Hindawi
Mathematical Problems in Engineering
Volume 2018, Article ID 6982821, 10 pages
https://doi.org/10.1155/2018/6982821

Research Article

Source Codes Oriented Software Trustworthiness Measure Based on Validation

Hongwei Tao [1] **and Jie Zhao** [2]

[1] *College of Computer and Communication Engineering, Zhengzhou University of Light Industry, Zhengzhou, China*
[2] *College of Energy and Power Engineering, Zhengzhou University of Light Industry, Zhengzhou, China*

Correspondence should be addressed to Hongwei Tao; tthhww_811@163.com

Received 11 June 2018; Revised 29 August 2018; Accepted 6 November 2018; Published 19 November 2018

Academic Editor: Ioannis T. Christou

Validation is critical to the success of software trustworthiness measurement. A large number of software trustworthiness measures are proposed; however, most of them are not validated from a theory perspective. Therefore, they lack theoretical foundation and will induce unnecessary cost and useless calculation. In this paper, we bring measurement theory into software trustworthiness measurement, construct a source codes oriented software trustworthiness measure based on extensive structure in the measurement theory, and validate the developed measure by use of axiomatic approaches. Compared with some software trustworthiness measures that are already presented, this measure can evaluate software trustworthiness better from a theory perspective.

1. Introduction

Because of the outstanding problems with software trustworthiness, people pay more and more attention to the research into trustworthy software. One of the core scientific problems in this research is the software trustworthiness measurement [1]. However, few research works carry out theoretical validation for their measures. It is important that the measures we use are valid. That is, measures must measure what they are supposed to measure. Theoretical validation is a required activity for using or defining measures that make sense and is a necessary step for the empirical validation of measures [2]. On the other side, many researches do not present the methods to translate the measure results back to the empirical world. Then it is difficult to use the measure results to provide guidance for the improvement of software trustworthiness. Meanwhile, most researchers in software trustworthiness measurement area do not address scale types. Scale types are very important for meaningful statistical operations. Because many empirical and numerical conditions are not covered by a certain scale type, for example, comparisons of arithmetic means are not a meaningful statistic [3, 4].

There are two main approaches in theoretical validation and they are measurement theory and axiomatic approaches [5, 6]. In measurement theory, the empirical understanding of an attribute is formalized through the definition of an empirical relational system and a numerical relational system is defined to provide measure values for that attribute and relations among these values [7]. Measurement theory presents a clear definition of a measure under the assumption of a homomorphism from the empirical world into the numerical world and gives the conditions for the use of measures on certain scale levels. The major advantage of measurement theory is the representational condition, which attests that the properties of the attributes in the empirical world should be preserved by the measures in the numerical world and vice versa. Part of the empirical understanding of a software attribute may be formally defined as desirable properties of the measures for that attribute. Axiomatic approaches are used with a number of properties for theoretical validation [8, 9]. A property is a condition or a basic assumption of reality. The measure's properties in axiomatic approaches must be seen as properties that characterize the numerical world. However, they indirectly affect the empirical world. Therefore, properties in axiomatic approaches can be used as guidelines for the definition of a measure.

Software trustworthiness measures based on measurement theory and axiomatic approaches are appropriate

methods to deal with problems, like theoretical validation of measures, measurement scales, the empirical interpretation of numerical results, and properties of measures. However, measurement theory and axiomatic approaches are mostly applied to measure internal attributes such as size, complexity, and cohesion or used to validate the software internal attributes measures from theory [2, 9–13]. The fundamental problem of the application of measurement theory or axiomatic approaches in software external attributes measurement is that, for many software external attributes, it is not known what the empirical relational system looks like. In order to make software trustworthiness measurement more rigorous, we used axiomatic approaches to measure software trustworthiness and presented some desirable properties of the measures for software trustworthiness in the view of trustworthy attributes [8], including monotonicity, acceleration, sensitivity, and substitutivity. In this paper, we construct a source codes oriented software trustworthiness measure based on extensive structure in the measurement theory, present four desirable properties of the measures for software trustworthiness from the standpoint of module, and carry out theoretical validation for the developed measure by use of axiomatic approaches.

The rest of this paper is organized as follows. In Section 2 we review some related works. We describe some basic definitions and notations from measurement theory and present extensive structure based software trustworthiness measures in Section 3. In Section 4 we introduce four desirable properties of software trustworthiness measures from the standpoint of module and validate the nonadditive software trustworthiness measure based on extensive structure established in Section 3 by use of axiomatic approaches. The measurement procedure based on the measures built in Section 3 is presented in Section 5. We make a case study in Section 6. Section 7 contains the comparative study. The conclusion and future work come in the last section.

2. Related Works

There are three kinds of software trustworthiness measurements: attribute-based software trustworthiness measurement, behavior-based software trustworthiness measurement, and process-based software trustworthiness measurement [14].

The general framework for the attribute-based software trustworthiness measurement is as follows. The attributes that affect software trustworthiness are firstly selected. Then, measurement models are built to measure these selected attributes. Finally, models and methods are presented to measure software trustworthiness based on measurement results of the selected attributes. Typical models and methods include axiomatic approaches [8, 15–20], fuzzy comprehensive evaluation method [21, 22], rough set theory [23], Bayesian Networks [24–26], evidence theory [27, 28], weakness analysis [29], questionnaire survey and statistical analysis [30–33], and dynamic system [34–36].

The software normal behavior model is firstly built under trustworthy environment in the behavior-based software trustworthiness measurement. Then, the software actual behavior is extracted based on the monitored actual operating statuses of the software at the checkpoint. Finally, the actual behavior is compared with the normal behavior to evaluate software trustworthiness. Typical comparison methods include program slicing [37, 38], data mining [39, 40], and concurrency theory [41–46].

Process-based software trustworthiness measurement uses the collected data describing the behaviors and attributes of the process as evidence to measure software trustworthiness. Classic methods contain Trusted Software Methodology [47, 48], Trustworthy Process Management Framework [49, 50], and so on.

3. Software Trustworthiness Measure Based on Extensive Structure

We first present some basic concepts from measurement theory.

Definition 1 (empirical relational system [7]). Given an attribute, an empirical relational system is an ordered tuple $ERS = (E, R_1, \cdots, R_m, o_1, \cdots, o_n)$, where

(1) E denotes a nonempty set of empirical objects for which we want to measure the attribute;

(2) $R_i (1 \le i \le m)$ are k_i-ary empirical relations on E capturing our intuitive knowledge on the attribute;

(3) $o_j (1 \le j \le n)$ are empirical binary operations on E, i.e., $o_j : E \times E \longrightarrow E$.

The empirical relational system is used to model our empirical understanding of the objects' attributes that we want to measure.

Definition 2 (numerical relational system [7]). Given an attribute, a numerical relational system is an ordered tuple $NRS = (V, S_1, \cdots, S_m, \oplus_1, \cdots, \oplus_n)$, where

(1) V is the set of values that we use to measure the attribute;

(2) $S_i (1 \le i \le m)$ denote k_i-ary relations on V;

(3) $\oplus_j (1 \le j \le n)$ denote numerical binary operations on V, i.e., $\oplus_j : V \times V \longrightarrow V$.

The numerical relational system is purposefully defined to mirror the empirical relational system in the realm of values. The link between the empirical relational system and the numerical relational system is made via measure.

Definition 3 (measure [7]). A function $\mu : E \longrightarrow V$ is said to be a measure.

However, with the exception of empirical objects, the empirical relational system also presents information about our empirical understanding of an attribute of a set of empirical objects. If this information is not considered, any function that maps E into V is a measure according to Definition 3. In order to discard all of those measures that are at variance with our intuition, measurement theory introduces the definition of scale.

Definition 4 (scale [7]). Let $ERS = (E, R_1, \cdots, R_m, o_1, \cdots, o_n)$ be an empirical relational system, $NRS = (V, S_1, \cdots, S_m, \oplus_1, \cdots, \oplus_n)$ be a numerical relational system, and $\mu : E \longrightarrow V$ be a measure. The triple (ERS, NRS, μ) is a scale if and only if for all $i \in \{1, \cdots, m\}$, $j \in \{1, \cdots, n\}$ and for $\forall a_l, \cdots, a_k, b, c \in E$, the following holds:

$$\begin{aligned} &R_i(a_l, \cdots, a_k) = S_i(\mu(a_l), \cdots, \mu(a_k)) \\ &\text{and } \mu\left(b\ o_j\ c\right) = \mu(b) \oplus_j \mu(c). \end{aligned} \tag{1}$$

Extensive structure is one of the most important measurement structures. It is essential in physics and is very important in the area of software measurement too. They are not brought into software external attributes measurement and software trustworthiness measurement. The reason is that it is difficult to build the extensive structure about software external attributes or software trustworthiness. The following is the definition of extensive structure.

Definition 5 (extensive structure [7]). Let A be a nonempty set, R a binary relation on A, and $\circ$ a closed binary operation on A. The empirical relational system $(A, R, \circ)$ is an extensive structure if the following axioms are satisfied:

(1) Weak order: (A, R) is a weak order; i.e., it satisfies transitivity and strong completeness.

(2) Weak associativity: $\forall a, b, c \in A : (a \circ b) \circ c \approx a \circ (b \circ c)$.

(3) Monotonicity: $\forall a, b, \in A : (a, b) \in R \Longleftrightarrow ((a \circ c), (b \circ c)) \in R \Longleftrightarrow ((c \circ a), (c \circ b)) \in R$.

(4) Archimedean: $\forall a, b, c, d \in A$: if $(a, b) \in R_s$, then for any c, d there exists a natural number n, such that $((n.a \circ c), (n.b \circ d)) \in R$.

One has

$$\begin{aligned} a \approx b &\Longleftrightarrow (a, b) \in R \wedge (b, a) \in R, \\ (a, b) \in R_s &\Longleftrightarrow (a, b) \in R \wedge (b, a) \notin R, \\ 1a &= a, \\ (n+1)\, a &= na \circ a. \end{aligned} \tag{2}$$

The most practical use of measurement is the representation theorem of extensive structure.

Theorem 6 (representation theorem of extensive structure [7]). *Suppose A is a nonempty set, $\succeq$ a binary relation on A, $\circ$ a closed binary operation on A, and $\Re$ the set of all real numbers. Then there is a real-valued function $\mu : A \longrightarrow \Re$ satisfying*

$$\begin{aligned} a \succeq b &\Longleftrightarrow \mu(a) \geq \mu(b) \\ \mu(a \circ b) &= \mu(a) + \mu(b) \end{aligned} \tag{3}$$

if and only if $(A, \succeq, \circ)$ is an extensive structure. If another function μ' fulfills these properties, then there exists a positive real value α that

$$\mu' = \alpha\mu \tag{4}$$

holds.

In order to apply extensive structure in software trustworthiness measurement, the first step is to define an empirical relational system about software trustworthiness. Therefore, the software entity must be represented in one way or another. In this paper, we use the software abstraction presented in [12] and define software as a system. However, we modify the definition of module given in [12].

Definition 7 (system [12]). A system S is represented as a pair (E, R), where E is the set of elements of S and R represents a binary relation on $E (R \subseteq E \times E)$ expressing the relationships between S's elements.

Definition 8 (module). For a given system $S = (E, R)$, a module $m = (E_m, R_m)$ is a system such that

$$\left(\forall e \in E_m \Longrightarrow e \in E\right) \wedge R_m \subseteq R. \tag{5}$$

As an example, E can be defined as the set of code statements and R as the set of data flows from one statement to another. A module m may be a code segment or a subprogram.

Given a system $S = (E, R)$, let MS be the set of all the modules in S. Then MS is the set that are to be measured. Suppose $m_1 = (E_{m_1}, R_{m_1})$, $m_2 = (E_{m_2}, R_{m_2}) \in MS$ and let

$$\begin{aligned} &R_{m_1 m_2} \\ &\quad = \left\{(e_1, e_2) \mid e_1 \in E_{m_1} \wedge e_2 \in E_{m_2} \wedge (e_1, e_2) \in R\right\}. \end{aligned} \tag{6}$$

Then, we define the binary operation on MS as

$$m_1 \circ_T m_2 = \left(E_{m_1} \uplus E_{m_2}, R_{m_1} \cup R_{m_2} \cup R_{m_1 m_2}\right), \tag{7}$$

where $E_{m_1} \uplus E_{m_2}$ is a multiset composed of the elements in E_{m_1} and E_{m_2}. From the definition of module, we know that $m_1 \circ_T m_2 \in MS$.

Now we need to define the empirical ordering $\succeq_T$: equally or more trustworthy than. Software trustworthiness is the ability of software to satisfy user expectation with its behaviors and results. Both the software behaviors and the results are decided by the implementations of software. Then the software trustworthiness can be obtained by computing the quantification of the conformance of the implementations adopted by software with the implementations expected by users. The implementations expected by users are referenced as the properties and functions that the program elements and program unit should carry if a program is trustworthy; for example, a variable should be initialized before used [51]. Moreover, software is composed of the modules through some relations and the module is the collection of program elements. So we let $\succeq_T$ be determined by the following observation criterion: If $m_1 = (E_{m_1}, R_{m_1}), m_2 = (E_{m_2}, R_{m_2}) \in MS$, then $m_1 \succeq_T m_2$ is observed if and only if when the ratio of the number of program elements with the implementations expected by users in E_{m_1} to the number of program elements in E is greater or equal than the ratio of the number of program elements with the implementations expected by users in E_{m_2} to the number of program elements in E.

Claim 1. $(MS, \succeq, \circ_T)$ is an extensive structure.

Proof. (1) Transitivity: let $m_1 = (E_{m_1}, R_{m_1}), m_2 = (E_{m_2}, R_{m_2}), m_3 = (E_{m_3}, R_{m_3}) \in MS$. Suppose the ratios of the number of program elements with the implementations expected by users in $E_{m_1}, E_{m_2}, E_{m_3}$ to the number of program elements in E are k_1, k_2, and k_3, respectively.

Then, from the definition of $\succsim_T$, we know that $m_1 \succsim_T m_2$ implies $k_1 \geq k_2$ and $m_2 \succsim_T m_3$ implies $k_2 \geq k_3$. As the relation $\geq$ is transitive for real numbers, then it holds that $k_1 \geq k_3$, leading to the observation that $m_1 \succsim_T m_3$.

(2) Strong completeness: since any pair of real numbers can be compared using $\geq$, then strong completeness is shown by the definition of $\succsim_T$.

From (1) and (2), we can get $(MS, \succsim)$ is a weak order.

(3) Weak associativity: because

$$m_1 \circ_T m_2 = \left(E_{m_1} \uplus E_{m_2}, R_{m_1} \cup R_{m_2} \cup R_{m_1 m_2}\right), \tag{8}$$

therefore

$$\begin{aligned}\left(m_1 \circ_T m_2\right) \circ_T m_3 = \Big(\left(E_{m_1} \uplus E_{m_2}\right) \uplus E_{m_2}, R_{m_1} \cup R_{m_2} \\ \cup R_{m_3} \cup R_{m_1 m_2} \cup R_{m_1 m_3} \cup R_{m_2 m_3} \cup R_{m_1 m_2 m_3}\Big).\end{aligned} \tag{9}$$

Similarly, since

$$m_2 \circ_T m_3 = \left(E_{m_2} \uplus E_{m_3}, R_{m_2} \cup R_{m_3} \cup R_{m_2 m_3}\right), \tag{10}$$

then

$$\begin{aligned}m_1 \circ_T \left(m_2 \circ_T m_3\right) = \Big(E_{m_1} \uplus \left(E_{m_2} \uplus E_{m_2}\right), R_{m_1} \cup R_{m_2} \\ \cup R_{m_3} \cup R_{m_1 m_2} \cup R_{m_1 m_3} \cup R_{m_2 m_3} \cup R_{m_1 m_2 m_3}\Big).\end{aligned} \tag{11}$$

Notice that associativity is satisfied by $\uplus$, therefore

$$\left(m_1 \circ_T m_2\right) \circ_T m_3 \approx m_1 \circ_T \left(m_2 \circ_T m_3\right). \tag{12}$$

(4) Monotonicity: we first prove that $\forall m_1 = (E_{m_1}, R_{m_1}), m_2 = (E_{m_2}, R_{m_2}), m_3 = (E_{m_3}, R_{m_3}) \in MS : m_1 \succsim_T m_2 \Longrightarrow m_1 \circ_T m_3 \succsim_T m_2 \circ_T m_3$.

The number of program elements with the implementations expected by users in $E_{m_1} \uplus E_{m_3} (E_{m_2} \uplus E_{m_3})$ is the sum of the number of program elements with the implementations expected by users in $E_{m_1} (E_{m_2})$ and that in $E_{m_3} (E_{m_3})$. Because $m_1 \succsim_T m_2$ implies the number of program elements with the implementations expected by users in E_{m_1} is equal or more than the number of program elements with the implementations expected by users in E_{m_2}, therefore, the number of program elements with the implementations expected by users in $E_{m_1} \uplus E_{m_3}$ is equal or more than the number of program elements with the implementations expected by users in $E_{m_2} \uplus E_{m_3}$; then it follows that

$$m_1 \circ_T m_3 \succsim_T m_2 \circ_T m_3. \tag{13}$$

Likewise, if $m_1 \circ_T m_3 \succsim_T m_2 \circ_T m_3$, we can deserve that $m_1 \succsim_T m_2$.

In a similar way, we can prove that $\forall m_1 = (E_{m_1}, R_{m_1}), m_2 = (E_{m_2}, R_{m_2}), m_3 = (E_{m_3}, R_{m_3}) \in MS : m_1 \succsim_T m_2 \Longleftrightarrow m_3 \circ_T m_1 \succsim_T m_3 \circ_T m_2$.

(5) Archimedean: since $(MS, \succsim_T, \circ_T)$ satisfies monotonicity, then it is easy to get that Archimedean holds for $(MS, \succsim_T, \circ_T)$.

In summary, $(MS, \succsim_T, \circ_T)$ is an extensive structure. □

Definition 9 (an additive module trustworthiness measure based on extensive structure). For a given software system $S = (E, R)$, let MS be the set of all the modules in S and $\mathfrak{R}^+$ be the set of all positive real number. Suppose $m = (E_m, R_m), m_1 = (E_{m_1}, R_{m_1}), m_2 = (E_{m_2}, R_{m_2}) \in MS$. Denote the number of program elements with the implementations expected by users in E_m by $\#(E_m)_{expect}$ and the number of program elements in E by $\#(E)$. Then, the additive module trustworthiness measure based on extensive structure T_1 is defined as $T_1 : (MS, \succsim_T, \circ_T) \longrightarrow (\mathfrak{R}^+, \geq, +)$, where

$$T_1(m) = \frac{\#\left(E_m\right)_{expect}}{\#(E)}. \tag{14}$$

According to the representation theorem of extensive structure and the definition of T_1, it is easy to obtain the following result.

Claim 2. (1) T_1 assumes the extensive structure $(MS, \succsim_T, \circ_T)$.

(2) For $\forall m_1 = (E_{m_1}, R_{m_1}), m_2 = (E_{m_2}, R_{m_2}) \in MS$,

$$T_1\left(m_1 \circ_T m_2\right) = T_1\left(m_1\right) + T_1\left(m_2\right), \tag{15}$$

i.e., T_1 is an additive homomorphism from $(MS, \succsim_T, \circ_T)$ to $(\mathfrak{R}^+, \geq, +)$.

Definition 10 (a nonadditive module trustworthiness measure based on extensive structure). For a given software system $S = (E, R)$, let MS be the set of all the modules in S and $\mathfrak{R}^+$ be the set of all positive real numbers. Suppose $m = (E_m, R_m), m_1 = (E_{m_1}, R_{m_1}), m_2 = (E_{m_2}, R_{m_2}) \in MS$, $0 < \rho < 1$, $0 < \beta$. Then, the nonadditive module trustworthiness measure based on extensive structure T_2 is defined as $T_2 : (MS, \succsim_T, \circ_T) \longrightarrow (\mathfrak{R}^+, \geq, \oplus)$, where

$$\begin{aligned}T_2(m) &= \left(\frac{1}{\beta} T_1(m)\right)^{1/\rho} \\ T_2\left(m_1\right) \oplus T_2\left(m_2\right) &= \left(\left(T_2\left(m_1\right)\right)^{\rho} + \left(T_2\left(m_2\right)\right)^{\rho}\right)^{1/\rho}.\end{aligned} \tag{16}$$

Claim 3. (1) T_2 assumes the extensive structure $(MS, \succsim_T, \circ_T)$, and for $\forall m_1 = (E_{m_1}, R_{m_1}), m_2 = (E_{m_2}, R_{m_2}) \in MS$,

$$T_2\left(m_1 \circ_T m_2\right) = \left(\left(T_2\left(m_1\right)\right)^{\rho} + \left(T_2\left(m_2\right)\right)^{\rho}\right)^{1/\rho}, \tag{17}$$

i.e., T_2 is a nonadditive homomorphism from $(MS, \succsim_T, \circ_T)$ to $(\mathfrak{R}^+, \geq, \oplus)$.

(2) T_2 can be used as a ratio scale.

Proof. (1) Let $m_1 = (E_{m_1}, R_{m_1}), m_2 = (E_{m_2}, R_{m_2}), m_3 = (E_{m_3}, R_{m_3}) \in MS$. Because

$$T_2(m) = \left(\frac{1}{\beta} T_1(m)\right)^{1/\rho}, \tag{18}$$

then

$$T_1(m) = \beta \left(T_2(m)\right)^{\rho}, \tag{19}$$

and

$$T_1(m_1 \circ_T m_2) = \beta (T_2(m_1 \circ_T m_2))^{\rho}. \tag{20}$$

From Claim 2, we know that

$$\begin{aligned} T_1(m_1 \circ_T m_2) &= T_1(m_1) + T_1(m_2) \\ &= \beta (T_2(m_1))^{\rho} + \beta (T_2(m_2))^{\rho}. \end{aligned} \tag{21}$$

Then we can get

$$T_2(m_1 \circ_T m_2) = \left((T_2(m_1))^{\rho} + (T_2(m_2))^{\rho} \right)^{1/\rho}. \tag{22}$$

Because $0 < \rho < 1$, therefore T_2 is a nonadditive measure and can be modified to the additive measure T_1 by a strictly monotonic function. From the result in [18], it follows that T_2 assumes the extensive structure $(MS, \succsim_T, \circ_T)$ too.

(2) Noticing that

$$\begin{aligned} &\left((\alpha T_2(m_1))^{\rho} + (\alpha T_2(m_2))^{\rho} \right)^{1/\rho} \\ &= \alpha \left((T_2(m_1))^{\rho} + (T_2(m_2))^{\rho} \right)^{1/\rho} \\ &= \alpha T_2(m_1 \circ_T m_2), \end{aligned} \tag{23}$$

it follows that T_2 can be used as a ratio scale.

To sum up, the conclusion can be verified. □

In the following, on the basis of the module trustworthiness measures described in Definitions 9 and 10, we present an additive software trustworthiness measure based on extensive structure and a nonadditive software trustworthiness measure based on extensive structure. For a given software system $S = (E, R)$, let $M(\subseteq MS)$ be a collection of modules of S such that

$$\forall e \in E \left(\exists m = \langle E_m, R_m \rangle \in M (e \in E_m) \right) \tag{24}$$

and

$$\begin{aligned} &\forall m_i = \left\langle E_{m_i}, R_{m_i} \right\rangle, m_j = \left\langle E_{m_j}, R_{m_j} \right\rangle \\ &\in M \left(E_{m_i} \cap E_{m_j} = \emptyset \right), \end{aligned} \tag{25}$$

i.e, the set of elements E of S is partitioned into the sets of elements of the modules in M. Suppose the number of the module in M is n and denote them by $m_1, \cdots, m_n$ separately.

Definition 11 (an additive software trustworthiness measure based on extensive structure). For a given software system $S = (E, R)$, an additive software trustworthiness measure based on extensive structure $T'(S)$ is defined as

$$\begin{aligned} T'(S) &= T_1(m_1 \circ_T m_2 \circ_T \cdots \circ_T m_n) \\ &= T_1(m_1) + T_1(m_2) + \cdots + T_1(m_n). \end{aligned} \tag{26}$$

Definition 12 (a nonadditive software trustworthiness measure based on extensive structure). For a given software system $S = (E, R)$, a nonadditive software trustworthiness measure based on extensive structure $T(S)$ is defined as

$$\begin{aligned} T(S) &= T_2(m_1 \circ_T m_2 \circ_T \cdots \circ_T m_n) \\ &= \left(T_2(m_1)^{\rho} + T_2(m_2)^{\rho} + \ldots T_2(m_n)^{\rho} \right)^{1/\rho}. \end{aligned} \tag{27}$$

4. Numerical Relational System Based on Axiomatic Approaches

There may be many nonadditive software trustworthiness measures that assume the extensive structure $(MS, \succsim_T, \circ_T)$ and can be used as a ratio scale. Now the problem is how to prove T is a proper software trustworthiness measure. Axiomatic approaches have already been used to check if the existing measures comply with a specific set of properties for the attribute that they purport to measure. We once used axiomatic approaches to measure software trustworthiness and presented some desirable properties of software trustworthiness measures from the standpoint of trustworthy attributes [8]. Consulting the properties given in [8], in this section we give four desirable properties of software trustworthiness measures in the view of module and prove that T complies with these four properties.

For generality, let

$$T(y_1, y_2, \cdots, y_n) = \left(y_1^{\rho} + y_2^{\rho} + \cdots + y_n^{\rho} \right)^{1/\rho}, \tag{28}$$

where

(i) $0 < \rho < 1$: a parameter related to the substitutivity between module trustworthiness

(ii) $0 \le y_i$: the module trustworthiness obtained through T_2 with $i = 1, \cdots, n$

The desirable properties of software trustworthiness measures in the view of module are depicted below.

(1) Monotonicity

$$\begin{aligned} &T(y_1, \cdots, y_i + \Delta y_i, \cdots, y_n) - T(y_1, \cdots, y_i, \cdots, y_n) \\ &\quad \ge 0, \end{aligned} \tag{29}$$

where $1 \le i \le n$. It means that the increment of a module trustworthiness leads to software trustworthiness increase.

(2) Acceleration

$$\begin{aligned} &\frac{\Delta T(y_1, \cdots, y_i + \Delta y_i, \cdots, y_n) - \Delta T(y_1, \cdots, y_i, \cdots, y_n)}{\Delta y_i} \\ &\quad \le 0, \end{aligned} \tag{30}$$

where $1 \le i \le n$. It states that the increase of the module trustworthiness leads to their utilization efficiency to decrease.

(3) Sensitivity

$$0 \le \frac{\Delta T}{\Delta y_i} \frac{y_i}{T}, \quad 1 \le i \le n. \tag{31}$$

Sensitivity is used to describe the percentage changes of software trustworthiness caused by the percentage changes of module trustworthiness. They should be positive.

(4) Substitutivity

$$\left(\exists c_1, c_2 \in \mathfrak{R}^+ \right) \quad c_1 \le \sigma_{ij} \le c_2, \tag{32}$$

where

$$\sigma_{ij} = \frac{\Delta\left(y_i/y_j\right)}{\Delta\left(\Delta y_i/\Delta y_j\right)} \times \frac{\Delta y_i/\Delta y_j}{y_i/y_j}, \quad 1 \le i, j \le n,\ i \ne j. \tag{33}$$

$\sigma_{ij}(1 \le i, j \le n,\ i \ne j)$ are used to express the difficulty of the substitution between module trustworthiness. Substitutivity implies that the module trustworthiness can substitute each other to some extent.

Claim 4. Monotonicity holds for T.

Proof. From the definition of T, we know that T is a continuously differentiable function. By solving the partial derivatives of T with respect to y_i $(1 \le i \le n)$, we can obtain

$$\frac{\partial T}{\partial y_i} = \left(y_1^\rho + y_2^\rho + \cdots + y_n^\rho\right)^{1/\rho - 1} y_i^{\rho - 1}. \tag{34}$$

Since $0 \le y_i$ $(1 \le i \le n)$. Then

$$\frac{\partial T}{\partial y_i} \ge 0, \quad 1 \le i \le n. \tag{35}$$

Therefore, the conclusion can be verified. □

Claim 5. T satisfies the acceleration.

Proof. By computing the second order partial derivatives of T with respect to y_i $(1 \le i \le n)$, we get that

$$\begin{aligned}\frac{\partial^2 T}{\partial {y_i}^2} &= (1-\rho)\left(y_1^\rho + y_2^\rho + \cdots + y_n^\rho\right)^{1/\rho - 2} \\ &\quad \cdot y_i^{\rho-2}\left(y_i^\rho - \left(y_1^\rho + y_2^\rho + \cdots + y_n^\rho\right)\right).\end{aligned} \tag{36}$$

Because $0 < \rho < 1$ and $0 \le y_i$ $(1 \le i \le n)$, therefore

$$\frac{\partial^2 T}{\partial {y_i}^2} \le 0, \quad 1 \le i \le n. \tag{37}$$

Then we can obtain the conclusion that the acceleration is satisfied by T. □

Claim 6. Sensitivity is satisfied by T.

Proof. For $1 \le i \le n$, by calculating it is easy to get that

$$\frac{\partial T}{\partial y_i}\frac{y_i}{T} = \frac{y_i^\rho}{y_1^\rho + y_2^\rho + \cdots + y_n^\rho}. \tag{38}$$

Due to $0 \le y_i$ $(1 \le i \le n)$, then

$$\frac{\partial T}{\partial y_i}\frac{y_i}{T} \ge 0, \quad 1 \le i \le n. \tag{39}$$

In summary, T is sensitive to module trustworthiness. □

Claim 7. T complies with substitutivity.

Proof. Since T is a continuously differentiable function, and

$$\frac{\partial T}{\partial y_i} = \left(y_1^\rho + y_2^\rho + \cdots + y_n^\rho\right)^{1/\rho - 1} y_i^{\rho - 1}. \tag{40}$$

Then, for $1 \le i, j \le n,\ i \ne j$, we can get

$$d\left(\frac{\partial T/\partial y_j}{\partial T/\partial y_i}\right) = d\left(\frac{y_j^{\rho-1}}{y_i^{\rho-1}}\right) = (1-\rho)\, y_i^{-\rho} y_j^\rho d\left(\frac{y_i}{y_j}\right). \tag{41}$$

According to (33), it follows that

$$\begin{aligned}\sigma_{ij} &= \frac{d\left(y_i/y_j\right)}{d\left(-\left(\partial T/\partial y_j\right)/\left(\partial T/\partial y_i\right)\right)} \\ &\quad \times \frac{-\left(\partial T/\partial y_j\right)/\left(\partial T/\partial y_i\right)}{y_i/y_j} = \frac{1}{1-\rho}, \\ &\qquad 1 \le i, j \le n,\ i \ne j.\end{aligned} \tag{42}$$

Since $0 < \rho < 1$. Then substitutivity holds for T. □

5. The Measurement Procedure Based on the Measures Established in Section 3

The measures given in this paper are used to measure source codes trustworthiness in the software implementation phase, and the source codes can be written in an imperative language or in an object-oriented language. We have presented specifications for the implementations expected by users of the program elements involved in the imperative languages [51]; if you want to use the measures built in this paper to measure the trustworthiness of source codes written in an object-oriented language, you need to propose the specifications for the implementations expected by users of the program elements involved in the corresponding object-oriented language.

The measurement procedure based on the proposed measures in Section 3 consists of four steps as shown in Figure 1. For a given software system $S = (E, R)$, we first find a partition M of MS, where MS is the set of all modules in S. Then, for $\forall m \in MS$, we compute $T_1(m)$ according to Definition 9 in Step 2. For $\forall m \in MS$, based on the results of computation of $T_1(m)$, $T_2(m)$ are calculated according to Definition 10 in Step 3. In the final step, on the basis of the results of Step 3, the software trustworthiness $T(S)$ is obtained by utilizing Definition 12.

6. Case Study

To demonstrate the effectiveness of our measure, we apply T to measure the trustworthiness of quick sort program written in the C language in Algorithm 1. For simplicity, we just consider the program element: variable. Here we use the trustworthy properties of variable given in [51] as variables' implementations expected by users. Let $M = \{m_1 = \langle E_{m_1}, R_{m_1}\rangle, m_2 = \langle E_{m_2}, R_{m_2}\rangle, m_3 = \langle E_{m_3}, R_{m_3}\rangle\}$, where

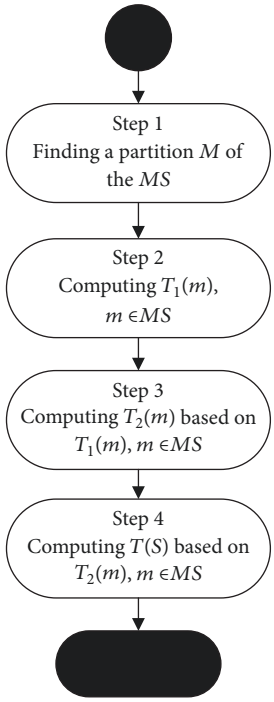

Figure 1: The measurement procedure based on the measures established in Section 3.

(i) E_{m_1} is composed of the variables in line (1)-(18), i.e., $E_{m_1} = \{R, i, j, pivot\}$

(ii) E_{m_2} consists of the variables in line (19)-(28), i.e., $E_{m_2} = \{R, low, high, pivotpos\}$

(iii) E_{m_3} is made up of the variables in line (29)-(42), i.e., $E_{m_3} = \{s[100], i, j\}$

Then $\#(E) = 11$. From the trustworthy properties of variable described in [51], we know that $i, j \in E_{m_1}$ do not have one and only one purpose, the same identifiers are still used in main function, and then they are not expected by users. Moreover, $R \in E_{m_1}$ and $R \in E_{m_2}$ do not have meaningful names; therefore, they are not expected by users. So, we can get $\#(E_{m_1})_{expect} = 1$, $\#(E_{m_2})expect = 3$, and $\#(E_{m_3})expect = 1$. It follows that $T_1(m_1) = 1/11$, $T_1(m_2) = 3/11$, and $T_1(m_3) = 1/11$.

Suppose $\rho = 0.4, \beta = 0.6$; then $T_2(m_1) = (5/33)^{2.5}$, $T_2(m_2) = (5/11)^{2.5}$, $T_2(m_3) = (5/33)^{2.5}$, and

$$\begin{aligned} T(S) &= T_2\left(m_1 \circ_T m_2 \circ_T \cdots \circ_T m_n\right) \\ &= \left(T_2(m_1)^\rho + T_2(m_2)^\rho + \ldots T_2(m_n)^\rho\right)^{1/\rho} \\ &= 0.50. \end{aligned} \tag{43}$$

7. Comparative Study in Terms of Measurement Theory and the Properties Given in Section 4

In this section, we compare T with the decomposition of trustworthy attributes based software trustworthiness measure (DTABSTM) [16], axiomatic approaches based software trustworthiness measure (AABSTM) [17], fuzzy theory based software trustworthiness measure (FTBSTM) [22], and evidence theory based software trustworthiness measure (ERBSTM) [28] through the measurement theory and the properties presented in Section 4. The comparative results are given in Table 1, where $\surd$ represents that the measure holds for corresponding property and $\times$ expresses that the measure does not comply with the corresponding property. DTABSTM, AABSTM, ERBSTM, and FTBSTM do not build measurement structures in the measurement theory; therefore they do not comply with measurement theory. References [16, 17] have proved that monotonicity, acceleration, sensitivity, and substitutivity hold for DTABSTM

```
(1)  #include "stdio.h"
(2)  int Partition(int *R,int i,int j)
(3)  {
(4)    int pivot=R[i];
(5)    while(i<j)
(6)    {
(7)      while(i<j&&R[j]>=pivot)
(8)        j--;
(9)      if(i<j)
(10)       R[i++]=R[j];
(11)     while(i<j&&R[i]<=pivot)
(12)       i++;
(13)     if(i<j)
(14)       R[j- -]=R[i];
(15)   }
(16)   R[i]=pivot;
(17)   return i;
(18) }
(19) void QuickSort(int *R,int low,int high)
(20) {
(21)   int pivotpos;
(22)   if(low<high)
(23)   {
(24)     pivotpos=Partition(R,low,high);
(25)     QuickSort(R,low,pivotpos-1);
(26)     QuickSort(R,pivotpos+1,high);
(27) }
(28) }//QuickSort
(29) int main()
(30) {
(31)   int s[100];
(32)   printf("input 10 number");
(33)   for(int i=0;i<10;i++)
(34)   {
(35)     scanf("%d",&s[i]);
(36)   }
(37)   QuickSort(s,0,9);
(38)   for (int j=0;j<10;j++)
(39)   {
(40)     printf("%d ",s[j]);
(41)   }
(42) }
```

Algorithm 1: Quick sort program written in the C language.

Table 1: Comparative study in terms of measurement theory and the properties given in Section 4.

Property	T	DTABSTM	AABSTM	FTBSTM	ERBSTM
Measurement theory	√	×	×	×	×
Monotonicity	√	√	√	√	√
Acceleration	√	√	√	×	×
Sensitivity	√	√	√	√	√
Substitutivity	√	√	√	×	×

and AABSTM separately. ERBSTM and FTBSTM do not satisfy acceleration and substitutivity. The reason is that both of them do not consider the efficiency of using attributes and the quantitative relations between trustworthy attributes, while correlations between trustworthy attributes are the cause that trustworthy attributes can substitute each other to some extent. From Table 1, we can get that T is better than all the four measures in terms of measurement theory and the properties introduced in Section 4.

8. Conclusion and Future Work

In this paper, we apply measurement theory to measure the source codes trustworthiness, present an extensive structure based software trustworthiness measure T, and validate T with axiomatic approaches from a theory perspective. Compared with some popular software trustworthiness measures, the measure T can evaluate software trustworthiness better in terms of measurement theory and the properties described in Section 4.

However, only a small case is given to demonstrate the effectiveness of T, in order to support T's practical usefulness, we will experiment T using the real cases and develop a tool based on T. The specifications for the implementations expected by users of the program elements involved in the imperative languages that are given by us in [51] are not complete, and the specifications are only suitable for imperative languages. In the future, we will expand the specifications proposed in [51] and establish specifications for the implementations expected by users of the program elements involved in the object-oriented languages. We do not give methods for computing the values of ρ and β; how to determine the values of ρ and β is an important future work. The set of properties presented in Section 4 should be considered sets of necessary properties that need to be satisfied by a software trustworthiness measure. It is possible that some important properties are omitted due to oversight; extending and refining the set of properties are the future works too.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Disclosure

A preliminary version of this work was presented at the 2016 International Symposium on System and Software Reliability (ISSSR2016) [Research of Software Trustworthiness Measure Based on Validation [52]].

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is supported by the Innovation Group Project of National Natural Science Foundation (61321064), National Key R&D Plan Inter-Governmental R&T Cooperation Subproject (2016YFE0100600, 2016YFE0100300), National Natural Science Foundation of China (61370100), Doctoral Research Fund of Zhengzhou University of Light Industry (13501050045), and Science and Technology Project of Henan Province (182102210617).

References

[1] K. Liu, Z. Shan, J. Wang, J. He, Z. Zhang, and Y. Qin, "Overview On Major Research Plan of Trustworthy Software," *Bulletin of National Natural Science Foundation of China*, vol. 22, no. 3, pp. 145–151, 2008.

[2] S. Morasca, "Fundamental aspects of software measurement," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Preface*, vol. 7171, pp. 1–45, 2013.

[3] L. Briand, K. El Emam, and S. Morasca, "On the application of measurement theory in software engineering," *Empirical Software Engineering*, vol. 1, no. 1, pp. 61–88, 1996.

[4] H. Zuse, *A Framework for Software Measurement*, Walter de Gruyter, Berlin, Germany, 1998.

[5] A. Meneely, B. Smith, and L. Williams, "Validating software metrics: A spectrum of philosophies," *ACM Transactions on Software Engineering and Methodology*, vol. 21, no. 4, 2012.

[6] K. Srinivasan and T. Devi, "Software Metrics Validation Methodologies in Software Engineering," *International Journal of Software Engineering & Applications*, vol. 5, no. 6, pp. 87–102, 2014.

[7] D. Krantz, R. Luce, P. Suppes, and A. Tversky, "Foundations of Measurement," in *Additive and Polynomial Representations*, vol. 1, Academic Press, London, UK, 1971.

[8] H. Tao and Y. Chen, "A metric model for trustworthiness of softwares," in *Proceedings of the 2009 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology - Workshops, WI-IAT Workshops 2009*, pp. 69–72, September 2009.

[9] G. Poels and G. Dedene, "Distance-based software measurement: Necessary and sufficient properties for software measures," *Information and Software Technology*, vol. 42, no. 1, pp. 35–46, 2000.

[10] E. J. Weyuker, "Evaluating software complexity measures," *Institute of Electrical and Electronics Engineers. Transactions on Software Engineering*, vol. 14, no. 9, pp. 1357–1365, 1988.

[11] K. B. Lakshmanan, S. Jayaprakash, and P. K. Sinha, "Properties of Control-Flow Complexity Measures," *IEEE Transactions on Software Engineering*, vol. 17, no. 12, pp. 1289–1295, 1991.

[12] L. C. Briand, S. Morasca, and V. R. Basili, "Property-based software engineering measurement," *IEEE Transactions on Software Engineering*, vol. 22, no. 1, pp. 68–86, 1996.

[13] S. Morasca, "Refining the axiomatic definition of internal software attributes," in *Proceedings of the 2nd International Symposium on Empirical Software Engineering and Measurement, ESEM 2008*, pp. 188–197, Germany, October 2008.

[14] V. Del Bianco, L. Lavazza, S. Morasca, and D. Taibi, "A survey on open source software trustworthiness," *IEEE Software*, vol. 28, no. 5, pp. 67–75, 2011.

[15] H. Tao and Y. Chen, "A new metric model for trustworthiness of softwares," *Telecommunication Systems*, vol. 51, no. 2-3, pp. 95–105, 2012.

[16] H. Tao, Y. Chen, and J. Pang, "A software trustworthiness measure based on the decompositions of trustworthy attributes and its validation," *Lecture Notes in Electrical Engineering*, vol. 349, pp. 981–990, 2015.

[17] H. W. Tao, Y. X. Chen, and J. M. Pang, "Axiomatic approaches based on the software trustworthiness measure," in *Proceedings of the International Conference on Applied System Innovation, ICASI 2015*, pp. 135–142, Japan, May 2015.

[18] J. Wang, Y. Chen, B. Gu et al., "An approach to measuring and grading software trust for spacecraft software," *Scientia Sinica Techologica*, vol. 45, pp. 221–228, 2015.

[19] H. W. Tao and J. Zhao, "An improved attributes-based software trustworthiness metric model," *Journal of Wuhan University. Natural Science Edition. Wuhan Daxue Xuebao. Lixue Ban*, vol. 63, no. 2, pp. 151–157, 2017.

[20] Y. Li and Y. Chen, "A measurement model for trustworthy software based on trusted evidences," in *Proceedings of the 2nd International Symposium on System and Software Reliability, ISSSR 2016*, pp. 20–24, China, October 2016.

[21] Y. Zhang, Y. Zhang, and M. Hai, "An Evaluation Model of Software Trustworthiness Based on Fuzzy Comprehensive Evaluation Method," in *Proceedings of the 2012 International Conference on Industrial Control and Electronics Engineering (ICICEE)*, pp. 616–619, Xi'an, China, August 2012.

[22] S. Huiling, M. Jun, and Z. Fengyi, "A fuzzy comprehensive evaluation model for software dependability based on entropy weight," in *Proceedings of the International Conference on Computer Science and Software Engineering, CSSE 2008*, pp. 683–685, IEEE Computer Society Press, Washington, DC, USA, December 2008.

[23] B. Li and Y. Cao, "An improved comprehensive evaluation model of software dependability based on rough set theory," *Journal of Software* , vol. 4, no. 10, pp. 1152–1159, 2009.

[24] N. Fenton, B. Littlewood, M. Neil, L. Strigini, A. Sutcliffe, and D. Wright, "Assessing dependability of safety critical systems using diverse evidence," *IEE Proceedings Software*, vol. 145, no. 1, pp. 35–39, 1998.

[25] G. Si, Y. Ren, J. Xu, and J. Yang, "A dependability evaluation model for internetware based on Bayesian network," *Jisuanji Yanjiu yu Fazhan/Computer Research and Development*, vol. 49, no. 5, pp. 1028–1038, 2012.

[26] H.-Q. Liang and W. Wu, "Research of trust evaluation model based on dynamic Bayesian network," *Tongxin Xuebao/Journal on Communication*, vol. 34, no. 9, pp. 68–76, 2013.

[27] S. Yang, S. Ding, and W. Chu, "Trustworthy software evaluation using utility based evidence theory," *Jisuanji Yanjiu yu Fazhan/Computer Research and Development*, vol. 46, no. 7, pp. 1152–1159, 2009.

[28] S. Ding, S.-L. Yang, and C. Fu, "A novel evidential reasoning based method for software trustworthiness evaluation under the uncertain and unreliable environment," *Expert Systems with Applications*, vol. 39, no. 3, pp. 2700–2709, 2012.

[29] X. Li, X. Wang, T. Zhang, and J. Yi, "Software Trustworthness Evaluation Based on Weakness Analysis and Testing Assessment," *Journal of Tsinghua University (Science and Technology)*, vol. 51, no. 10, pp. 1287–1293, 2011.

[30] A. Immonen and M. Palviainen, "Trustworthiness evaluation and testing of open source components," in *Proceedings of the 7th International Conference on Quality Software, QSIC 2007*, pp. 316–321, USA, October 2007.

[31] D. Taibi, V. del Bianco, D. D. Carbonare, L. Lavazza, and S. Morasca, "Towards the evaluation of OSS trustworthiness: Lessons learned from the observation of relevant OSS projects," *International Federation for Information Processing*, vol. 275, pp. 389–395, 2008.

[32] V. del Bianco, L. Lavazza, S. Morasca, and D. Taibi, "Quality of Open Source Software: The QualiPSo Trustworthiness Model," in *Open Source Ecosystems: Diverse Communities Interacting*, vol. 299 of *IFIP Advances in Information and Communication Technology*, pp. 199–212, 2009.

[33] L. Lavazza, S. Morasca, D. Taibi, and D. Tosi, "Predicting OSS trustworthiness on the basis of elementary code assessment," in *Proceedings of the 4th International Symposium on Empirical Software Engineering and Measurement, ESEM 2010*, September 2010.

[34] Z. Zheng, S. Ma, W. Li et al., "Dynamical characteristics of software trustworthiness and their evolutionary complexity," *Science China Information Sciences*, vol. 52, no. 8, pp. 1328–1334, 2009.

[35] Z. Zheng, S. Ma, W. Li et al., "Complexity of software trustworthiness and its dynamical statistical analysis methods," *Science in China, Series F: Information Sciences*, vol. 52, no. 9, pp. 1651–1657, 2009.

[36] X. Zhang, W. Li, Z. Zheng, and B. Guo, "Optimized statistical analysis of software trustworthiness attributes," *Science China Information Sciences*, vol. 55, no. 11, pp. 2508–2520, 2012.

[37] D. Wang, Y. Lu, W. Zhao, and L. Fu, "Trust-measuring model for software using dependent relation between variables," *Huazhong Keji Daxue Xuebao (Ziran Kexue Ban)/Journal of Huazhong University of Science and Technology (Natural Science Edition)*, vol. 41, no. 1, pp. 41–45, 2013.

[38] D. Wang, H.-Y. Sun, J. Wang, and L.-H. Fu, "Trusted analysis model for interactive behavior of a software system based on slicing technology," *Beijing Gongye Daxue Xuebao/Journal of Beijing University of Technology*, vol. 39, no. 5, pp. 713–721, 2013.

[39] Y. Yuan and Q. Han, "A Software Behavior Trustworthiness Measurement Method based on Data Mining," *International Journal of Computational Intelligence Systems*, vol. 4, no. 5, pp. 817–825, 2011.

[40] Y. Yuan and Q. Han, "A data mining based measurement method for software trustworthiness," *Journal of Electronics*, vol. 21, no. 1, pp. 13–16, 2012.

[41] L. Zhuang, M. Cai, and C. X. Shen, "Hierarchical verification of behavior trustworthiness," *Journal of Beijing University of Technology. Beijing Gongye Daxue Xuebao*, vol. 38, no. 9, pp. 1396–1401, 2012.

[42] L. Zhuang, C. X. Shen, and M. Cai, "Research on state space reduction of behavior-based trusted dynamic measurement," *Chinese Journal of Computers. Jisuanji Xuebao*, vol. 37, no. 5, pp. 1071–1081, 2014.

[43] F. Zhang, M. Xu, and L. You, "A behavior modeling method based on system call and algebra process CCS," *Journal of Wuhan University*, vol. 56, no. 2, pp. 133–137, 2010.

[44] F. Zhang, M. Jiang, H. Wu, and M. Xu, "Approach for trust analysis of software dynamic behavior based on noninterference," *Computer Science*, vol. 39, no. 1, pp. 101–104, 2012.

[45] Q. Zhao, H.-Q. Wang, G.-S. Feng, and J. Zhao, "Measuring method of software dependability based on Pi calculus," *Jilin Daxue Xuebao (Gongxueban)/Journal of Jilin University (Engineering and Technology Edition)*, vol. 41, no. 6, pp. 1684–1689, 2011.

[46] N. N. Fu, X. Zhou, and T. Zhan, "QtPi: a calculus to enforce trustworthiness requirements," *Journal of Computer Research and Development*, vol. 48, no. 11, pp. 2120–2130, 2011.

[47] E. Amoroso, T. Nguyen, J. Weiss, J. Watson, P. Lapiska, and T. Starr, "Toward an approach to measuring software trust," in *Proceedings of the 1991 IEEE Computer Society Symposium on Research in Security and Privacy*, pp. 198–218, Oakland, CA, USA.

[48] E. Amoroso, C. Taylor, J. Watson, M. Marietta, and J. Weiss, "A process-oriented methodology for assessing and improving software trustworthiness," in *Proceedings of the 2nd ACM Conference on Computer and Communications Security, CCS 1994*, pp. 39–50, USA, November 1994.

[49] H. Zhang, F. Shu, Y. Yang, X. Wang, and Q. Wang, "A Fuzzy-Based Method for Evaluating the Trustworthiness of Software Processes," in *New Modeling Concepts for Today's Software Processes*, vol. 6195 of *Lecture Notes in Computer Science*, pp. 297–308, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[50] J. Du, Y. Yang, Q. Wang, and M. S. Li. Evidence-Based Trustworthy, "Software Process Assessment Method," *Journal of Frontiers of Computer Science and Technology*, vol. 5, no. 6, pp. 501–512, June 2011.

[51] H. Tao and Y. Chen, "A Quantitative Relation Model Between Trustworthy Attributes," in *Proceedings of the 3rd International Conference on Quantitative Logic and Soft Computing*, pp. 197–204, Xi'an, China, 2012.

[52] H. Tao and J. Zhao, "Research of software trustworthiness measurement based on validation," in *Proceedings of the 2nd International Symposium on System and Software Reliability, ISSSR 2016*, pp. 7–12, October 2016.