed the confidence software developing trends and its key technical points based on formal methods[13]. Fang *et al*. proposed a software assurance model S^3R(security, safety, reliability, survivability) to describe the discipline of software assurance[14]. Liu *et al*. introduced the background, significance, current status, scientific objectives, associated scientific problems and the expected results of the major research plan of trustworthy software set up by the National Natural Science Foundation of China[15]. As the software behavior has become more and more complex in the open environment of Internet, measurement of software behavior to evaluate software quality is the critical difference from traditional measurement method in closed environment. Qu put forward the new subject of 'Software Behavior' to describe software behavior[16] systematically and detailed. Mei *et al*. discussed the concept of software analysis, following main software analysis technologies from the view of static analysis and dynamic analysis[17]. Shen *et al*. summarized novel software theories and technologies for trusted computing[18]. In the area of software trustworthiness measurement standards, Yuan *et al*. proposed a set of recommended standards introducing the model for trustworthiness of services platform as well as a certification and monitoring scheme for trustworthy service applications[19]. Trustworthy Service measurement cannot lose support of service computing technologies. For this aspect, Zhang *et al*. illustrated foundations, and realization of services computing[20,21].

Data mining, the automated or semi-automated extraction of useful knowledge from large poorly structured data sources, is a hot topic in many fields[22,23]. Gay *et al*. showed in their paper that treatment learners can outperform traditional numerical optimization routines in isolating small sets of critical system parameters[24]. As we have argued above, trust is a subjective concept derived from human social networks. Huang *et al*. presented an approach to a formal-semantic-based calculus of trust and explored how to develop decentralized public-key certification and verification[25,26] scheme. For trusted cloud computing, Hwang suggested using layered trust-overlay networks over cloud-based data centers to implement reputation systems[27]. For causes of distrust in software, Zhan *et al*. presented a software distrust chain[28]. Addressing the problem of anticipating software execution effect and behavior, Fang *et al*. proposed research methods for behavior-aware networked software trustworthiness[29]. Trustworthiness measurement has close relations to services composition. Zeng *et al*. proposed a dynamic evolution mechanism for trustworthy software based on service composition[30]. Shao et al. discussed design, assets evaluation and evidence collection mechanism for software trustworthiness[31,32,33]. Bao et al. researched the trustworthiness evaluation method for domain software based on actual evidence generated from software lifecycle[34]. Besides the above mentioned research, there are several papers about trustworthiness and dynamic evolutionary complexity[35,36,37,38,39,40] as well as reputation propagation[41,42,43].

The difference between the above mentioned work and our work is that they focused on trustworthiness measurement or trust management, whereas our work provides a detailed measurement concept with formal trustworthiness calculation formulas and algorithms to improve consistency and matching subjective trust expectations with objective trustworthiness.

3. Measurement Concept

In this section, we introduce the formal definitions of trust and trustworthiness, static and dynamic trustworthiness data and measurement criteria.

3.1. *Trust and Trustworthiness*

Definition 1. Trust is a three-tuple $(E_1, E_2, t_{E_1}^{E_2})$, where : E_1 is trustor, E_2 is trustee, $t_{E_1}^{E_2}$ is the value of trust made by E_1 upon E_2, where:
$E1 \bigcap E2 = \varnothing, E1 \bigcup E2 \neq \varnothing; t_{E_1}^{E_2} \in [0,1]$

Definition 2. Software Trustworthiness T is a combination attribute consisting of sub-attributes according to the requirement. $T \in [0,1]$; the greater the value of T, the higher the trust in the software is.

Definition 3. Software Initialization Trustworthiness $T_{sit}(s)$ is set at software startup, $T_{sit}(s) \in [0,1]$; greater values of $T_{sit}(s)$ means higher trust the initialized software is required.

Definition 4. Software Trusted Threshold $T_{stt}(s)$ **,** which is set by the user before software running, $T_{stt}(s) \in [0,1]$, greater values of $T_{sit}(s)$ means higher trust in the terminated software is required.

Definition 5. Software Runtime Trustworthiness $T_{srt}(s)$, which is measured at software run time by a software measurement tool or agent according to its actual behavior and evaluated by the user.

There is no doubt that trustworthy software running condition should be $T_{sit}(s) \geq T_{srt}(s) \geq T_{stt}(s)$. Otherwise, the software should be terminated.

3.2. *Static and Dynamic Trustworthiness Data*

From the view of software engineering, all of the software's initial attributes can be reflected in Software Test Data (*STD*). It is well known that any partition of *STD* can be uniquely associated with an equivalence relation on *STD* . So we define *STD* as the static trustworthiness data to reflect Software Initialization Trustworthiness $T_{sit}(s)$ through equivalence partition of Black Box Test before delivering the software.

On the contrary, all dynamic attributes of the software can only be reflected by Software Executed Data (*SED*). So we define *SED* as dynamic trustworthiness data to reflect Software Runtime Trustworthiness $T_{srt}(s)$ through an equivalence partition approach to Black Box Test after delivering software and comparing with equivalence partition on *STD* .

Definition 6. Assume X is a finite collection of *STD* or *SED* **,** we recall that an equivalence relation R on X is a mapping $R: X \times X \to \{0,1\}$. Therefore, we now denote *Rs* as Static Data when *Rs* a real case of is R defined above and collected from a software test environment before it was delivered to use.

Definition 7. Taking the equivalence relation R as Rule-Type information, according to Artificial Intelligence Theory, we can apply introduce the theory for software trustworthiness measurement and evaluation. R represent in:

$if \quad R \quad then \quad H \quad ; \quad [0 \leq (CF(R), CF(H,R)) \leq 1]$

Where: H means trustworthiness of owning trustee. The rule can be explained:

Given R occurred with probability $CF(R)$, trustee is the software itself, the trustworthiness of rule (R,H) with probability $CF(H,R)$. So the trustworthiness of software is H with probability $CF(H)$.

We can calculate $CF(H)$ through **Criteria 1-5**:

Criteria 1. According to the definitions above, $CF(H)$ can be calculated as follows:

$$T_{sit}(s) = CF(H) = CF(H,R) \times CF(R) \tag{1}$$

Criteria 2. Given $R = (R_1 \wedge R_2 \wedge ... \wedge R_n)$, then

$$CF(R) = Min(CF(R_1), CF(R_2), ..., CF(R_n)) \tag{2}$$

Criteria 3. Given $R = (R_1 \vee R_2 \vee ... \vee R_n)$, then

$$CF(R) = Max(CF(R_1), CF(R_2), ..., CF(R_n)) \tag{3}$$

Criteria 4. Given that $R_1, R_2, ..., R_i, ..., R_n$ have same H , then Software Initialization Trustworthiness $T_{sit}(s)$ can be represented by the combination trustworthiness $CF(H)$ generated from *STD* as:

$$CF(H) = \begin{cases} \left(\sum_{i=1}^{n} CF_i(H)\right) \times \left(\prod_{i=1}^{n} CF_i(H)\right) \times \frac{1}{n}, & \left[CF_i(H) \geq 0.5, i \in [1,n]\right] \\ \left(\sum_{i=1}^{n} CF_i(H)\right) \times \left(\prod_{i=1}^{n} CF_i(H)\right) \times n, & \left[CF_i(H) \leq 0.5, i \in [1,n]\right] \\ \left(\sum_{i=1}^{n} CF_i(H)\right) \times \frac{1}{n}, & [otherwise] \end{cases} \tag{4}$$

4. Congruence Measurement by Partitions

In section 3 we have introduced the definition of *STD* and *SED* associated with the $T_{sit}(s)$ calculated on *STD* .

In this section, we consider formulating congruence measurement from the perspective of partitions on *STD* and *SED* in order to calculate $T_{srt}(s)$.

Assume that we have two partitions of the test space X. According to the definition above, X is comprised of:

$$X = \bigcup_{i=1}^{Card(\{(R,H)\})} R_i \quad (5)$$

$$P_{STD} = R_{T1},...,R_{Tp}, R_{Ti} \cap R_{Tj} = \varnothing, i \neq j, \bigcup_{i=1}^{Tp} R_{Ti} = STD \subseteq X \quad (6)$$

$$P_{SED} = R_{E1},...,R_{Eq}, R_{Ei} \cap R_{Ej} = \varnothing, i \neq j, \bigcup_{i=1}^{Eq} R_{Ei} = SED \subseteq X \quad (7)$$

It is critical is to obtain a mapping Cong: $P_{STD} \times P_{SED} \to [0,1]$ indicating the degree of congruence or similarity between P_{STD} and P_{SED}.

4.1. *General Measure congruence*

Here we calculate the congruence between P_{STD} and P_{SED} using the underlying equivalence relations. We note that if for $x \neq y$ we indicate by $\langle x, y \rangle$ an unordered pair, $\langle x, y \rangle = \langle y, x \rangle$, then if X has $n = Card(\{(R,H)\})$ elements, then we have $\binom{n}{2} = \frac{(n)(n-1)}{2} = n^c 2$ unordered pairs. We now suggest a general measure of congruence between partitions of P_{STD} and P_{SED} which we express in terms of their underlying equivalence relations.

$$\overline{Cong}(P_{STD}, P_{SED}) = \frac{\sum_U CF(R_{SED}\langle x, y \rangle) \times CF(R_{SED}, H)}{\sum_U CF(R_{STD}\langle x, y \rangle) \times CF(R_{STD}, H)} \times \left(1 - \frac{Diff\_Val(P_{STD}, P_{SED})}{\binom{n}{2}}\right) \quad (8)$$

Here $D = Diff\_Val(P_{STD}, P_{SED})$ is the number of pairs that have different values in P_{STD} and P_{SED}. Then we can calculate the Software Runtime Trustworthiness $T_{srt}(s)$ from the Software Initialization Trustworthiness $T_{sit}(s)$

$$T_{srt}(s) = \overline{Cong}(P_{SID}, P_{SED}) \times T_{sit}(s) = R_{SED/SID} \times \left(1 - \frac{D}{\binom{n}{2}}\right) \times T_{sit}(s) \quad (9)$$

4.2. *Measure congruence by partitions*

In the subsection above, we have introduced a general measure of similarity or congruence, between two partitions on *STD* and *SED* using the underlying equivalence relations. That formula (9) implies that we should traversal all of the equivalence relations from the *STD* and *SED* circularly. So the largest complexity of the formula (9) is $O\left(\left(Card(R) \mid (R,H) \in X\right)^2\right)$.

We now consider the perspective of the partitions themselves. Taking into account the formulas (6) and (7), without loss of generality we can assume $q = p$. If $q > p$ we can augment the partition P_{SED} by adding $q - p$ subsets $R_{Ep+1} = R_{Ep+2} = ... = R_{Eq} = \varnothing$. Thus in the following we assume the two partitions have the same number of classes, q. We now introduce an operation called a pairing of P_{STD} and P_{SED}, denoted $g(P_{STD}, P_{SED})$, which associates with each subset R_{Ti} of P_{STD} a unique R_{Ei} from P_{SED} grouped according to $(R,H)_i \in X$. We then have that a pairing $g(P_{STD}, P_{SED})$ is a collection of q pairs, $g(P_{Ti}, P_{Ei})$. We now associate with each pairing a score, $g(P_{STD}, P_{SED})$, defined as follows. Denoting $D_{g.j} = P_{Tj} \cap P_{Ej}$, for $j = 1$ to q, we obtain:

$$Score\left(g(P_{STD}, P_{SED})\right) = \sum_{j=1}^{q} \left(Card\left(D_{g.j}\right) \times \frac{CF(H_{Ei})}{CF(H_{Ti})} \right) \quad (10)$$

We will now use this to obtain congruence,

$$\overline{\overline{Cong}}(P_{STD}, P_{SED}) = \frac{Score\left(g(P_{STD}, P_{SED})\right)}{Card(X)} \quad (11)$$

Then we can calculate $T_{srt}(s)$ from $T_{sit}(s)$

$$T_{srt}(s) = \overline{\overline{Cong}}(P_{STD}, P_{SED}) \times T_{sit}(s) \quad (12)$$

Through analysis, the complexity of formula (12) is, $O\left(Card(STD) \times Card(SED)\right)$, which is less than or equal to the complexity of $\overline{Cong}(P_{STD}, P_{SED})$ in formula (17) because $Card(STD) \leq Card(X)$ and $Card(SED) \leq Card(X)$ according to formula (6) and (7).

Therefore, can we conclude that the performance of formula (12) is much better than formula (9) just by

their different complexity? Indeed, with the trend of SaaS, more and more software components are coming from third parties, so there no longer exists a steady and closed *STD* . For this reason, the precondition of formula (11) that cluster *STD* into the test space X would visibly increase its complexity.

5. Measurement Framework

An important application of measuring congruence between different partitions on the same test space X proposed in section 4 is the trustworthiness concept hierarchies[44]. In this section, we first conceptualize the trustworthiness into a basic concept hierarchy chart with congruence measurement formulas. Then the measurement architecture and algorithm based on the chart are presented.

5.1. *Trustworthiness Concept Hierarchies*

Assume that X is a finite collection of *STD* or *SED* . A trustworthiness concept hierarchy is a collection of partitions, $P_1,...,P_r$. Here P_k is called the k^{th} level partition. The fundamental property of the concept hierarchy is that each class (granular or cluster) in a lower level partition is fully contained in one class of the next more coarse as we go up.

Formally we have Matrix T as the collection of partitions:

$$\begin{bmatrix} P_{Top}(1,1,\wedge/\vee) \\ \cdot \\ P_r : T_{r1}(r_{r1},w_{r1},c),T_{r2}(r_{r2},w_{r2},c),...,T_{rqr}(r_{rqr},w_{rqr},c) \\ \cdot \\ P_2 : T_{21}(r_{21},w_{21},c),T_{22}(r_{22},w_{22},c),...,T_{2q2}(r_{2q2},w_{2q2},c) \\ P_1 : T_{11}(r_{11},w_{11},\wedge),T_{12}(r_{12},w_{12},\wedge),...,T_{1q1}(r_{1q1},w_{1q1},\wedge) \end{bmatrix} \quad (13)$$

We note that for $m > k$ we have $q_m < q_k$. For any class in the k^{th} level, T_{kj} , there exists a class in the m^{th} level, T_{mi} such that $T_{kj} \subseteq T_{mi}$. Then with $m > k$ we have for any T_{mi} that $T_{mi} = \bigcup_{j \in S_{mi/k}} T_{kj}$ where $S_{mi/k} \subseteq \{1,...,qk\}$. In addition, every element T_{mi} in matrix represents one (R,H) in formula (1) occurring with probability r_{mi} , weight w_{mi} and composition mode c_{mi} of its sublevel classes. Every element T_{mi} has two components: Q and H . $T_{mi}.Q = Card(T_{mi})$, $T_{mi}.H$ means its value contributes to $T_{(m+1)j}$ $\left(T_{mi} \subseteq T_{(m+1)j}\right)$. Then we obtain the formula as:

$$\sum_{j=1}^{Card(S_{mi/k})} w_{kj} = 1, c_{mi} = \wedge \quad (14)$$

$$\sum_{j=1}^{Card(S_{mi/k})} \left(w_{kj} = 1\right) = Card\left(S_{mi/k}\right), c_{mi} = \vee \quad (15)$$

$$0 \le r_{mi} = \prod_{j=1}^{Card(S_{mi/k})} \left(r_{kj} \times w_{kj}\right) \le 1, c_{mi} = \wedge \quad (16)$$

$$0 \le \sum_{j=1}^{Card(S_{mi/k})} \left(r_{kj} \times w_{kj}\right) \le Card\left(S_{mi/k}\right), c_{mi} = \vee \quad (17)$$

Now we consider the three classic distribution charts with highest probability of (13). The first is mapping all partitions of *STD* and *SED* into *x* rows. The second is mapping them into *y* columns. The third is mapping them into *x* rows and *y* columns. Then formulas (18)-(21) can represent the three classic distribution charts.

$$\overline{\overline{Cong}}^{w1} = \left(\sum_{k=1}^{Card(P_{k+1})} \left(w_{k+1} \times \sum_{j}^{(S_{(k+1)j/k})} \left(w_{kj} \times \overline{\overline{Cong}}(P_{STD}^{kj}, P_{SED}^{kj}) \right) \right) \right) \quad (18)$$

$$\overline{\overline{Cong}}^{w2} = \left(\prod_{i=Min(L)}^{Max(L)} \left(w_{k+1} \times \sum_{j}^{(S_{(k+1)j/k})} \left(w_{ij} \times \overline{\overline{Cong}}(P_{STD}^{kj}, P_{SED}^{kj}) \right) \right) \right) \quad (19)$$

$$\overline{\overline{Cong}}^{w3} = R_{SED/STD} \times \left(\sum_{Rows} Cong^{w1}_{1} + \sum_{Columns} Cong^{w2}_{1} \right) \quad (20)$$

$$T_{srt}(s) = \overline{\overline{Cong}}^{w3} \times T_{sit}(s) \quad (21)$$

5.2. *Measurement Architecture*

The current subsection describes the architecture of the measurement system. The component which provides the initial interaction of the customer with the system is the web-based user interface. It has functionalities for uploading three classes of documents: P_{STD}, P_{SED}, and Concept Matrix T. For example, they may be uploaded at company level, and continue to be initialized by the Matrix Processor into the Partitions Processor stage. After that stage, the three classes are stored in three Repositories. Next, the Congruence Engine creates $R_{SED/STD}$ as the trust ratio of SED and STD. Finally, the Measurement Processor calculates $T_{sit}(s)$, $T_{srt}(s)$ to Recommender in order to give users recommendations. The specific components of the architecture are described in Fig.1, which illustrates the architecture. Moreover, trust is a subjective concept, so it may continuously evolve and be analyzed by the Reputation Analyzer for revising recommendations more precisely and correctly. We will specially discuss problems related to Reputation Propagation in subsequent papers.

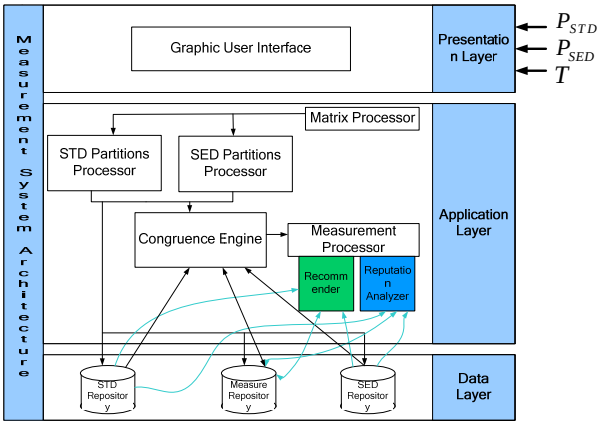

Fig.1 Measurement System Architecture

5.3. *Measurement Algorithm*

According to the formulas (1)-(21) presented above, we now propose a measurement algorithm by integrating the formulas. The aim is to archive a recommendation to the users at services selection time according to trustworthy software running condition in section 3.

Here the algorithm is illustrated as follows:

0: Initialization: $T_{stt}(s)$, P_{STD}, P_{SED}, Concept Matrix T of Hierarchy Trustworthiness with every element' component Q and H initialized to Zero.

Stage1: Calculate $T_{sit}(s)$.

1: $i = j = k = 1$

2: **for each** partition R_{Ti} in P_{STD}:

2: **for each** level partition P_k in T:

3: **for each** class T_{kj} in P_k:

4: **if** $\exists(x,y) \in R_{Ti} \Rightarrow (x,y) \in T_{kj}$

5: $T_{kj}.Q = T_{kj}.Q + Card(R_{Ti})$

6: **endif**

7: **endfor**

8: **endfor**

9: **endfor**

10: $j = k = 1$

11: **for each** level partitions P_k in T:

12: **for each** class T_{kj} in P_k:

13: **if** $(T_{kj}.Q > 0) \& (T_{(k+1)i}.c = \wedge) \& (T_{kj} \subseteq T_{(k+1)i})$

14: $T_{(k+1)i}.H = Min(T_{kj}.H) \times T_{(k+1)i}.r$

15: **else if**

16: $(T_{kj}.Q > 0) \& (T_{(k+1)i}.c = \vee) \& (T_{kj} \subseteq T_{(k+1)i})$

17: $T_{(k+1)i}.H = Max(T_{kj}.H) \times T_{(k+1)i}.r$

18: **endif**

19: **endfor**

20: **endfor**

21: $T_{sit}(s) = T_k.H$

Stage2: Calculate $T_{srt}(s)$

22: $i = j = k = 1$

23: **for each** partition R_{Ei} in P_{SED} :

24: **for each** level partitions P_k in T :

25: **for each** class T_{kj} in P_k :

26: **if** $\exists(x, y) \in R_{Ei} \Rightarrow (x, y) \in T_{kj}$

27: **if** $T_{kj}.Q > 0$

28: $0 \to T_{kj}.Q$ **else**

29: $T_{kj}.Q = T_{kj}.Q + Card(R_{Ei})$

30: **endif**

31: **endif**

32: **endfor**

33: **endfor**

34: **endfor**

35: **for each** row and **each** column in T :

36: calculate $\sum \overline{\overline{Cong}}^{w1}$.

37: calculate $\sum \overline{\overline{Cong}}^{w2}$.

38: **endfor**

40: calculate $R_{SED/STD}$ of formula (16).

41: calculate $\overline{\overline{Cong}}^{w3}$ of formula (28).

42: calculate $T_{srt}(s) = \overline{\overline{Cong}}^{w3} \times T_{sit}(s)$

Stage3: Generate Recommendation

43: **if** $T_{sit}(s) \geq T_{srt}(s) \geq T_{stt}(s)$

44: $Recommending = 'True'$.

45: **else** $Recommending = 'False'$.

46: **output**: *Recommending* .

Algorithm End.

6. Conclusion and Future Work

Many initiatives have been proposed to extract structure and apply trustworthiness measurement to Internet-based software. In this paper, a testing and data mining based measurement method applying dynamic behavior datasets and static attributes datasets for software trustworthiness is proposed. The aim is to make recommendations to users of whether or not the verified software is running following consistently what it has declared to do. The method complies with subjective trust expectations of users; hence, the subjective trust judgment is directly linked with the objective software trustworthiness; this is the main new point of this paper.

In future work we include the development of a measurement method for executing a client program in a distributed computing environment to demonstrate its performance. In addition, this paper is mainly applying through Black Box Test to study external behavior feature datasets. In the future, we plan to study software internal behavior track datasets through Write Box Test based on Stochastic Petri Net technology.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (90818006, 61010306008); National Science and Technology Major Project of China (2009ZX01039-001-002). National Basic Research 973 program of China (2007CB311101). Ningxia Natural Science Foundation of China (NZ0955). Furthermore, we shall express our deep thanks for the recommendation of Jorgen Boegh and the anonymous viewers.

References

1. V. G. Cerf, Trust and the Internet, *IEEE Internet Computing.* **14**(5) (2010) 95-96.
2. G. Pallis, Cloud Computing: The New Frontier of Internet Computing, *IEEE Internet Computing.* **14**(5) (2010) 70-73.
3. M. Cusumano, Cloud Computing and SaaS as New Computing Platforms, *Communications of the ACM*. **53**(4) (2010) 27-29.
4. D. R. Wallace and R. U. Fujii, Software Verification and Validation: An Overview, *IEEE Software*, **6**(3) (1989) 10-17.
5. R. R. Yager, Some Measures Relating Partitions Useful For Computational Intelligence, *International Journal of Computational Intelligence Systems*. **1**(1) (2008) 1-18.
6. L. Liu and W. Shi, Trust and Reputation Management, *IEEE Internet Computing.* **14**(5) (2010) 10-13.
7. A. S. Patrick, Building Trustworthy Software Agents, *IEEE Internet Computing.* **6**(6) (2002) 46-53
8. R. S. Pressman, Software Engineering: *A Practitioner Approach* (McGraw-Hill International, 6th edition, 2006).
9. R. Sandhu, The Technology of Trust, *IEEE Internet Computing.* **6**(6) (2002) 28-29.
10. H. Wang, Y. Tang, G. Yin and L. Li, Trustoworthiness of Internet-based software, *Science in China Series F: Information Sciences*, **49**(6) (2006) 759-773.
11. S. P. Marsh, Formalising Trust as a Computational Concept, doctoral thesis, University of Stirling, Scotland, 1994.
12. N. Limam and R. Boutaba, Assessing Software Service Quality and Trustworthiness at Selection Time, *IEEE Transactions on Software Engineering*. **36**(4) (2010) 559-574.
13. H. Chen, J. Wang and W. Dong, High Confidence Software Engineering Technologies, *Acta Electronic Sinca*, **31**(12A) (2003) 1933-1938. (in Chinese with English abstract).
14. B. Fang, T. Lu and C. Li, Survey of software assurance, Journal on Communications, **30**(2) (2009) 106-117. (in Chinese with English abstract).
15. K. Liu, Z. Shan, J. Wang, J. He, Z. Zhang and Y. Qin, Overview on Major Research Plan of Trustworthy Software, *Bulletin of National Science Foundation of China*, **22**(3) (2008) 145-151. (in Chinese with English abstract).
16. Y. Qu, *Software Behavior* (Publish House of Electronics Industry, Beijing, CN, 2004). (in Chinese)
17. H. Mei, Q. Wang, L. Zhang and J. Wang, Software Analysis: A Road Map, *Chinese Journal of Computers.* **32**(9) (2009) 1697-1710. (in Chinese with English abstract).
18. C. Shen, H. zhang, H. Wang, J. Wang, B. Zhao, F. Yan, F. Yu, L. Zhang and M. Xu, Research and Development of Trusted Computing, *Sci. China Ser F-inf Sci*, **40**(2) (2010) 139-166. (in Chinese).
19. Y. Yuan, S. Witold and J. Boegh, *Research on Key Technologies of Software Trustworthiness* (Publish House of Electronics Industry, Beijing, CN, 2010).
20. L. Zhang and J. Zhang, Architecture-Driven Variation Analysis for Designing Cloud Applications, in *Proc. Int. Conf. Cloud Computing*,eds. L. Zhang (Bangalore, India, 2009), 125-134.
21. L. Zhang , J. Zhang and H. Cai, *Services Computing* (Tsinghua University Press and Springer, Beijing, CN, 2007).
22. R. J. Hall, Editorial: data mining in software engineering, Automated Software Engineering, **17**(4) (2010), 373-374.
23. A. Garcia-Crespo, R. Colomo-Palacios, J. M. Gomez-Berbis and M. Mencke, BMR: Benchmarking Metrics Recommender for Personnel issues in Software Development Projects, *International Journal of Computational Intelligence Systems*. **2**(3) (2009) 256-266.
24. G. Gay, T. Menzies, M. Davies and K. Gundy-Burlet, Automatically finding the control variables for complex system behavior, Automated Software Engineering, **17**(4) (2010), 439-468.
25. J. Huang and D. M. Nicol, A Formal-Semantics-Based Calculus of Trust, *IEEE Internet Computing.* **14**(5) (2010) 38-46.
26. J. Huang and D. M. Nicol, A Calculus of Trust and Its Application to PKI and Identity Management, in *Proc. 8th Symp. Identity and Trust on the Internet*, eds. K. Seamons (Gaithersburg, MD, 2009)
27. K. Hwang and D. Li, Trusted Cloud Computing with Secure Resources and Data Coloring, *IEEE Internet Computing.* **14**(5) (2010) 14-22.
28. J. Zhan, X. Zhou and J. Zhao, Analysis of the Original Cause of Software Distrust, in *Proc. 2nd Int. Conf. Software Engineering and Data Mining*, eds. G. Kou, Y. Peng, F. I. S. Ko, Y. Chen, T. Tateyama (Chengdu, China, 2010), pp. 240-245.
29. X. Fang, C. Jiang and X. Fan, Behavior-aware Trustworthiness Study of Networked Software, *International Journal of Computational Intelligence Systems*. **3**(5) (2010) 542-552.
30. J. Zeng, H. Sun, X. Liu, T. Deng and J. Huai, Dynamic Evolution Mechanism for Trustworthy Software Based on Service Composition, *Journal of Software*. **21**(2) (2010) 261-276. (in Chinese with English abstract).
31. Y. Liu, Z. Ma, X. He and W. Shao, Approach to Transforming UML Model to Reliability Analysis Model, *Journal of Software*. **21**(2) (2010) 287-304. (in Chinese with English abstract).
32. S. Cai, Y. Zou, L. Shao, B. Xie and W. Shao, Framework Supporting Software Assets Evaluation on Trustworthiness, *Journal of Software*. **21**(2) (2010) 359-372. (in Chinese with English abstract).

33. L. Gu, Y. Guo, H. Wang, Y. Zou, B. Xie and W. Shao, Runtime Software Trustworthiness Evidence Collection Mechanism Based on TPM, *Journal of Software*. **21**(2) (2010) 373-387. (in Chinese with English abstract).
34. T. Bao, S. Liu and X. Wang, Research on Trustworthiness Evaluation Method for Domain Software Based on Actual Evidence, *Chinese Journal of Electronics*. **20**(2) (2011) pp195-199.
35. J. Pan, F. Xu, J. Lu, Reputation-Based Recommender Discovery Approach for Service Selection, *Journal of Software*. **21**(2) (2010) 388-400. (in Chinese with English abstract).
36. Z. Zheng, S. Ma, W. Li, X. Jiang, Z. Zhang and B. Guo, Dynamical characteristic of software trustworthiness and their evolutionary complexity, *Science in China Series F: Information Sciences*, **52**(8) (2009) 1328-1334.
37. Z. Zheng, S. Ma, W. Li, X. Jiang, W. Wei, L. Ma and S. Tang, Complexity of software trustworthiness and its dynamical statistical analysis methods, *Science in China Series F: Information Sciences*, **52**(9) (2009) 1651-1657.
38. W. Wang and G. Zeng, Trusted dynamical level scheduling based on Bayes trust model, *Science in China Series F: Information Sciences*, **50**(3) (2007) 456-469.
39. B. Lang, A computational trust model for access control in P2P, *Science in China Series F: Information Sciences*, **53**(5) (2010) 896-910.
40. R. Zhu, Research on Key Technologies for Trustworthy Service Composition, doctoral thesis, National University of Defense Technology, China, 2009.
41. H. Hu, Research on Distributed Access Control Based on Trusted Computing, doctoral thesis, University of Science and Technology of China, China, 2009.
42. Y. Zhang, H. Chen, X. Jiang, H. Sheng and Z. Wu, RCCtrust: A Combined Trust Model for Electronic Community, *Journal of Computer Science and Technology*, **24**(5) (2009) 883-892.
43. X. Feng, J. Pan and W. Lu, A Trust-Based Approach to Estimating the Confidence of the Software System in Open Environments, *Journal of Computer Science and Technology*, **24**(2) (2009) 373-385.
44. National University of Defense Technology, Peking University, Beihang University and CVICSE, http://www.trustie.net .