International Journal of Hybrid Information Technology
Vol.8, No.1 (2015), pp.35-50
http://dx.doi.org/10.14257/ijhit.2015.8.1.04

The Software Trustworthiness Evaluation Model Based on Subjective Logic

Jiao Hongqiang[1,2] and Tian Junfeng[3]

[1]*School of Management, Hebei University, Baoding, China, 071002*
[2]*Electronic and Information Engineering Experiment and Training Center, Handan College*
[3]*College of Mathematics & Computer Science, HeBei University, Baoding, China, 071002*
[1]*jhq1983@163.com,* [2]*tjf@hbu.edu.cn*

Abstract

In recent years, the trustworthiness of the software has become the focus of software quality. The software trustworthiness analysis and measurement has become a hot issue. Because of the great influence of software environment dynamic openness and uncertainty to the trustworthiness of the software, this paper attempts to examine the changes of software running environment, consider the impact on the software trustworthiness, and build the software trustworthiness evaluation (STE) model with subjective logic. The model can be real-time control the software running state and evaluate the software trustworthiness dynamically. In addition, on the setting of weight aspect, a subjective group preference weight setting algorithm is designed. Simulation results show that the model is reasonable and effective can be more accurate to evaluate the software trustworthiness.

Keywords: *Software trustworthiness; Subjective logic; Running environment; Dynamic uncertainty*

1. Introduction

In recent years, the development and running environment of software systems have been transformed into dynamic open from closed. The open and changeable running environment cause kinds of uncertain factors which make trustworthiness of the software (correctness, safety, reliability, availability, effectiveness and survivability) difficult to guarantee. In the open and changeable environment, the software system is becoming more and more fragile that let people feel difficult to trust. Most of the time, the software is not work in the ways they expect. There are kinds of fault and failure happened, which cause accidents and disasters directly or indirectly. Therefore, the trustworthiness of the software system is very concerned by the current domestic and international academic and industry circles and a heavy inputs research topic [1]. For objective judgment of the software entities trusted state, it is an urgent need to study the software trustworthiness evaluation techniques. "Multiscale quantization index system of multidimensional trustworthiness attribute, measurement, assessment and evaluation system" has become a research hotspot in trusted software field [2].

The traditional STE model or method usually uses software metrics or logic verification method for quantitative estimation of software. Although these models or methods have solved some practical problems, applying to the uncertainty evolution software system with in large scale and complex function, they still exist: fail to fully consider the key factors that affect the evaluation result of the correctness (software runtime environment) and do not have the dynamic adaptive

ISSN: 1738-9968 IJHIT

characteristics. Faced with such dynamic uncertainty software operating environment, choice suitable theory tools is the key to evaluate its trustworthiness. In fact, the software trustworthiness can be viewed as a special case in trust management. People have a wide range of research on this aspect and theoretical achievements. Among them, subjective logic [14] proposed by Audun Jøsang is one of the classic trust models in trust management area. From the D-S evidence theory, Jøsang put the uncertainty into the model and proposed the subjective logic theory. And then through modeling the trust relationship based on the subjective logic, achieved good results. Subjective logic can describe the subjective cognition uncertainty and it is more suitable for modeling the trustworthiness of the dynamic uncertainty object. Therefore, this paper attempts to examine the changes of software running environment and use the classic trust management theory-subjective logic to build the STE model.

The rest of the paper is organized as follows. Section 2 related work Section 3 introduces software trustworthiness evaluation model based subjective logic Section 4 Describes the experimental setup and evaluates the performance and verify the rationality of the model. Section 5 concludes the paper.

2. Related Work

Research on the software trustworthiness is mainly in two aspects. One is improving the software trustworthiness through software adaptability and software evolution theory. Mohammad M *et al.* [3] present a formal approach for the development of trustworthy component-based systems. The approach involves a formal component model for the specification of component's structure, functional, and non-functional (trustworthiness) properties. Calinescu R *et al.*, [4] use quantitative verification at runtime and model checking as a way to strengthen the software adaptation.

The other one is to set up the evaluation model for software trustworthiness. Needs to be pointed out, this is the focus of the study. Wang Yue *et al.* [5] introduce a meta-model of trustworthiness, founds a knowledge base according to this meta-model, presents a trustworthiness requirements pattern and a method about how to generate patterns from the knowledge base to help eliciting trustworthiness requirements. Si Guannan *et al.* [6] build a multitier dependability evaluation metrics system by structure analysis and structure pattern of Internet ware. It uses bottom up approach on the basis of Bayesian network to calculate dependability metrics of Internet ware system and its components from many aspects. Wang X [7] proposes evidence driven framework for STE based on rules is put forward, rules are used as expression method of trustworthiness evaluation logic, and evidence is used to drive the operation of trustworthiness evaluation process. Most of the works above are the framework of STE, which provide the preparation work for modeling the STE. There are no detail designs for how to evaluate the software trustworthiness, so, they can not be used to evaluate the software trustworthiness in practice.

Ding Shuai *et al.* [8] focuses on the STE problems that include both quantitative and qualitative indicators with uncertainties and propose an objective weight based evidential reasoning approach that employs total uncertainty measure to solve STE problems with specific trustworthiness requirements. But its weight setting, using only objective information entropy method, regardless of people's subjective understanding of software trustworthiness index, and it is not very reasonable. Liu Yuling *et al.* [9] propose a trust model called (CBRA-TM) which is based on checkpoint behavioral risk assessment. Although the model can evaluate the key checkpoints, the real-time is still poor.

Zhou Xianzhong *et al*. [10] proposed a model based on the concept of software trustworthiness. The relationships among software distrust, failure, fault, defect and error are analyzed from converse thinking. Furthermore, the concept of distrust factor is proposed originally. A framework based on WBS-RBS is proposed to collect, identify and classify the distrust factors. The model mentioned above, they are mainly aiming at the non-functional attribute index or software behavior for STE, while the influence of the dynamic change of the software running environment to the software is seldom considered. In many application scenarios, only to ensure the functional correctness and validity is not enough. For example, in the network environment, in the process of software system to provide services, it may face various interference, such as denial of service attack, virus and unpredicted load threats come from the environment. Especially in the situation the popularity of current Internet technology, on the one hand, the software system of the environment gradually shows a characteristic of dynamic and open. In order to achieve specific business functions, the software system may need close interact with the presence of various objects (including people, software, hardware, sensors and other objects) in the environment. And the environment object diversity and difference the characteristic of the environment will bring about negative effects of unexpected software behavior. On the other hand, dynamic and open environment also makes environment object with malicious purposes having a negative impact on the normal behaviors through interact with the software system, which leads to reduce trustworthiness the of system software[5].

Gu Liang *et al*. [11] propose a runtime software trustworthiness evidence collection mechanism based on trusted computing technology. Based on the features provided by TPM (trusted platform module), as well as the late launch technology, a trusted evidence collection agent is introduced in an operating system kernel. The agent can securely monitor executing programs and collect their trustworthiness evidence accordingly. The agent also provides some trusted services for programs to collect application specific evidences and guarantees the trustworthiness of these evidences. Bettini C *et al*. [12] propose context modeling and reasoning techniques.

Ding Bo *et al*. [13] propose a component model named ACOE (adaptive component model for open environment) that supports the online fine-grained adjustment to software adaptability. The model studies the software trustworthiness from the two aspects of behavior and the environment. But this work was just context acquisition and could not evaluate the software trustworthiness effectively.

As mention in section 1, subjective logic proposed by Jøsang *et al*. [14] can express subjective uncertainty, and has achieved gratifying results [15-20]. Subjective logic provides a good trust representation and reasoning for the theoretical foundation of trust management. Its Beta trust model, has successfully applied in the open community of the trust management. Therefore, different from previous STE model, this paper investigates the changes of software running environment, calculates the influence of the software environment dynamic changes to the software trustworthiness by subjective logic theory, and gets the STE model based on environment.

3. The Software Trustworthiness Evaluation Model based on Subjective Logic

3.1. Multinomial Subjective Logic

Subjective opinions express subjective beliefs about the truth of propositions with degrees of uncertainty, and can indicate subjective belief ownership whenever required. A distinction can be made between multinomial and binomial opinions. A multinomial

opinion is denoted as ω_X^A where *A* is the belief owner, also called the subject, and *X* is the target frame, also called state space, to which the opinion applies. An alternative notation is ω (*A*:*X*). In case of binomial opinions, the notation is ω_x^A, or alternatively ω (*A*:*x*), where *x* is a single proposition that is assumed to belong to a frame e.g. denoted as *X*, but the frame is usually not included in the notation for binomial opinions. The propositions of a frame are normally assumed to be exhaustive and mutually disjoint, and belief owners are assumed to have a common semantic interpretation of propositions. The belief owner (subject) and the propositions (object) are attributes of an opinion. Indication of subjective belief ownership can be omitted whenever irrelevant.

A general multinomial opinion is a composite function consisting of a belief vector $\vec{b}$, an uncertainty mass *u* and a base rate vector $\vec{a}$. These components are defined next.

Definition 1 Belief Mass Vector

Let X = {x_i|i = 1, . . . , k} be a frame and let $\vec{b}$ be a vector function from X to [0, 1]^k representing belief masses over X satisfying:

$$\vec{b}(\phi) = 0 \text{ and } \sum_{x \in X} \vec{b}(x) < 1 \qquad (3\text{-}1\text{-}1)$$

$\vec{b}$ is called a belief mass vector, or belief vector for short.

An element $\vec{b}$ (x_i) is interpreted as belief mass over x, *i.e.* the amount of positive belief that x is true. The belief vector can be interpreted as a sub-additive probability function because the sum can be less than one. Additivity is achieved by including the uncertainty mass defined below.

Definition 2 Uncertainty Mass

Let X = {x_i|i = 1, . . . , k} be a frame and let $\vec{b}$ be a vector function from X to [0, 1]^k representing belief masses over X satisfying:

$$u + \sum_{x \in X} \vec{b}(x) = 1 \qquad (3\text{-}1\text{-}2)$$

The parameter u is then called an uncertainty mass.

The uncertainty mass can be interpreted as the lack of committed belief mass in the truth of any of the propositions of X. In other words, uncertainty mass reflects that the belief owner does not know which of the propositions of X in particular is true, only that one of them must be true.

Definition 3 Subjective Opinion

Let X = {x_i|i = 1, . . . , k} be a frame, *i.e.* a set of exhaustive and mutually disjoint propositions x_i. Let $\vec{b}$ be a belief vector, let u be the corresponding uncertainty mass, and let $\vec{a}$ be a base rate vector over X, all seen from the viewpoint of a subject entity A. The composite function $\omega_X^A = (\vec{b}, u, \vec{a})$ is then A's subjective opinion over X. This represents the traditional belief notation of opinions.

Definition 4 Probability Expectation Vector

Let X = {x_i|i = 1, . . .k} be a frame and let ω_X be an opinion on X with belief vector $\vec{b}$ and uncertainty mass u. Let $\vec{a}$ be a base rate vector on X. The function $\vec{E}_X$ from X to $[0, 1]^k$ expressed as:

$$\vec{E}_X(x_i) = \vec{b}(x_i) + \vec{a}(x_i)\mathrm{u} \qquad (3\text{-}1\text{-}3)$$

is then called the probability expectation vector over X.

Dirichlet distributions translate observation evidence directly into probability density functions. The representation of the observation evidence, together with the base rate, can be used to denote opinions.

Definition 5 Evidence Notation

Let X be a frame with a Dirichlet distribution Dirichlet($\vec{p}$ | $\vec{r}$, $\vec{a}$). The evidence notation of opinions can then be expressed as the ordered tuple ω = ($\vec{r}$, $\vec{a}$).

Theorem 1 Evidence Notation Equivalence

Let ω_X^{bn} =($\vec{b}$, u, $\vec{a}$) be an opinion expressed in belief notation, and ω_X^{en} =($\vec{r}$, $\vec{a}$) be an opinion expressed in evidence notation, both over the same frame X. The opinions ω_X^{bn} and ω_X^{en} are equivalent when the following equivalent mappings hold:

$$\begin{cases} \vec{b}(x_i) = \dfrac{\vec{r}(x_i)}{W + \sum_{i=1}^{k} \vec{r}(x_i)} \\ u = \dfrac{W}{W + \sum_{i=1}^{k} \vec{r}(x_i)} \end{cases} \Leftrightarrow \begin{cases} \vec{r}(x_i) = \dfrac{W\,\vec{b}(x_i)}{u} \\ 1 = u + \sum_{i=1}^{k} \vec{b}(x_i) \end{cases} \qquad (3\text{-}1\text{-}4)$$

The default non-informative prior weight is W = 2, but larger values are also possible. Formula (4) is the mapping relationship between evidence space and belief space. The main idea about subject logic and more details see the literature [14].

3.2. Select the Software Running Environment Attributes

Software trustworthiness by some properties: correctness, safety, reliability, usability, efficiency, software trustworthiness evaluation is considered to be a kind of multiple attribute decision making problems, and this kind of model evaluation results and technical indicators often comes from expert opinion and past their subjective experience, the latest knowledge and information due to a lack of any assessment results of [8]. Therefore, running state as can the real-time control software, dynamically to evaluate the software trustworthiness, this paper pays more attention to the software running environment property, simple, mentioned below (property index) refers to can be observed and software operating environment quantitatively describing the properties. The software running environment including the process context, thread information, method, field, however, the monitoring and control property is basically a collection of monitoring in the field, and cannot cover all the monitoring requirements. Therefore, according to the requirement analysis and design specification different, choose the corresponding properties of [22].

As noted above, brought huge impact dynamic software environment and uncertainty of the trustworthiness of the software, through the study of the software running environment changes to evaluate the software trustworthiness is a reasonable and feasible method. Many of software running environments attributes which can be observed, each attribute of software trustworthiness degree is different, therefore, this paper attempts to attribute the first definition. Great for those changes to the attributes of the credibility of the software, which is defined as the key attribute; attribute has some changes, but the credibility of the software not less influence or effect, which is defined as the non-key attributes. The key attribute for the assessment must detect the attributes, rather than key attributes, can be chosen by the user. This is the meaning of: A, model requires only the key observation key attributes, reduce the overhead. Two, the distinction between attributes and attribute weights, complete the general settings. For the second point, because of the different types of software, the environmental attributes focused on is different. For example, for some of the commonly used office software, it will get more attention about CPU, memory and other system attributes, while the bandwidth, connection network properties will not effect the software trustworthiness; and some browsers or chat tools of this kind of interaction with the network software is more close attention network properties. Therefore, the evaluation of a software, considering the demand and application software, weight setting attributes can be more accurate, the subjective preference of these with experts or evaluators are closely related. This paper designs a set of species preference subjective weights.

3.3. Weigh Setting of the Software Running Environment Attributes

Compare with the following six kinds of objective weighting method: Principal Component Analysis, Factor Analysis, TOPSIS, Rank-sum ratio, grey correlation method, entropy method, and the study [24] found that the evaluation results of objective method cannot be accepted. So, in this section, a subjective group preference weight setting algorithm is designed with comprehensive decision makers and experts. Here are the details of the algorithm.

Suppose that P decision-makers or experts provide the weight preference ordinals on attributes set $A=\{A_1,A_2,\ldots,A_M\}$ The set of alternatives ranked in the k_{th} position by decision-maker D_i is $A_{ik}=\{A_j \in A\}$, here, the decision-maker can give any number of belonging to the A attribute in the one location. For example, suppose there are two decision makers (D_1, D_2) and the four attributes $\{A_1, A_2, A_3, A_4\}$. Two weight preference ordinals are given by the decision-makers respectively:

$A_{11}=\{A_2\}$, $A_{12}=\{A_3\}$,

$A_{13}=\{A_1\}$, $A_{14}=\{A_4\}$;

$A_{21}=\{A_2\}$, $A_{22}=\{A_2,A_3\}$,

$A_{23}=\{A_1,A_3\}$, $A_{24}=\{A_1,A_3,A_4\}$.

For the convenience of calculation, weight preference ordinals are denoted by the matrix A_{kj}^{i} , if the decision-maker D_i puts the j_{th} attribute in the k_{th} position, then $a_{kj}=1$, else $a_{kj}=0$.

The weight preference matrix corresponding to the two weight preference ordinals are given by decision maker:

$$A_{kj}^{1} = (a_{kj}^{1})_{4\times 4} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad A_{kj}^{2} = (a_{kj}^{2})_{4\times 4} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{bmatrix}$$

If we do not consider the decision-makers weights or decision-makers weights are the same, we can think that each decision makers weights is 1, then, statistic the number (t_{ki}) of the i_{th} can be a place for i_{th} attribute through the weight preference ordinals matrix A_{kj}^{i}, so we can get the statistical weight preference matrix; if considering the decision maker's weight, assuming the P decision-makers and their weights are P=($\omega_1, \omega_2, \ldots, \omega_P$),the weight of the decision-makers can be allocated according to the size of the responsibility on project and the prestige in the decision-making group, the weight preference matrix A_{kj}^{i} and the weight of the decision-makers need to multiply, and then make a statistic and obtain a statistic weight preference matrix with decision-makers own weight information. That is:

$$T_{ki} = \sum_{i=1}^{P} A_{kj}^{i} \omega_i \qquad (3\text{-}3\text{-}1)$$

With the last example, if decision-makers have the same weight, the statistical weight preference matrix T_{ki} is:

$$T_{kj} = (t_{kj})_{4\times 4} = \begin{bmatrix} 0 & 2 & 0 & 0 \\ 0 & 1 & 2 & 0 \\ 2 & 0 & 1 & 0 \\ 1 & 0 & 1 & 2 \end{bmatrix}$$

Suppose the importance of the k_{th} position is expressed as:

I_k=1-(k-1)/M (3-3-2)

Finally, calculate the weight preference ordinals value of the attributes:

P=[T_{kj}]^T* I_k (3-3-3)

By the equation (2), we get I_1=1, I_2=0.75, I_3=0.5, I_4=0.25
Which denoted by vector:

I = (1, 0.75, 0.5, 0.25)^T

By the equation (3), we get weight preference ordinals value of the 4 attributes:

P=[T_{kj}]^T* I = (1.25,2.75,2.25,1)

By comparing the weight preference ordinals value, we can draw the attribute's weight preference relationship. They are $A_2 \succ A_3 \succ A_1 \succ A_4$.

According to the above method to calculate the attribute weight preference ordinals value, we can get corresponding attribute weights after the normalized.

With the last example, let us see the A sets as attribute sets and the 4 attribute weights are:

(0.172, 0.379, 0.311, 0.138).

3.4. The Process of the Software Trustworthiness Evaluation Model based on Subjective Logic

Evaluation model of the flow diagram shown in Figure 1 and evaluation model of the steps are as follows:

The first step is the software monitoring and event collection. The system monitor software behavior and collect software runtime context state information (running scene information), and then, delivery the collected information to event storage.

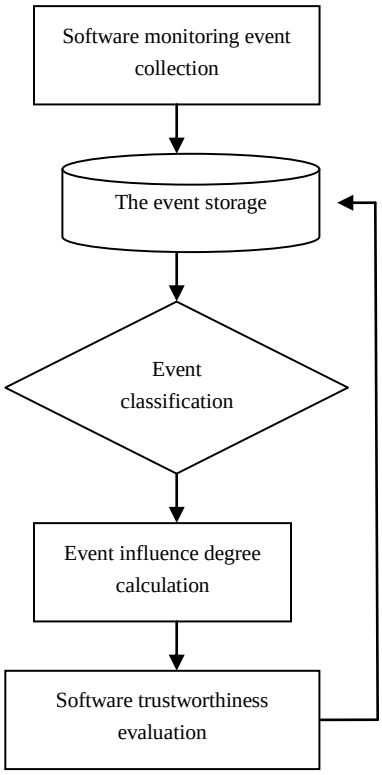


Figure 1. Evaluation Flow Diagram

The second step, the event database storage and standardized the software running at steady and run-time state of relevant environmental attribute information.

The third step, when the software running environment changes, event sensor should have such knowledge, which can determine whether the change of attribute belonging to the normal or abnormal behavior. Such as software behavior monitor, and then, the event is divided into positive or negative events.

The fourth step, according to the current evaluation strategy (see details of the coarse-grained or Fine-grained strategy), calculate the influence of positive and negative events on the software trustworthiness.

The fifth step, according to subjective logic thought fusion the trustworthiness influence quantity in all the observation periods and evaluates software trustworthiness dynamically.

3.5. Evaluation Algorithm

The following will be described the specific evaluation algorithm in fourth steps and fifth. Suppose there are an attribute index set $\{x_1, x_2, \ldots, x_n\}$, x_{ij} indicates that the attribute i in j_{th} time observed value, which forms a matrix denoted by $X_{ij} = (x_{ij})_{m\times n}$. Needs to be pointed out, the M observations can be composed by the key checkpoints,

regular checks and random failure events, therefore, it makes the model with high real-time and flexible dynamic.

$$X_{ij} = (x_{ij})_{m\times n} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix}$$

Attributes for the evaluation scheme usually has the following types: The benefit attributes are those attributes for maximization such as profit, income, and so on, whose values are always the larger the better. The cost attributes are those for minimization such as cost, crime rate, whose values are the smaller the better. The stability attributes are those attributes for tends to a fixed value and in this paper we call it stable core.

In order to balance the calculation accuracy and consumption, the calculation strategy about the influence of the change of the software running environment to the software trustworthiness is divided into coarse-grained and fine-grained.

For the attributes column x_i in matrix, there is a training standard value $\tilde{X}_i$. Set the attribute i positive events number r_i^+ and negative events number s_i^-. Detailed definition sees in Section 3.5.1 and 3.5.2.

3.5.1. The Coarse-grained Strategy: For any concerned attribute i, if the changes of the attribute in actual observation value has produced an event that can be found by software monitor, judge the nature of the event to the change according to the type of attribute (cost type, stable type and benefit type). The following is the definition of positive and negative events about three type of attribute.

Cost type: find the maximum value of the attribute as training standard value $\tilde{X}_i$ which is not influence the software trustworthiness through training. That is, if the actual observation value greater than $\tilde{X}_i$, it will have an impact on the software trustworthiness. The event defined as a negative event and the attribute negative event number plus 1, otherwise, the event defined as a positive event and the attribute positive event number plus 1.

Benefit type: find the minimum value of the attribute as training standard value $\tilde{X}_i$, which is not influence the software trustworthiness through training. That is, if the actual observation value less than $\tilde{X}_i$, it will have an impact on the software trustworthiness. The event defined as a negative event and the attribute negative event number plus 1, otherwise, the event defined as a positive event and the attribute positive event number plus 1.

Stable type: find the minimum and maximum value of the attribute as training standard interval range which is not influence the software trustworthiness through training. That is, if the actual observation value is in this range, it will have an impact on the software trustworthiness. The event defined as a positive event and the attribute positive event number plus 1, otherwise, the event defined as a negative event and the attribute negative event number plus 1.

Cumulative m times observations, we get the positive (r_i^+) and negative (s_i^-) events number of attribute i. On the basis of the subjective logic formula (3-1-4) and (3-1-3), the computing formula of software trustworthiness expectation evaluation is as follow:

$$Trust = \frac{\sum_{i=1}^{n} r_i^+ + W \sum_{i=1}^{n} a_i^+}{W + \sum_{i=1}^{n} r_i^+ + \sum_{i=1}^{n} s_i^-} \quad (3\text{-}5\text{-}1)$$

Needs to be pointed out, in the coarse-grained strategy, it is not consider the influence degree of event to the software trustworthiness precisely. It is also not consider the weight of each attribute. Therefore, this strategy is simple and effective with low computational complexity. The coarse-grained strategy is relatively simple. We describe the detail of the fine-grained strategy in the following.

3.5.2. The Fine-grained Strategy: In the fine-grained strategy, it need measure the influence degree of this attribute change to the software trustworthiness exactly. Taking the benefit type attribute as an example, the same as the definition in the coarse-grained strategy, if the actual observation value great than training standard value $\tilde{X}_i$, the event defined as a positive event, shown in Figure 2. Both the observed value e and f are greater than the training standard value. They are both positive event. But there are difference between e and f obviously. That is the influence of f to the software trustworthiness is greater than e. So, we defined the positive influence as follow:

$$k_i^+ = \frac{X_i - \tilde{X}_i}{Max - \tilde{X}_i} \quad (3\text{-}5\text{-}2)$$

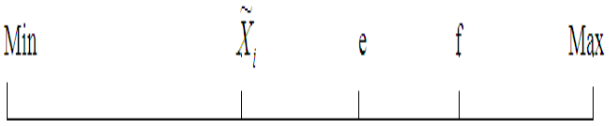


Figure 2. A Diagram about Positive Influence of the Benefit Type Attributes

Next, the influence quantity is defined according to different types of attributes. Set a_i and b_i are minimum and maximum values of attribute i. X_i is the actual observed value.

Benefit type:

The positive influence quantity:

$$k_i^+ = Min\left(\frac{X_i - \tilde{X}_i}{b_i - \tilde{X}_i}, 1\right) \quad (3\text{-}5\text{-}3)$$

The negative influence quantity:

$$k_i^- = \frac{\tilde{X}_i - X_i}{b_i - X_i} \quad (3\text{-}5\text{-}4)$$

$\tilde{X}_i = a_i$.

Cost type:

The positive influence quantity:

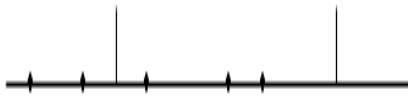
$$k_i^+ = Min\left(\frac{\tilde{X}_i - X_i}{\tilde{X}_i - a_i}, 1\right) \quad (3\text{-}5\text{-}5)$$

The negative influence quantity:

$$k_i^- = \frac{X_i - \tilde{X}_i}{X_i - a_i} \qquad (3\text{-}5\text{-}6)$$

$\tilde{X}_i = b_i$.

For the stable type, there are the following five situations.



A B M D C E N

Figure 3. A Diagram about Influence of the Stable Type Attributes

As shown in Figure 3, M, N were training standard value, the corresponding threshold interval [a_i, b_i], C is stable core value c_i.

Situation one, if the observed value is in the threshold range, it is positive event, and arbitrary value in the interval has the same influence on software trustworthiness without difference, namely, the two points D, E have no difference. Otherwise, negative events, and there is no difference between A and B.

Situation two, if the observed value is in the threshold range, it is positive event, but the difference influence on software trustworthiness. Namely, the two points there is difference between D and E. The distance between E and stable core C is closer than D, therefore, E has a greater influence to the software trustworthiness than D. Otherwise, negative events, and there is no difference between A and B.

Situation three, if the observed value is in the threshold range, it is positive event, and arbitrary value in the interval has the same influence on software trustworthiness without difference, namely, the two points D, E have no difference. Otherwise, negative events, but the difference influence on software trustworthiness. Namely, the two points there is difference between A and B. The distance between A and stable core C is farther than B, therefore, A has a greater influence to the software trustworthiness than B.

Situation four, there is difference between A and B, meanwhile, there is difference between D and E.

Situation five, the attribute value tends to a fixed value (stable core value c_i). That means, if the observed value is equal to c_i, it is positive event and the positive influence is 1, otherwise, the negative positive influence is 1.

For the above five cases, if the two points are no difference or there is no stable core, taking situation one as example, the influence degree of trustworthiness is 1.

Calculation formula is given below only in situation four, other similar. Positive and negative influences of the trustworthiness are:

$$X_i \in [a_i, c_i], k_i^+ = \frac{X_i - a_i}{c_i - a_i} \qquad (3\text{-}5\text{-}7)$$

$$X_i \in [c_i, b_i], k_i^+ = \frac{b_i - X_i}{b_i - c_i} \qquad (3\text{-}5\text{-}8)$$

$$X_i < a_i, \; k_i^- = \frac{a_i - X_i}{c_i - X_i} \qquad (3\text{-}5\text{-}9)$$

$$X_i > b_i,\ k_i^- = \frac{X_i - b_i}{X_i - c_i} \qquad (3\text{-}5\text{-}10)$$

To sum up, the weight of the attribute i is ω_i . The calculation method of ω_i see in Section 3.3.

On the basis of the subjective logic theory, calculation formula for software trustworthiness:

$$Trust = \frac{\sum_{i=1}^{n} \omega_i r_i^+ + W \sum_{i=1}^{n} \omega_i a}{W + \sum_{i=1}^{n} \omega_i r_i^+ + \sum_{i=1}^{n} \omega_i s_i^-} \qquad (3\text{-}5\text{-}11)$$

4. Simulation Experiment

In order to verify the validity of the model, the simulation experiment is carried out in the CPU Intel (R) Core (TM) 2 Duo CPU E7500 2.93GHz, memory 1.98GB, Windows XP system. The evaluation software is 360 browsers and the measured access site is NetEase. Select CPU usage (peak), occupied memory, file system size, connections number, response time and network speed six monitoring environmental attributes as an example, to evaluate the trustworthiness of the software. Using 360 flow firewall and Window task manager measure the experimental data. In a relatively stable environment, the training standard values get by 100 times of observation environment attribute, the results in Table 1.

Table 1. Environmental Attribute Standard Value

attributes standard	CPU usage (%)	occupied memory(k)	connections number	file system size (byte)	response time (s)	network speed (kb/ s)
maximum value	51	160238	3	2377152	2.2	256
minimum value	40	98056	3	2377152	0.23	183

Taking1 10_{th} times observation as a period, a total of 10 periods, the observation data in third period as an example, see Table 2.

Table 2. The Observation Data in Third Period

attributes times	CPU usage (%)	occupied memory(k)	connections number	file system size (byte)	response time (s)	network speed (kb/ s)
1	42	101924	3	2377152	0.34	251
2	46	105368	3	2377152	0.23	194
3	40	108416	3	2377152	0.23	229
4	49	103244	3	2377152	0.23	236
5	50	113964	3	2377152	0.31	235
6	48	105660	3	2377152	0.25	230
7	46	107252	3	2377152	1.32	217
8	47	119944	3	2377152	0.77	225
9	47	115248	3	2377152	1.5	228
10	45	105164	3	2377152	1.34	239

In the five attributes, CPU usage and occupied memory for these two attributes are stable type (in situation five). The connections number and file system size for these two

attributes are also stable type (in situation one). The response time is cost type, while the net work speed is benefit type. According to the attribute types, respectively, using formula (3-5-2) ~ (3-5-10), the influence of positive and negative events is calculated for each of the software trustworthiness, the results in Table 3.

Table 3. The Positive Influence on the Software Trustworthiness (the Third Periodic)

attributes times	CPU usage	occupied memory	connections number	file system size	response time	network speed
1	1.000	1.000	1.000	1.000	0.952	0.932
2	1.000	1.000	1.000	1.000	1.000	0.151
3	1.000	1.000	1.000	1.000	1.000	0.630
4	1.000	1.000	1.000	1.000	1.000	0.726
5	1.000	1.000	1.000	1.000	0.965	0.712
6	1.000	1.000	1.000	1.000	0.991	0.644
7	1.000	1.000	1.000	1.000	0.520	0.466
8	1.000	1.000	1.000	1.000	0.762	0.575
9	1.000	1.000	1.000	1.000	0.441	0.616
10	1.000	1.000	1.000	1.000	0.511	0.767

There are five experts or evaluators (D_1, D_2, D_3, D_4, D_5) with the same weight and give the six attributes {A_1, A_2, A_3, A_4, A_5, A_6} weight preference matrix:

	1st	2nd	3rd	4th	5th	6th
D_1	$\{A_5,A_6\}$	$\{A_1,A_5,A_6\}$	$\{A_1,A_2,A_4\}$	$\{A_1,A_2,A_4\}$	{-}	$\{A_3\}$
D_2	$\{A_6\}$	$\{A_2,A_6\}$	$\{A_2,A_4,A_5,A_6\}$	$\{A_1,A_4,A_5,A_6\}$	$\{A_1,A_4,A_5,A_6\}$	$\{A_3,A_6\}$
D_3	$\{A_2\}$	$\{A_6\}$	$\{A_5\}$	$\{A_1\}$	$\{A_4\}$	$\{A_3\}$
D_4	$\{A_6\}$	$\{A_2,A_6\}$	$\{A_2,A_5\}$	$\{A_1,A_4,A_5\}$	$\{A_1,A_3,A_4\}$	$\{A_1,A_3,A_4\}$
D_5	$\{A_6\}$	$\{A_2,A_5,A_6\}$	$\{A_2,A_5,A_6\}$	$\{A_4\}$	$\{A_1,A_4\}$	$\{A_3\}$

According to the weight calculation algorithm described in Section 3.3, the six attributes weight is {0.138, 0.197, 0.044, 0.143, 0.167, 0.310}. Using formula (3-5-11), calculate the software trustworthiness expectation evaluation is 0.905 (W=2, a=0.5).

5. Conclusions

This paper proposes a software trustworthiness evaluation model based on subjective logic. Difference from other works, this model examine the changes of software running environment, consider the impact on the software trustworthiness, and build the STE model with subjective logic. The model can monitor the software running state in real-time and evaluate the software trustworthiness dynamically. In addition, on the setting of weight aspect, a subjective group preference weight setting algorithm is designed. Simulation results show that the model is reasonable and effective can be more accurate to evaluate the software trustworthiness. This work puts forward from a new perspective method to describe the software trustworthiness with hoping that the work can promote the development of STE theory and its application.

Acknowledgements

This work was supported in part by The National Natural Science Foundation of China (61170254, 60873203), the university of Hebei province science and technology research program (ZH2012029), the natural science foundation of Hebei province (F2014201117), the project as level of school of Handan colledge in 2014 (Electronic commerce model

based on extended subjective logic, 14209) and the project of Handan city science and technology research and development program(The internet of things node authentication technology research based on trusted computing, 1221103041G).

References

[1] Z. Zheng, S.-l. Ma and W. Li, "Dynamic characteristics and evolution complexity of the software trustworthiness", Science in China, vol. 39, **(2009),** pp. 946-950.
[2] K. Liu, Z. Shan and J. Wang, "Overview on major research plan of trustworthy software", Bulletin of National Natural Science Foundation of China, vol. 22, **(2008),** pp. 145-151.
[3] M. Mohammad and V. Alagar, "A formal approach for the specification and verification of trustworthy component-based systems", Journal of Systems and Software, vol. 84, **(2011),** pp. 77-104.
[4] R. Calinescu, C. Ghezzi and M. Kwiatkowska, "Self-adaptive software needs quantitative verification at runtime", Communications of the ACM, vol. 55, **(2012),** pp. 69-77.
[5] Y. Wang, C. Liu and W. Zhang, "Knowledge guided software trustworthiness requirements elicitation", Chinese Journal of Computers, vol. 34, no. 11, (**2011**), pp. 2165-2175.
[6] G. Si, Y. Ren, J. Xu and J. Yang, "A Dependability Evaluation Model for Internetware Based on Bayesian Network. Journal of Computer Research and Development, vol. 49, **(2012),** pp. 1028-1088.
[7] X. Wang, S. Liu and T. Bao, "An Evidence-Driven Framework for Trustworthiness Evaluation of Software Based on Rules", Chinese Journal of Electronics, vol. 21, **(2012),** pp. 589-593.
[8] S. Ding, X.-J. Ma and S.-L. Yang, "A software trustworthiness evaluation model using objective weight based evidential reasoning approach", Knowl. Inf. Syst., vol. 33, **(2012),** pp. 171–189.
[9] Y. Liu and R. Du, "Trust model of software behaviors based on check point risk assessment", Journal of Xidian University, vol. 39, **(2012),** pp. 179-190.
[10] X. Zhou, J. Zhan and M. Li, "Original cause of software untrustworthiness: distrustable factor", Systems Engineering-Theory & Practice, vol. 31, **(2011),** pp. 2410-2418.
[11] L. Gu, Y. Guo and H. Wang, "Runtime software trustworthiness evidence collection mechanism based on TPM", Journal of Software, vol. 21, **(2010),** pp. 373-387.
[12] C. Bettini, O. Brdiczka and K. Henricksen, "A survey of context modelling and reasoning techniques", Pervasive and Mobile Computing, vol. 6, **(2010),** pp. 161-180.
[13] B. Ding, H. M. Wang and D. X. Shi, "Component model supporting trustworthiness-oriented software evolution", Journal of Software, vol. 22, **(2011),** pp. 17-27.
[14] A. Jøsang, "A Logic for Uncertain Probabilities", International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, vol. 9, **(2001),** pp. 1-31.
[15] A. Jøsang, R. Hayward and S. Pope, "Trust Network Analysis with Subjective Logic", 29th Australasian Computer Science Conference, vol. 48, **(2006),** pp. 85-94.
[16] A. Jøsang, R. Ismail and C. Boyd, "A Survey of Trust and Reputation Systems for Online Service Provision", Decision Support Systems, vol. 43, **(2007),** pp. 618-644.
[17] A. Jøsang, "Conditional Reasoning with Subjective Logic", Multiple-Valued Logic and Soft Computing, vol. 15, **(2008),** pp. 5-38.
[18] A. Jøsang and W. Quattrociocchi, "Advanced Features in Bayesian Reputation Systems", Computer Science, vol. 5695, **(2009),** pp. 105-114.
[19] A. Jøsang, J. Diaz and M. Rifqi, "Cumulative and Averaging Fusion of Beliefs", Information Fusion, vol. 11, **(2010),** pp. 192-200.
[20] A. Jøsang and Z. Elouedi, "Redefining Material Implication with Subjective Logic", Proceedings of the 14th International Conference on Information Fusion, Chicago, USA, **(2011)**, pp. 1-6.
[21] J. Tian, C. Li and X. He, "Trust model based on the multinomial subjective logic and risk mechanism for P2P network of file sharing", Journal of Electronics, vol. 28, **(2011),** pp. 108-117.
[22] J. Wen and H. Wang, "Toward a Software Architectural Design Approach for Trusted Software Based on Monitoring", Chinese Journal of Computers, vol. 33, **(2010),** pp. 2321-2334.
[23] A. Jøsang, "Subjective logic", Draft book Available at: http://persons. unik. no/josang/papers/subjective_logic. pdf, visited, vol. 26, **(2010).**
[24] Y. Liping, P. Yuntao and W. Yishan, "Comparing objective weighting with subjective weighting in Sci-tech education institute assessment", Science Research Management, vol. 4, no. 019, **(2009).**

Authors

Jiao Hongqiang, he was born in 1983, a Ph.D. candidate and lecturer. Email: jhq1983@163.com. His research interests focus on trusted computing, information security and decision making.

Tian Junfeng, he was born in 1965, Ph.D. professor. Email: tjf@hbu.cn. His research interests focus on parallel and distributed computing, trusted computing, information security and network technology.

