Hindawi
Mathematical Problems in Engineering
Volume 2022, Article ID 7290001, 15 pages
https://doi.org/10.1155/2022/7290001

Research Article

The Trustworthiness Measurement Model of Component Based on Defects

Yanfang Ma,[1,2] **Xiaotong Gao**,[2] **Wei Zhou**,[2] **and Liang Chen**[1,3]

[1]*School of Computer Science and Engineering, Changzhou Institute of Technology, Changzhou 213032, China*
[2]*School of Computer Science and Technology, Huaibei Normal University, Huaibei 235000, China*
[3]*School of Mathematics, Changzhou Institute of Technology, Changzhou 213032, China*

Correspondence should be addressed to Xiaotong Gao; gaoxt626@163.com

Received 7 July 2022; Revised 25 October 2022; Accepted 19 November 2022; Published 12 December 2022

Academic Editor: Weifeng Pan

In modern software engineering, the component-based development approach has become one of the important trends in software development technology. The trustworthiness of components plays a vital role in developing component-based trustworthy software. If there exist defects in components, then the trustworthiness of the component will be reduced, and the trustworthiness of the software system will be influenced. In this case, it is necessary to measure the trustworthiness of the component in terms of the defect. In this paper, a trustworthiness measurement model of components will be proposed based on defects. Firstly, the defect types are formalized according to the component specification. Secondly, the weight allocation method of defect types is designed based on the correlation between defect types and experts' evaluation. The value of the trustworthiness attribute is estimated by using the risk value of the defect and the weight of the defect type. Furthermore, the trustworthiness measurement model of the component is proposed, the corresponding algorithm is designed, and some algebra properties are proved. Finally, a case study is used to illustrate the application of the model.

1. Introduction

As software grows in complexity with many subsystems and components, measuring software quality in multiple dimensions is a challenging task [1]. The concept of trustworthiness appeared in software engineering in the 1990s, which was proposed and distinguished from reliability by Laprie [2]. As a comprehensive concept, software trustworthiness means that the software system operates according to the user's operational requirements [3].

The study of software trustworthiness mainly includes the evaluation and measurement of software trustworthiness, as well as the requirement analysis of trustworthy software, the assurance methods, techniques of development [4], and so on. As one of the core issues of software trustworthiness, software trustworthiness measurement is a quantitative evaluation of software trustworthiness, which can provide a quantitative evaluation basis for software design and development [5]. Software trustworthiness, as a complex and comprehensive concept, involves many quality attributes, including correctness, privacy, reliability, security, and so on. Stephen et al. proposed that software trustworthiness can be described by attributes [6], and some remarkable results about software trustworthiness have been achieved by scholars from attributes [7–10]. Software trustworthiness can also be measured based on trustworthy evidence. For example, these models in [11–13] effectively measured software trustworthiness through evidence. Based on the software life cycle process of service use, development, deployment, and operation process, Cerbo et al. collected the credibility evidence at different stages and proposed different development methodologies [14]. Tao et al. constructed the software trustworthiness measure from the source codes view based on the extensive structure in the measurement theory, presented four desirable properties of the measures from the standpoint of the module, and carried out theoretical validation for the developed measure by the use of axiomatic approaches [15]. Wang et al. developed a

trustworthiness evidence set to support the hierarchical evaluation, and a quantitative model of software development process trustworthiness was proposed for the first time, which can effectively support the objectivity and comprehensiveness of software process credibility evaluation [16]. Due to the increase in software size, the volume of software files and data has increased dramatically, which makes collecting positive data difficult. Therefore, the researchers started to collect and analyze the data from the perspective of defects to measure software trustworthiness laterally. For example, Li and Chen mapped the defect evidence to trustworthy attributes and proposed a trustworthiness measurement model by combining the complexity of program slicing [17]. Li et al. evaluated and predicted trustworthy attributes in software trustworthiness from the perspective of software defect prediction [18]. The authors of [19] established a trustworthy evidence specification of aerospace model software, which distinguished between critical, positive evidence, and negative evidence, and proposed a trustworthy measurement model and a trustworthy grading model for aerospace software based on source code trustworthy evidence. As an effective way to improve software productivity and quality, component-based software development has become one of the research hotspots in the field of software [20, 21]. Because the development method of component-based software is different from the classical software, which emphasizes the reuse of components, it is necessary to study the trustworthiness of component-based software [22]. There exist many research results for component-based software. For example, Wang et al. proposed a component-based trustworthiness measurement model based on component relationships. The logical relationships among components were analyzed, and the corresponding computational model was established to calculate the software trustworthiness through the trustworthy attributes of the components [23]. Wang et al. rationalized the trustworthiness of each component to optimize the software trustworthiness [24].

However, components are assembled to form component-based software, and the trustworthiness of individual component produces a vital impact on the trustworthiness of the software. Therefore, the trustworthiness of components should be researched to achieve the trustworthiness of the whole software. The research on component trustworthiness has become an important branch of trustworthiness research. For example, Dimriet al. proposed a path-based trustworthiness estimation method according to the usage coefficient of components [25]. Based on the performance specification, Wang and Chen gave two kinds of matching, Boolean matching and quantitative matching, and presented a performance-based component matching method to study component trustworthiness [26]. Meyer established five classifications called ABCDE, which can be used to evaluate the components [27]. In the reference [28], a trustworthiness model based on components is proposed, and by evaluating the associated trustworthiness model, the method can predict the trustworthiness of component-based software. Mohammad and Alagar proposed an architecture description language TADL for trustworthy component-based software systems, and a formal approach for the development of trustworthy component software systems has been presented in [29].

Considering the subjective factors of users, Zhou et al. combined the initial trustworthiness of components with the user feedback to assess the final trustworthiness of components [30].

However, as the size of the component increases, it becomes more and more difficult to collect trustworthy evidence. Therefore, many researchers have started to collect untrustworthy evidence to study component trustworthiness. The defect is an important part of untrustworthy evidence. For component-based software, if there exists a defect in the component, then the trustworthiness of the component will be decreased, and the trustworthiness of the software will be affected. So, it is key to research the trustworthiness of components based on the defect. In this paper, a formal description of the defect types of components will be proposed, and the trustworthiness measurement model of components will be established based on defects.

This paper is organized as follows: Section 2 introduces the fuzzy analytical hierarchy process (FAHP) method and the Pearson correlation coefficient (PPCs) method that will be used in this paper. Section 3 gives the formalization description of defect types. Section 4 presents the trustworthiness measurement model of components based on defects in detail. In Section 5, the computation of the model is illustrated with a case study. Section 6 shows the conclusion.

2. Preliminaries

In this section, we will review some existing methods that will be used in this paper.

2.1. FAHP Method. The weight often is used to describe the importance of some elements, and there exists the weight of some elements during some models [22, 24]. It is necessary to find a method to obtain the weight of elements in the model. The weight usually comes from the expert's experience and cognition. However, there exist some uncertainty and ambiguity in experts' evaluations. To solve this problem, the fuzzy analytical hierarchy process (FAHP) method is used to allocate weights. Because the FAHP method is more conducive to solving the fuzzy data that is difficult to quantify, it can reduce the data errors caused by the uncertainty and ambiguity in the expert's evaluation and make the evaluation more practical and accurate. The calculation process of the FAHP method is shown as follows, and for the detailed content, we can refer to [31].

Generally, the triangular fuzzy number is defined as $N = (l, m, u)$, where $l \le m \le u$. Let U be the domain. If $u_N(x)$: $U \longrightarrow [0, 1]$ exists, then $u_N(x)$ is the degree of affiliation of $x \in N$. The values of l, u determine the degree of fuzziness, and the larger $u - l$ is, the greater the degree of fuzziness.

$$u_N(x)\begin{cases} x-\dfrac{l}{m}-l, & x\in[l,m],\\ \dfrac{u-x}{u-x}, & x\in[m,u],\\ 0, & \text{other.}\end{cases} \tag{1}$$

The triangular fuzzy numbers $N_1(l_1,m_1,u_1), N_2\ (l_2,m_2,u_2)$ satisfy the following rules:

$$N_1+N_2=(l_1+l_2,m_1+m_2,u_1+u_2),$$
$$\frac{1}{N_1}=\left(\frac{1}{u_1},\frac{1}{m_1},\frac{1}{l_1}\right). \tag{2}$$

Suppose that there are n attributes. Let $h_{ij}=(l_{ij},m_{ij},u_{ij})$ be the relative importance of attribute i to attribute, where $h_{ji}=(h_{ij})^{-1},\ i\neq j,\ i,j=1,\ldots n$. Referring to the [1, 9] evaluation scale method proposed in the FAHP method [31], the evaluation standards of the triangular fuzzy numbers are given in Table 1. Experts compare the importance of trustworthy attributes two-by-two through their experience and knowledge and give the value of h_{ij}.

Step 1. Suppose that there are n elements that need to allocate the weight. Experts compare the importance of the n elements by using triangular fuzzy numbers and construct the fuzzy judgment matrix $H=(h_{ij})_{n\times n}$, where h_{ij} is the relative importance of ith element and jth element, $h_{ij}=(l_{ij},m_{ij},u_{ij})$, where l is the lower bound of experts' evaluation, u is the upper bound of experts' evaluation, and m is the evaluation coefficient favored by experts. $h_{ji}=(h_{ij})^{-1}=(1/u_{ij},1/m_{ij},1/l_{ij})$, $i\neq j,\ i,j=1,\ldots 9$. And the value of h_{ij} takes a value between 1–9 and it's reciprocal.

Step 2. Calculate the fuzzy subjective weight $\widetilde{w}_i=(\widetilde{w}_1,\ldots,\widetilde{w}_n)$ by fuzzy judgment matrix $H=(h_{ij})_{n\times n}$.

$$\widetilde{w}_i=\left(\widetilde{w}_i^l,\widetilde{w}_i^m,\widetilde{w}_i^u\right)=\left(\frac{\sum_{i=1}^n l_{ij}}{\sum_{j=1}^n\sum_{i=1}^n l_{ij}},\frac{\sum_{i=1}^n m_{ij}}{\sum_{j=1}^n\sum_{i=1}^n m_{ij}},\frac{\sum_{i=1}^n u_{ij}}{\sum_{j=1}^n\sum_{i=1}^n u_{ij}}\right). \tag{3}$$

Step 3. The subjective weight $w=(w_1^S,\ldots,w_n^S)$ is obtained by defuzzifying the fuzzy weight.

$$w_i^S=\frac{\widetilde{w}_i^l+4\widetilde{w}_i^m+\widetilde{w}_i^u}{6}, i=1,2,\ldots,n. \tag{4}$$

2.2. Correlation. There is a correlation among different elements, and it is necessary to master the correlation among these elements to assign weights reasonably. The Pearson correlation coefficient (PPCs) is used to describe the closeness of the relationship between vector quantities, and samples are used to infer the correlation of vector quantity in the domain. If the correlation of elements is higher and the conflict quantity is smaller, the degree of data redundancy is larger, and the amount of information is smaller, so the weight allocation should be reduced [32].

There are n vector quantities or elements which need to research for some domain, and the samples are obtained through testing m times, denoted as $x_i=(x_{i1},x_{i2},\ldots x_{in})$, where $i=1\ldots m$. The calculation process is described as follows, and for the detailed content, we can refer to [32].

Step 1. Calculate the standard deviation for each vector quantity. D_j is the standard deviation of the jth vector quantity, and $\overline{x}_j$ is the average of the jth vector quantity, where $j=1,2,\ldots,n$.

$$\begin{cases}\overline{x}_j=\dfrac{\sum_{i=1}^m x_{ij}}{m},\\ D_j=\sqrt{\dfrac{\sum_{i=1}^m\left(x_{ij}-\overline{x}_j\right)^2}{m}}.\end{cases} \tag{5}$$

Step 2. Calculate the correlation coefficient between vector quantities, and r_{ij} is the correlation coefficient between ith vector quantity and jth vector quantity.

$$r_{ij}=\frac{\sum_{k=1}^m\left(x_{ki}-\overline{x}_i\right)\left(x_{kj}-\overline{x}_j\right)}{\sqrt{D_iD_j}}, \tag{6}$$

where $i,j=1,2,\ldots,n$.

Step 3. The conflict quantity of vector quantity is characterized by the correlation coefficient, and R_j is the conflict quantity of jth vector quantity.

$$R_j=\sum_{i=1}^n\left(1-r_{ij}\right). \tag{7}$$

3. The Formal Model of Defect Type

To overcome the ambiguity of natural languages, formal descriptions have become an inevitable trend in software system development as the size of software systems and the complexity of the code increase. Reference [29] states that a formal description of the software can facilitate development and ditto for components. Component-based software engineering offers many advantages for software development, including reusability, ease of management, and reduced development time, effort, and cost. Software developers divide the software into multiple components, and different components perform different functions to achieve the users' requirements. They manage components separately to improve management efficiency. To improve the efficiency of component reuse, components are described formally so that developers can obtain detailed information about the components. The formal description of components accurately describes the essential characteristics of components, including the structure and behavior of components, standards, and so on. The authors in [33] presented the specification of the component as follows. For a detailed definition of component type, we can refer to [33].

Table 1: The number of defects.

Compare standards	Triangular fuzzy number	Countdown
Both attributes are equally important	(1, 1, 1)	(1, 1, 1)
Between equally important and slightly important	(1, 2, 3)	(1/3, 1/2, 1)
The former is slightly more important than the latter	(2, 3, 4)	(1/4, 1/3, 1/2)
Between slightly important and more important	(3, 4, 5)	(1/5, 1/4, 1/3)
The former is more important than the latter	(4, 5, 6)	(1/6, 1/5, 1/4)
Between more important and strongly important	(5, 6, 7)	(1/7, 1/6, 1/5)
The former is more strongly important than the latter	(6, 7, 8)	(1/8, 1/7, 1/6)
Between strongly important and extremely important	(7, 8, 9)	(1/9, 1/8, 1/7)
The former is extremely more important than the latter	(8, 9, 9)	(1/9, 1/9, 1/8)

3.1. Component Type

Definition 1 (Component type [33]). *A* component type is defined as a tuple $CT = (\Sigma, \Pi, \sigma, \Lambda, \xi, \Xi, \mathscr{A}, \mathscr{C}, \mathscr{T})$, and the definition of component types includes definitions of events, interface types, architecture types, contracts, attributes, and constraints, where

(1) Σ is a set of events. An event in Σ denotes either a stimulus, a request for service, or a response and service provision, but not both.

(2) Π is a set of interfaces. Interfaces are access points to the services provided and requested by components.

(3) σ: $\Pi \longrightarrow \mathbb{P}\Sigma$ is a function that associates events to interfaces such that $\forall \pi_1, \pi_2 \in \Pi, \sigma(\pi_1) \cap \sigma(\pi_2) = \varnothing$.

(4) Λ is a set of data parameters, which stores the different data parameters.

(5) ξ: $\Sigma \longrightarrow \mathbb{P}$: is a function that associates each service request or provision with a set of data parameters.

(6) Ξ is the contract set of components. It is defined in the interface of the component, and the contract includes restrictions on data, time, and so on.

(7) $\mathscr{A}$ is a set of attributes, such as maximum response time.

(8) $\mathscr{C}$ is a set of constraints for the component.

(9) $\mathscr{T}$ is a set of architecture types describing the possible internal structures of composite component types.

The formalization description shows detailed information about the component including the event, interfaced, attributes, and contract. This information comes from the user's requirements. If the developed component satisfies the specification, it satisfies the users' requirements and operates as expected, and to a certain extent, the component is considered trustworthy. But, if the developed component doesn't satisfy the specification, that means there exist some elements which are not consistent with the specification, such as event or interface type and so on. These inconsistent elements can lead to defects happened. When the component is tested, these inconsistent elements can be found. So, we can classify the defect type according to the specification.

3.2. The Formalization of Defect Types.

The definition of defect data has been given in many references, such as [17–19], and the defect data have been classified into different classes. But there is no unified standard to classify the defect data. Since the reason for the occurred defect is that the implement cannot satisfy the specification, so we can distinguish the defect from the specification view and produce unified criteria to differentiate the defect data. Because there are 9 elements during the specification of components, we can classify the defect type into 9 types. The defect types of the component are summarized, and some defect description information is given, as shown in Table 2.

Table 2 shows the description of every defect type. To distinguish different defect types, we use a defect identifier to indicate each defect type. However, it is not enough to describe the defect type. If there exists some defect, the trustworthiness of the component will be influenced. The impact can be decomposed into the impact on one or more trustworthy attributes [15]. Different defect types affect different trustworthy attributes, so it is necessary to list the attributes affected by the defect type in a set. The defect data are collected during testing, and valuable defect data are obtained by removing redundant data through data-processing methods, and these data are classified into each defect type and recorded in a set. Each piece of defect data causes different negative impacts on component trustworthiness, and this impact can be quantified with risk value. Subsequently, to quantify the degree of influence of each piece of defect data on the components, a risk value is assigned to each piece of defect data. To facilitate later calculations, we formalize the definition of defect type as follows.

Definition 2 (Defect type). For any defect type identifier $L \in \{\Sigma, \Pi, \sigma, \Lambda, \xi, \Xi, \mathscr{A}, \mathscr{C}, \mathscr{T}\}$, the defect type DT_L can be defined as a tuple $DT_L = (P_L, N_L, V_L, w_L)$, which includes the set of trustworthy attributes, the set of defect data, the set of risk values, and the weight of this defect type for L, such that

(1) P_L is the set of trustworthy attributes for L, which includes the trustworthy attributes affected by L. Let S_C stores all attributes related to the component's trustworthiness, and then, $P_L \subseteq S_C$, where C expresses the component.

(2) N_L is the set of all defect data for L, which stores all defect data whose defect type is L. If there is no defect data for L, then $N_L = \varnothing$.

(3) V_L is the set of all risk values for the defect data in N_L. For any $z \in N_L$, there exists a risk value $v_z \in \{1 \ldots 10\}$, and $V_L = \{v_z | z \in N_L\}$, and $|V_L| = |N_L|$. If the risk

Table 2: The description of defect types.

Defect identifier	The description of defect type
Σ	Irregular event definitions; input/output format error
Π	Incomplete definition of the interface function; chaotic language expression
σ	The same event corresponds to multiple interfaces for a single component
Λ	The number of data parameters is smaller than the number of services, and the services have no parameter association
ξ	The data parameter does not correspond to the exemplary service
Ξ	Unclear limitation range of time data; complex protocol extension
$\mathscr{A}_{\mathscr{C}}$	Unclear definition range of each attribute; incomplete attribute definition
$\mathscr{C}_{\mathscr{C}}$	The function that is not highly dependent; wrong component limitation
$\mathscr{T}$	Unreasonable structural design.

value of the defect data is larger, then the greater the negative impact of the occurrence of the defect on the component's trustworthiness.

(4) w_L is the weight of this defect type for L, which expresses the importance of this defect type during the trustworthiness of the component.

From this definition, we can see that when a defect happened, we can collect the valuable defect data and classify these data according to the defect type identifier, and the set of defect data for the defect type identifier can be obtained. Then, according to the experience of the specialist and developer, the set of trustworthy attributes and the set of risk values for the defect data can be achieved. Since there exists a different influence of the defect type on the trustworthiness of the component, so the weight of the defect type should be given. Next, we will use this information to estimate the trustworthiness of this component.

4. The Trustworthiness Measurement Model of Component

In the literature, there are a large number of different component definitions. However, only a few of them have been considered as component models in a taxonomy of software component models, such as SOFA 2.0, Fractal, and Koala [34–37]. These component models provide a wide variety of component definitions and contributions to the advancement of component-based development. Reference [29] provided an overview of these component models and analyzed their relative merits and then proposed a new model. Based on the component model proposed in [29], we will propose the trustworthiness measurement model of the component as follows.

If the developed component fully satisfies the specification of the component, then the component meets the user's needs and expectations at that point, and the component is considered trustworthy. However, in the development process, the component cannot be developed exactly according to the specification due to investment, time, and environmental constraints. Therefore, to a certain extent, the component does not fully satisfy the specification and has certain defects, so it is necessary to evaluate to what extent the component satisfies the specification and establish a trustworthiness measurement model. For the trustworthiness of components, it can be portrayed by values of multiple attributes or by analyzing defect data. There are many research results from the attributes to establish the trustworthiness model. However, the defect is a better indicator reflecting the trustworthiness of the component. In this section, a trustworthiness measurement model of components will be proposed based on defects.

4.1. The Combination Weight of Defect Type. Different defect types have different effects on the component's trustworthiness. To distinguish the influence degree of defect type on component trustworthiness, it is necessary to allocate weights to defect types.

Reasonable weight allocation will produce a direct influence on the component trustworthiness measurement model. Usually, the weight is given by the experts' subjective evaluation of defect types. But, from the specification of the component and the definition of defect type, we can see that there exists a correlation among different defect types, such as the interface type being related to the event type. Therefore, it is necessary to consider the correlation among the defect types when deciding the importance of the defect type.

Firstly, the fuzzy analytic hierarchy process (FAHP) method is used to decide the subjective weights of defect types. According to expert experience, the different expert provides their evaluation of the different defect types. Then, according to the steps that are introduced in Subsection 2.1, we can obtain the subjective weights of the 9 defect types, written as $w_i^S, i = 1, 2, \ldots, 9$.

Secondly, considering the interaction among different defect types, the correlation among defect types is calculated through the PCCs in Subsection 2.2. Suppose that there are m times tests for the component, then the data that can produce the defect can be obtained for every test. Then, we can classify these data into 9 different classes according to the defect types, denoted as $x_i = (x_{i1}, x_{i2}, \ldots x_{i9}), i = 1, 2, \ldots m$. According to the formulas (5)–(7), the conflict quantity of the defect type $R_j, j = 1, 2, \ldots, 9$ will be obtained.

Finally, the subjective weight and the correlation coefficient are combined to obtain the combined weight of the defect type, and the combined weight is calculated by performing weight normalization.

$$w_i = w_i^S \times \frac{R_i}{\sum_{i=1}^{9} w_i^S \times R_i}, i = 1, 2, \ldots 9. \tag{8}$$

During the combination weight, we can see that if the expert evaluation for some defect type is higher, then the

weight of this defect type is larger. And if the conflict quantity of the defect type is bigger, then there exists a high correlation to other defect types; therefore, the weight of this defect type is larger, which also shows that this defect type can have an important effect on the trustworthiness of the component.

4.2. The Trustworthiness Model of Component Based on Defects. The trustworthiness of components is an important criterion to guarantee the trustworthiness of component-based software. If the defect is found when the component is tested, then the trustworthiness of the component will be decreased. At the same time, some defects can lead to the related attribute value being reduced, which also produces the trustworthiness of the component declining. Therefore, in this section, we will try to propose the trustworthiness of components according to the defect and the attributes that are related to the defect.

4.2.1. The Measurement Model of Trustworthiness for Component. Firstly, we all know that defective data are not good for the software. If there is a defect in the software, then there exists an unexpected risk for the software, such as the software doesn't run and so on. So, for different defect types, we can use the defect value of the defect type to express the risk, the definition is shown as follows:

Definition 3 (Defect value of defect type). Suppose that L is an identity of defect type, and N_L is the set of all defect data for L. The defect value of L, denoted as $DVDT\{\_L\}$, is defined as the sum of the risk value of all defect data in N_L:

$$DVDT_L = \sum_{z \in N_L} v_z, \tag{9}$$

where $z \in N_L$ is a piece of defect data for L, and $v_z \in \{1, 2, \ldots 10\}$ is the risk value for z. If a defect has a high value of risk, the greater the negative impact on component trustworthiness when the defect occurs.

Secondly, from Definition 2, we know that every defect type is related to some attributes. In other words, some attributes can be affected by the different defect types. If the defect has a high-risk value, then it may lead to the software which cannot satisfy some attributes. The larger the defect value of the defect type is, the greater the impact on the trustworthy attributes. That means when a defect happened, there exists some risk to make the attribute of trustworthiness which cannot be satisfied. Therefore, we propose the definition of the defect value of trustworthiness attribute to express the risk.

Definition 4 (Defect value of trustworthy attribute). Suppose that there is a trustworthy attribute p, the set of all defect types that are associated with p is $L_p = \{L | p \in P_L\}$, and the defect value of the attribute p, written as $DVTA_p$, is defined as follows:

$$DVTA_p = \sum_{L \in L_p} w_L \times DVDT_L. \tag{10}$$

Definition 4 shows that there exist some risks that make the system which cannot satisfy the attribute. Because the defect data come from the multiple-time test, and the trustworthy attribute value is negatively correlated with its defect value, so the attribute value can be evaluated through the defect data and the risk value.

Definition 5 (Attribute value). Suppose that there is a trustworthy attribute $p \in S_C$, where S_C stores all attributes related to the component C. The attribute value of p, denoted as y_p, is calculated by the following formula:

$$y_p = e^{-DVTA_p/f}, \tag{11}$$

where $f \in R^+$ is the control parameter.

The value of f is usually dependent on the typedef structure number of code, the larger the size of the component, the more typedef structures of code, the greater the value of f, and the smaller f is otherwise. In terms of the impact of defects on attributes, the attribute value model should satisfy the following criteria:

Theorem 1

(1) $(y_p/\partial DVTA_{\mathrm{p}}) \leq 0$ *that means the defect value of a trustworthy attribute is larger, and the value of the trustworthy attribute is lower. That also shows the value of the trustworthy attribute which is negatively correlated with the defect value of the trustworthy attribute.*

(2) $0 \leq y_p \leq 1$ *that means the value range of the attribute value which is in the interval [0, 1].*

Proof of Theorem 1

(1) In fact, $(\partial y_p/\partial DVTA_{\mathrm{p}}) = (-DVTA_{\mathrm{p}}/f)\, e^{(-DVTA_{\mathrm{p}}/f)} \leq 0$

(2) Since $0 \leq e^{(-DVTA_{\mathrm{p}}/f)} \leq 1$, $0 \leq y_p \leq 1$ □

Definition 6 (The trustworthiness of component). Suppose that there is a component C, the set of all attributes related to C is denoted as S_C, and $p \in S_C$ is a trustworthy attribute that associates with the component. Let T_C be the trustworthiness of the component C. Then, T_C can be calculated by the following formula:

$$T_C = \prod_{p \in S_C} y_p^{\alpha_p}, \tag{12}$$

where α_p is the weight of the trustworthy attribute p, which expresses the importance of the attribute during the trustworthiness of the component.

In [10], the author showed that the trustworthiness of software should satisfy some criteria. Since the component can be treated as an independent unit to some extent, which

can finish some special functions, it is necessary to prove the trustworthiness of the component which also satisfies these criteria.

Theorem 2. *Suppose that T_C represents the trustworthiness of the component C. Then, T_C satisfies the following criteria:*

(1) Nonnegativity: nonnegativity means that component trustworthiness is nonnegative, $T_C \geq 0$.

(2) Proportionality: proportionality refers to each value of the attribute which should have an appropriate proportion, which means $\exists c_1, c_2 \in R^+$ such that $c_1 \leq y_p/y_q \leq c_2, p, q \in S$.

(3) Monotonicity: it refers to the monotonous increase of components with respect to the value of the attribute, $\partial T_C/\partial y_p \geq 0$.

(4) Acceleration: the value of an attribute contributes less and less to the component trustworthiness when the value of the attribute is only increased, and the values of other attributes are kept unchanged, $\partial^2 T_C/\partial y_p^2 \leq 0$.

(5) Sensitivity: sensitivity describes the impact of attributes on component trustworthiness. The higher the sensitivity value is, the greater the impact is, $\partial T_C/\partial y_p \cdot y_p/T_C \geq 0$.

(6) Substitutability: substitutability refers to an attribute that can be replaced by other attributes to a certain extent, which means $\exists c_1, c_2 \in R^+$ such that $c_1 \leq \sigma_{pq} \leq c_2$, where σ_{pq} is the substitution difficulty between attribute p and attribute q.

$$\sigma_{pq} = \frac{d\left(y_p/y_q\right)}{d\left(-\left(\left(\partial T_C/\partial y_q\right)/\left(\partial T_{C_i}/\partial y_p\right)\right)\right)} \times \frac{-\left(\left(\partial T_C/\partial y_q\right)/\left(\partial T_C/\partial y_p\right)\right)}{y_p/y_q}. \tag{13}$$

(7) Stability: stability means that if all the values of attributes meet the users' expectations, the component trustworthiness also meets users' expectations, $y_0 \leq T_C \leq \max(y_p)$

Proof of Theorem 2. According to the verification method in [10] and the definition of T_C in this paper, it is easy to get the proof. □

4.2.2. The Algorithm of Trustworthiness Measurement Model of Component. To implement the automated calculation of the measurement model, it is indispensable to design the algorithm of the model. From the above discussion, we know that the trustworthiness of a component can be achieved by trustworthy attributes.

According to formula (12), the values of trustworthy attributes can be obtained by analyzing defect data. Next, we will try to design the algorithm to implement the automated computation of the trustworthiness of the component. Given the component C, we define the set of all trustworthy attributes related to C, S_C, and the importance weight of every attribute α_p. After the software is tested, the defect data are collected. According to the defect type, the defect data are classified into different classes $N_L, L \in \Sigma, \Pi, \sigma, \Lambda, \xi, \Xi, \mathscr{A}, \mathscr{C}, \mathscr{T}$, and the weight of every defect type can be achieved based on the formula (8). Because every defect data can affect some attributes, the set of attributes related to the defect type is collected. And, every defect data exist the risk for the software, so the set of the risk value of every defect data is given. Furthermore, the control parameter is defined. Then, the calculation processes are shown as follows:

(1) Based on Definition 3 and formula (9), the defect value of every defect type is calculated by the risk value of every defect data in different defect types

(2) According to the classification of defect data and the set of attributes related defect types, the set of defect types that are associated with every attribute is established

(3) According to the weight of every defect type, the defect value of every attribute is determined based on the formula (10)

(4) Based on the control parameter, the value of the trustworthy attribute is calculated according to the defect value of the trustworthy attribute and formula (11)

(5) The trustworthiness of component C can be obtained by the formula (12)

To design the algorithm of trustworthiness of component in detail, some declarations should be given as follows:

(i) S_C: the set of all attributes related to the component C

(ii) P_L: the set of attributes related to the defect type identifier $L \in \{\Sigma, \Pi, \sigma, \Lambda, \xi, \Xi, \mathscr{A}, \mathscr{C}, \mathscr{T}\}$

(iii) N_L: the set of defect data for $L \in \{\Sigma, \Pi, \sigma, \Lambda, \xi, \Xi, \mathscr{A}, \mathscr{C}, \mathscr{T}\}$

(iv) V_L: the set of risk values for $L \in \{\Sigma, \Pi, \sigma, \Lambda, \xi, \Xi, \mathscr{A}, \mathscr{C}, \mathscr{T}\}$

(v) f: the control parameter

(vi) α_p: the weight of attribute for every p, $p \in S_C$ is a trustworthy attribute that associates with the component

(vii) w_L: the weight of defect type for $L \in \{\Sigma, \Pi, \sigma, \Lambda, \xi, \Xi, \mathscr{A}, \mathscr{C}, \mathscr{T}\}$, which can be calculated by formula (8)

(viii) $DVDT_L$: the defect value of the defect type L

(ix) $DVTA_p$: the defect value of the trustworthy attribute p

(x) T_C: the trustworthiness of the component C

(xi) y_p: the attribute value of the trustworthy attribute p

(xii) L_p: the set of defect types that are associated with p

Based on the above calculation process, the algorithm is designed as Algorithm 1, written by the pseudo-code:

During this algorithm, S_C, P_L, N_L, V_L, w_L, f, and α_p are the input data, and the target data are T_C. To obtain T_C, some intermediate variables are necessary, such as $DVTA_p$, L_p, $DVDT_L$. In the beginning, these variables should be initialized. Since T_C is calculated by the iteration of itself, so the initial value is defined as $T_C = 1$. For every defect type $L \in \{\Sigma, \Pi, \sigma, \Lambda, \xi, \Xi, \mathscr{A}, \mathscr{C}, \mathscr{T}\}$, the initial value of $DVDT_L$ is 0. For every attribute $p \in S_C$, $DVTA_p$ and L_p are initialized as $DVTA_p = 0, L_p = \varnothing$. The algorithm includes four subroutines: the first subroutine between line 8 and 12 shows the defect value of the defect type; the second subroutine from line 13 to 19 finds the set of defect types that are associated with the attribute; the third subroutine between line 20 and 25 computes the value of the attribute; the last subroutine from line 26 to 28 gives the trustworthiness of the component.

4.2.3. Time Complexity. There is a loop from line 2 to 4 and from line 5 to 6, respectively. Suppose that there are n attributes considered for the component. Then, the number of executions for the first loop is n. For the loop between line 5 and line 7, Table 2 shows that there are 9 defect type identifiers, so the number of executions is 9. And from line 8 to line 12, there is a loop nesting, the number of executions for the external loop is 9, and the number of executions for the internal loop depends on the amount of defect data. Generally, the amount of defect data is finite, denoted as m. Since the computation of line 10 is the basic operation that needed constant time, so the number of executions for line 8 to line 12 is $9m$. There is also a loop nesting between line 13 and line 19. Since the maximal cardinality of the set S_C is n, so the number of executions for the loop nesting is $9n$. Similarly, the number of executions from line 20 to line 25 is also $9n$. It is easy to know the number of executions for the last loop is n. Generally, the number of defect data m is larger than the number of attributes n. Therefore, the time complexity of this algorithm is polynomial time.

To clarify the detailed process of the algorithm, the flowchart can be shown in Figure 1.

The authors in [17, 19] also proposed the model of the trustworthiness of software from the evidence. They divided the evidence into positive and negative evidence. The positive evidence cannot lead to the failure of software, which plays a role in protecting the system to some extent. The number of positive evidence is larger, and the trustworthiness of the software is higher. But, the number of negative evidence is larger, and the trustworthiness of the software is lower. The trustworthiness of software based on the positive evidence was calculated by the product of attributes. The trustworthiness of software based on the negative evidence was calculated by a monotone decreasing function. The final trustworthiness of the software was obtained by combining the two trustworthiness metrics. The differences between [17] and [19] were the method of classifying the evidences and the final combining model. During their models, they only showed the model from the mathematical perspective and didn't give a detailed algorithm design. Analyzing their models, the complexity of trustworthiness based on the positive evidence is polynomial time by [38]. The efficiency of calculating trustworthiness from the negative evidence is dependent on the number of negative evidence.

In our model, we only focus on the defect data which can produce the failure of the component, i.e., the negative evidence. The main reason is to characterize the user's strict requirement specification for trustworthiness. Especially, for some vital systems, the failure of a software system will produce a great loss for human lives and the economy. Therefore, defect data are not allowed during these systems. If there exists some defects in the system, then the trustworthiness of the software is lower, which will lead to the system that cannot satisfy the requirement. Through the above analysis, we know that the efficiency of our model is also polynomial time, depending on the number of defect data.

Therefore, if we only consider the amount of defect data, then the complexity in our model is the same as [17, 19]. But, in [17, 19], they have to spend some time calculating the trustworthiness based on the positive evidence. Therefore, our model is prior to the method in [19] to some extent.

Though the efficiency of our method is the same as [17, 19] from the perspective of defect, our method not only details the defect data according to the specification and considers the risk value of defect type but also the relationship between defect data and attribute is researched, which can pay more attention to the detail of defect data and help the developer and tester to find the reason defect happened. To state the reasonability of our method, a case will be shown in the next section.

5. Case Study

In [19], the authors used the data from the NASA's open-source Core Flight Explosive code (https://github.com/nasa/cFE) on GitHub. To explain our method, we also use the same data to simulate our model. In this section, the "msg" module of NASA's code will be used as the measurement object to verify the effectiveness and practicability of the trustworthiness measurement method based on the defect proposed in this paper. The code line of this module is 1256 lines. Suppose that the trustworthiness of the module is affected by seven essential trustworthy attributes: security, correctness, reliability, privacy, antirisk, availability, and performance. Next, the calculation process of this module's trustworthiness will be introduced in detail.

5.1. The Weight of Defect Type

5.1.1. Step 1. Calculate Subjective Weight. In order to obtain the importance of defect type, it is necessary to invite some software test experts to evaluate the significance of the above defect types and give a fuzzy judgment matrix. The fuzzy weight of each defect type is obtained by calculating the fuzzy matrix in the following. Defuzzify the data in Table 3 to obtain subjective weight, as shown in Table 4. The detailed process of calculation is based on the following formulas (3)–(4):

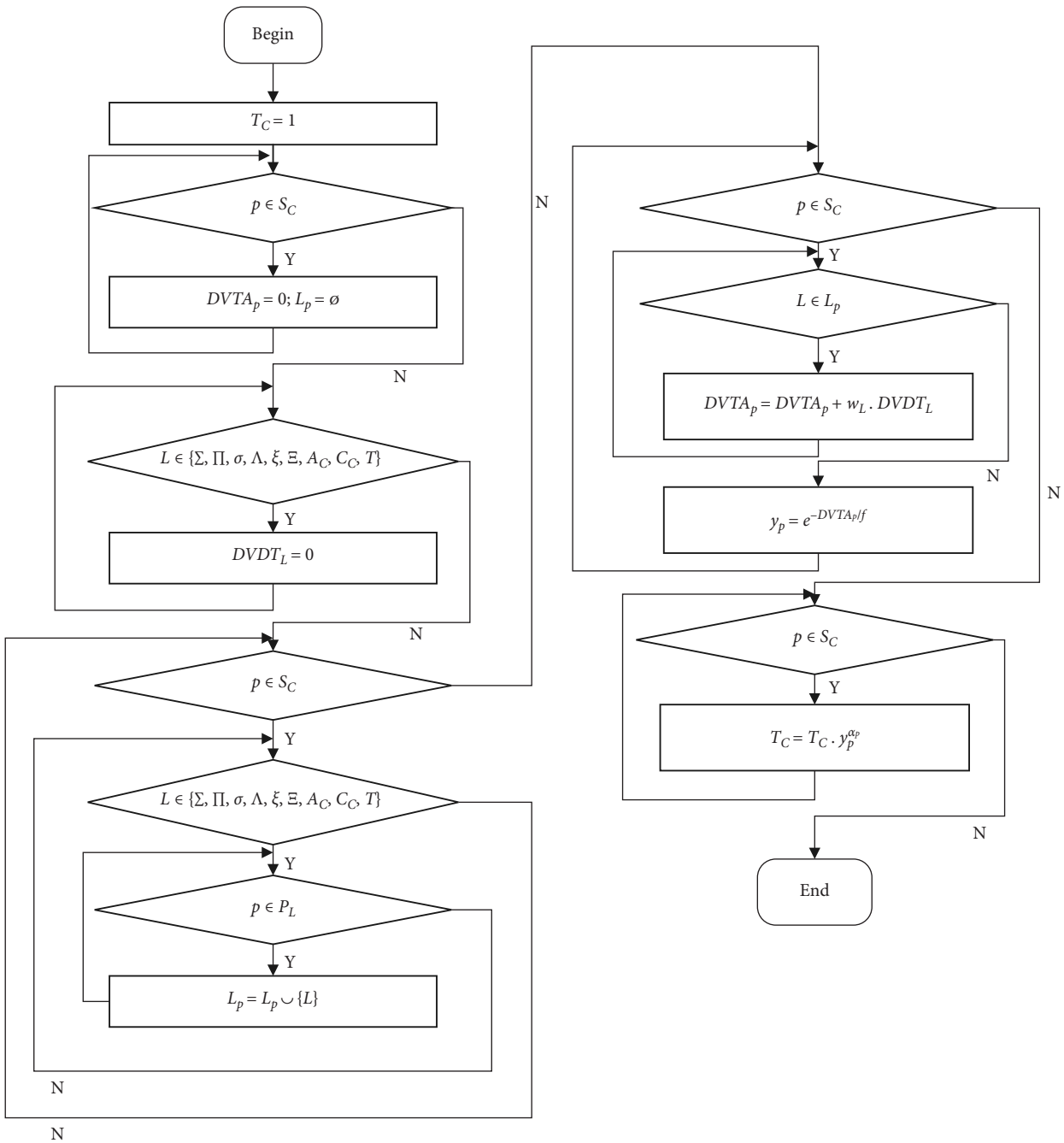

Figure 1: The flowchart of trustworthiness measurement model.

Input: S_C, P_L, N_L, V_L, w_L for every $L \in \{\Sigma, \Pi, \sigma, \Lambda, \xi, \Xi, \mathscr{A}, \mathscr{C}, \mathscr{T}\}$, f, $\alpha_p for p \in S_C$.
Output: T_C.
(1) **Initialize:** $T_C = 1$
(2) **for** $p \in S_C$ **do**
(3) $DVTA_p = 0, L_p = \varnothing$
(4) **end for**
(5) **for** $L \in \{\Sigma, \Pi, \sigma, \Lambda, \xi, \Xi, \mathscr{A}, \mathscr{C}, \mathscr{T}\}$**do**
(6) $DVDT_L = 0$
(7) **end for**
(8) **for** $L \in \{\Sigma, \Pi, \sigma, \Lambda, \xi, \Xi, \mathscr{A}, \mathscr{C}, \mathscr{T}\}$**do**
(9) **for** $z \in N_L$**do**
(10) $DVDT_L = DVDT_L + \sum_{v_z \in V_L} v_z$
(11) **end for**
(12) **end for**
(13) **for** $p \in S_C$**do**
(14) **for** $L \in \{\Sigma, \Pi, \sigma, \Lambda, \xi, \Xi, \mathscr{A}, \mathscr{C}, \mathscr{T}\}$**do**
(15) **if** $p \in P_L$**then**
(16) $L_p = L_p \cup \{L\}$
(17) **end if**
(18) **end for**
(19) **end for**
(20) **for** $p \in S_C$**do**
(21) **for** $L \in L_p$**do**
(22) $DVTA_p = DVTA_p + w_L \cdot DVDT_L$
(23) **end for**
(24) $y_p = e^{(-DVTA_p/f)}$
(25) **end for**
(26) **for** $p \in S_C$**do**
(27) $T_C = T_C \cdot y_p^{\alpha_p}$
(28) **end for**
(29) **return** T_C

Algorithm 1: Calculation of component trustworthiness.

$$\begin{bmatrix}
(1,1,1) & \left(\frac{1}{5},\frac{1}{3},\frac{1}{2}\right) & (2,3,4,) & (2,3,3,) & (3,3,4) & \left(\frac{1}{4},\frac{1}{3},\frac{1}{2}\right) & (2,3,3) & (3,3,4) & (2,3,5) \\
(2,3,5) & (1,1,1) & (3,3,4) & (3,4,4) & (3,4,4) & (1,1,2) & (4,4,5) & (3,4,5) & (2,4,4) \\
\left(\frac{1}{4},\frac{1}{3},\frac{1}{2}\right) & \left(\frac{1}{4},\frac{1}{3},\frac{1}{3}\right) & (1,1,1) & (1,1,2) & (1,2,2) & \left(\frac{1}{5},\frac{1}{4},\frac{1}{4}\right) & (1,1,2) & \left(\frac{1}{2},\frac{1}{2},\frac{1}{3}\right) & \left(\frac{1}{2},\frac{1}{2},1\right) \\
\left(\frac{1}{3},\frac{1}{3},\frac{1}{2}\right) & \left(\frac{1}{4},\frac{1}{4},\frac{1}{3}\right) & \left(\frac{1}{2},1,1\right) & (1,1,1) & (1,1,2) & \left(\frac{1}{6},\frac{1}{5},\frac{1}{4}\right) & (1,1,2) & \left(\frac{1}{2},1,1\right) & \left(\frac{1}{2},\frac{1}{2},1\right) \\
\left(\frac{1}{4},\frac{1}{3},\frac{1}{3}\right) & \left(\frac{1}{4},\frac{1}{4},\frac{1}{3}\right) & \left(\frac{1}{2},\frac{1}{2},1\right) & \left(\frac{1}{2},1,1\right) & (1,1,1) & \left(\frac{1}{6},\frac{1}{5},\frac{1}{5}\right) & (1,1,2) & \left(\frac{1}{3},\frac{1}{2},1\right) & (1,1,2) \\
(2,3,4) & \left(\frac{1}{2},1,1\right) & (4,4,5) & (4,5,6) & (5,5,6) & (1,1,1) & (5,5,6) & (4,4,6) & (4,4,6) \\
\left(\frac{1}{3},\frac{1}{3},\frac{1}{2}\right) & \left(\frac{1}{5},\frac{1}{4},\frac{1}{4}\right) & \left(\frac{1}{2},1,1\right) & \left(\frac{1}{2},\frac{1}{2},1\right) & \left(\frac{1}{2},1,1\right) & \left(\frac{1}{6},\frac{1}{6},\frac{1}{5}\right) & (1,1,1) & \left(\frac{1}{3},\frac{1}{2},\frac{1}{2}\right) & \left(\frac{1}{2},\frac{1}{2},1\right) \\
\left(\frac{1}{4},\frac{1}{3},\frac{1}{3}\right) & \left(\frac{1}{5},\frac{1}{4},\frac{1}{3}\right) & (1,2,2) & (1,1,2) & (1,2,3) & \left(\frac{1}{6},\frac{1}{4},\frac{1}{4}\right) & (2,2,3) & (1,1,1) & (1,2,3) \\
\left(\frac{1}{5},\frac{1}{3},\frac{1}{2}\right) & \left(\frac{1}{4},\frac{1}{4},\frac{1}{2}\right) & (1,2,2) & (1,2,2) & \left(\frac{1}{2},1,1\right) & \left(\frac{1}{6},\frac{1}{6},\frac{1}{4}\right) & (1,2,2) & \left(\frac{1}{3},\frac{1}{2},1\right) & (1,1,1)
\end{bmatrix}. \tag{14}$$

Table 3: Fuzzy weight of each defect type.

Defect identifier	Fuzzy weight
Σ	(0.155, 0.154, 0.154)
Π	(0.220, 0.219, 0.209)
σ	(0.057, 0.054, 0.062)
Λ	(0.053, 0.057, 0.063)
ξ	(0.050, 0.045, 0.067)
Ξ	(0.040, 0.041, 0.039)
$\mathscr{A}$	(0.295, 0.273, 0.252)
$\mathscr{C}$	(0.076, 0.084, 0.092)
$\mathscr{T}$	(0.055, 0.072, 0.063)

5.1.2. Step 2. Calculate the Conflict Quantity. The msg module is tested five times, collecting test data, and assigning risk value to defect data, and the numbers of defect data are recorded and classified into the defect type identifier. Subsequently, the correlation coefficient is calculated according to the defect data in Table 5. Then, the conflict quantity of defect types is obtained in Table 6 by correlation coefficient according to the following formula (7).

5.1.3. Step 3. Obtain the Combination of Weight. The subjective weight and the conflict quantity are combined by formula (8). The results are shown in Table 7.

5.2. Analysis of Defect Data. The defect data during every test are collected and classified into 9 defect types. Then, the trustworthiness of the module is measured by the defect data. For every test, the trustworthiness of the module can be obtained. Finally, the average of five metrics is taken as the final trustworthiness. Next, we will take the data of test 1 as an example, the number of defect data is 20, the same as the defect data in [19], and the calculation process is detailed. The defect data are classified according to component specifications. For the same defect type, the data with the same risk value are recorded in Table 8. For example, there are 5 defect data with the defect type Σ, and there are two data on the same risk value 3. The corresponding trustworthy attributes for every defect type identifier are also listed. The information can be concluded as follows.

5.3. Calculate the Trustworthiness of Component

5.3.1. Step 1. Calculate the Defect Value of the Defect Type. Calculate $DVDT_L$ for each defect type L. Taking the defect type with the identifier Σ as an example, $DVDT_\Sigma$ is equal to the sum of the product of the defect risk values and numbers of defects according to formula (9), $DVDT_\Sigma = 3 \times 2 + 1 \times 3 = 9$. Operations are repeated to calculate the risk value of each defect type as shown in Table 9.

5.3.2. Step 2. Calculate the Defect Value of the Attribute. The calculation process is illustrated by taking "correctness" as an example. Correctness is affected by the two defect types: σ and ξ. The defect value of correctness is determined by the $DVDT_\sigma$, $DVDT_\xi$ and the weight of the corresponding defect type. By formula (10), $DVTA_{\text{correctness}} = 4 \times 0.06 + 4 \times 0.05 = 0.44$. By using the same calculation, the defect values of the other trustworthy attributes are shown in Table 10.

Table 4: Subjective weight of each defect type.

Defect identifier	Subjective weight
Σ	0.15
Π	0.22
σ	0.06
Λ	0.06
ξ	0.05
Ξ	0.27
$\mathscr{A}$	0.04
$\mathscr{C}$	0.08
$\mathscr{T}$	0.07

Table 5: The number of defects.

Test	Σ	Π	σ	Λ	ξ	Ξ	$\mathscr{A}$	$\mathscr{C}$	$\mathscr{T}$
1	5	4	2	3	2	0	1	1	2
2	5	4	2	3	2	0	1	2	2
3	4	4	2	3	2	0	0	1	3
4	5	4	1	2	3	1	0	1	2
5	4	5	2	3	2	0	1	0	2

Table 6: Conflict quantity of each defect type.

Defect identifier	Conflict quantity
Σ	0.10
Π	0.11
σ	0.10
Λ	0.10
ξ	0.13
Ξ	0.13
$\mathscr{A}$	0.09
$\mathscr{C}$	0.11
$\mathscr{T}$	0.13

Table 7: Combination weight of each defect type.

Defect identifier	Combination weight
Σ	0.16
Π	0.23
σ	0.06
Λ	0.06
ξ	0.05
Ξ	0.29
$\mathscr{A}$	0.06
$\mathscr{C}$	0.02
$\mathscr{T}$	0.07

5.3.3. Step 3. Calculate the Values of Trustworthy Attributes. Through analyzing the module code, we know that there are 26 typedef structure definitions during the code. Thus, we define the control parameter $f = 26$. The value of correctness is calculated according to formula (11), i.e.,

$$y_{\text{correctness}} = e^{\left(-DVTA_{\text{correctness}}/f\right)} = 0.983. \tag{15}$$

The above calculations are performed to obtain the values of the other trustworthy attributes. And the weights of the trustworthy attributes can be allocated according to the FAHP method, refer to Table 11 for the specific contents.

Table 8: The defect data information.

Defect identifier (L)	Corresponding trustworthy attribute (P)	Risk value (C)	The number of defects (N)	Weight (w)
Σ	Performance, availability	3	2	0.16
		1	3	
Π	Security, privacy	4	1	0.23
		3	1	
		1	2	
σ	Availability, reliability, correctness	2	2	0.06
Λ	Reliability	3	1	0.06
		1	2	
ξ	Correctness, security	2	2	0.05
Ξ	Reliability, privacy, antirisk	0	0	0.29
$\mathscr{A}$	Performance, availability	2	1	0.06
$\mathscr{C}$	Antirisk	3	1	0.02
$\mathscr{T}$	Performance, reliability	1	2	0.07

5.3.4. Step 4. Calculate the Trustworthiness of Component "msg". Let T^1_{msg} be the trustworthiness of the component based on the first test. The trustworthiness of component "msg" can be calculated through the values of trustworthy attributes, i.e.,

$$T^1_{msg} = 0.916^{0.21} \times 0.983^{0.17} \times 0.974^{0.14} \times 0.926^{0.08} \times 0.997^{0.10} \times 0.933^{0.10} \times 0.937^{0.14} = 0.954. \quad (16)$$

Similarly, other test results are analyzed and calculated to obtain credibility, as shown in Table 12. Five tests are subsequently calculated and averaged to obtain a final trustworthiness $T_{msg} = 0.955$.

5.4. Comparison. In [19], the authors presented a method to measure the trustworthiness of software by analyzing the source code-oriented aerospace software, and every defect data can be treated as negative evidence. And, the measurement model is established based on the positive evidence and defect data. In our model, the trustworthiness of the component is measured based on the defect data, and it is stricter to evaluate the trustworthiness. It is more suitable to verify the trustworthiness of some vital components. But during the real practice, we notice that there are some defective data that can lead to the same error. These defect data should be considered as a piece of evidence instead of several pieces of evidence. At the same time, the reasons these defect data occurred are different, so there are different types of defect data. These different types of defect data can produce different influences on trustworthiness. Therefore, our method is different from the method in [19]. Our method considered the important degree of defect type during defect data. Though our method is focused on the component, if we treated software as the biggest component, then our method is also suitable for the software system.

Table 9: The defect value of each defect type.

Defect identifier	$DVDT_L$
Σ	9
Π	9
σ	4
Λ	5
ξ	4
Ξ	0
$\mathscr{A}$	2
$\mathscr{C}$	3
$\mathscr{T}$	2

Table 10: The defect value of each trustworthy attribute.

Attribute	$DVTA$
Security	2.27
Correctness	0.44
Reliability	0.68
Privacy	2.07
Antirisk	0.06
Availability	1.8
Performance	1.7

Figure 2 shows the different results of our model, and the model in [19] used on the same collected defect data. The orange block expresses our result, and the blue block is the result in [19]. The comparison shows that the component trustworthiness calculated with our model is slightly lower than that calculated with the trustworthiness measurement model based on source code in [19]. The main reason is that the model in [19] included the trustworthiness metric based on positive evidence, which can lead to the trustworthiness being higher.

Table 11: The defect value of each trustworthy attribute.

Attribute	Security	Correctness	Reliability	Privacy	Antirisk	Availability	Performance
Value	0.916	0.983	0.974	0.926	0.997	0.933	0.937
Weight	0.21	0.17	0.14	0.08	0.10	0.10	0.14

Table 12: The trustworthiness of tests.

Test	Trustworthiness
Test 1	0.954
Test 2	0.952
Test 3	0.955
Test 4	0.957
Test 5	0.957

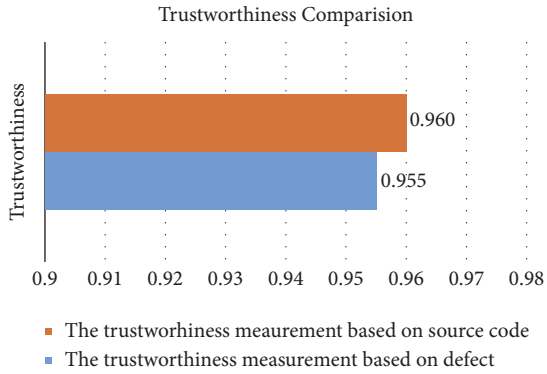

Figure 2: Trustworthiness comparison.

6. Conclusions

Component is an important part during the complexity system. The trustworthiness of component can produce the great influence for the quality of the system. Therefore, it is necessary to measure the trustworthiness of the component accurately. The defect data can reflect the trustworthiness of the component straightly. Therefore, we established the trustworthiness model based on the defect data. To describe the reason that the defect data occurred, the formal characterization of the defect type is established based on the component specification. And, the relationship between defect data and the attributes is established. The value of the attribute can be obtained by using the defect data. Finally, the trustworthiness model of the component is proposed based on the attributes and weight of attributes. The corresponding algorithm is designed to realize the automated calculation.

The model proposed in this paper not only considers the defect's effect on trustworthiness but also classifies the defect type based on the formal specification of the defect type. The complexity of the algorithm for calculating the trustworthiness of components is polynomial time, which is easy to implement automation computations. Compared with other related research, the contributions of this paper are shown as follows:

(1) From the formalization view, the defect data are graded to differentiate from different defect data

(2) The weight allocation method of defect types is innovated to express the importance of different types of defects

(3) The relationship between the defect data and the component trustworthiness attributes is established

(4) The component trustworthiness based on the defect data is measured by combining the value of attributes with the weight of attributes

The model proposed in this paper can effectively measure the trustworthiness of components from the perspective of defects, but the classification of defect data in our model is usually based on manual work. Though there are many tools to find the defect data, it is the first time to classify the defect data from the specification. On the other hand, some complex systems are developed by the component-based development process. There are much different connection structures for the component. But, our model only focuses on the components and does not consider the combination structure among components. Therefore, in future work, we will try to find an automated classified tool in order to expand the scope of our model usage. At the same time, to measure the trustworthiness of the whole component-based software from the perspective of defects, we will try to construct the trustworthiness measurement model from the different combinations of components and the interface connection.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was supported by the Natural Science Foundation of Anhui Province (No. 2108085MF204), the National Natural Science Foundation of China (Nos. 62162014 and 62077029), and the Abroad Visiting of Excellent Young Talents of Universities in Anhui Province (No. GXGWFX2019022).

References

[1] J. H. Cho, S. Xu, P. M. Hurley, M. Mackay, T. Benjamin, and M. Beaumont, "Matthew mackay, trevor benjamin, and mark beaumont. STRAM: measuring the trustworthiness of computer-based systems," *ACM Computing Surveys*, vol. 51, no. 6, pp. 1–47, 2019.

[2] J. C. Laprie, "Dependable computing and fault tolerance: concepts and terminology," in *Proceedings of the 25th International Symposium on Fault-Tolerant Computing*, pp. 2–11, IEEE, Pasadena, CA, USA, June 1995.

[3] N. G. Mohammadi and M. Heisel, "Patterns for identification of trust concerns and specification of trustworthiness requirements," in *Proceedings of the 21st European Conference on Pattern Languages of Programs*, pp. 1–20, ACM, Bavarian Kloster Irsee, Germany, July 2016.

[4] F. Brosig, P. Meier, S. Becker, A. Koziolek, H. Koziolek, and S. Kounev, "Quantitative evaluation of model-driven performance analysis and simulation of component-based architectures," *IEEE Transactions on Software Engineering*, vol. 41, no. 2, pp. 157–175, 2015.

[5] J. He, Z. Shan, J. Wang et al., "Review of the achievements of major research plan on "trustworthy software"," *Bulletin of National Natural Science Foundation of China*, vol. 32, no. 3, pp. 291–296, 2018.

[6] S. Becker, W. Hasselbring, A. Paul et al., "Trustworthy software systems: a discussion of basic concepts and terminology," *ACM SIGSOFT - Software Engineering Notes*, vol. 31, no. 6, pp. 1–18, 2006.

[7] H. Tao and Y. Chen, "A survey of software trustworthiness measurement validation," *International Journal of Performability Engineering*, vol. 14, no. 9, pp. 2056–2065, 2018.

[8] H. Tao, Y. Chen, and H. Wu, "A reallocation approach for software trustworthiness based on trustworthy attributes," *Mathematics*, vol. 8, no. 1, p. 14, 2019.

[9] T. Hongwei, C. Yixiang, W. Hengyang, and D. Rumei, "A survey of software trustworthiness measurements," *International Journal of Performability Engineering*, vol. 15, no. 9, pp. 2364–2372, 2019.

[10] H. Tao, Y. Chen, and J. Pang, "A software trustworthiness measure based on the decompositions of trustworthy attributes and its validation," in *Industrial Engineering, Management Science and Applications 2015*, M. Gen, K. J. Kim, X. Huang, and Y. Hiroshi, Eds., pp. 981–990, Springer, Berlin, Germany, 2015.

[11] K. Li, Y. Zhang, W. Liu, and J. Gao, "The application of fuzzy regression based on the trapezoidal fuzzy numbers to the software quality evaluation," *Journal of Convergence Information Technology*, vol. 7, no. 19, pp. 293–300, 2012.

[12] H. Tao and Y. Chen, "A quantitative relation model between trustworthy attributes," *Quantitative Logic and Soft Computing*, vol. 39, pp. 197–204, 2012.

[13] T. Bao, S. Liu, and X. Wang, "Research on trustworthiness evaluation method for domain software based on actual evidence," *Chinese Journal of Electronics*, vol. 20, no. 2, pp. 195–199, 2011.

[14] F. Di Cerbo, N. G. Mohammadi, and S. Paulus, "Evidence-based trustworthiness of internet-based services through controlled software development," in *Cyber Security and Privacy*, C Frances and F Massimo, Eds., pp. 91–102, Springer International Publishing, Midtown Manhattan, NY, USA, 2015.

[15] H. Tao and J. Zhao, "Source codes oriented software trustworthiness measure based on validation," *Mathematical Problems in Engineering*, vol. 2018, 2018.

[16] D. Wang, Q. Wang, and J. He, "Evidence-based software process trustworthiness model and evaluation method," *Journal of Software*, vol. 28, no. 7, pp. 1713–1731, 2017.

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

[18] K. Li, C. Chen, K. Zhao, and W. Liu, "A neural network ensemble model for software reliability prediction," *Information - An International Interdisciplinary Journal*, vol. 17, no. 4, pp. 1339–1350, 2014.

[19] L. Han, H. Tao, and Y. Chen, "An approach for trustworthy evidence of source code oriented aerospace software trustworthiness measurement," *Aerospace Contrd and Application*, vol. 47, no. 2, pp. 32–41, 2021.

[20] U. K. Tiwari, S. Kumar, and P. Matta, "Execution-history based reliability estimation for component-based software: considering reusability-ratio and interaction-ratio," *International Journal of System Assurance Engineering and Management*, vol. 11, no. 5, pp. 1003–1019, 2020.

[21] P. Malik, L. Nautiyal, and M. Ram, "A method for considering error propagation in reliability estimation of component-based software systems," *International Journal of Mathematical, Engineering and Management Sciences*, vol. 4, no. 3, pp. 635–653, 2019.

[22] T. Lu, C. Liu, H. Duan, and Q. Zeng, "Mining component-based software behavioral models using dynamic analysis," *IEEE Access*, vol. 8, pp. 68883–68894, 2020.

[23] B. Wang, D. Li, and S. Zhang, "The performance quantitative model based on the specification and relation of the component," *Mathematics*, vol. 7, no. 8, pp. 730–744, 2019.

[24] M. Wang, Y. Ma, G. Li, W. Zhou, and L. Chen, "Multi-value models for allocation of software component development costs based on trustworthiness," *IEEE Access*, vol. 8, pp. 122673–122684, 2020.

[25] S. C. Dimri, U. Kumar Tiwari, and M. Ram, "Reliability estimation of component based software system with path based model," *Nonlinear Studies*, vol. 26, no. 2, pp. 313–325, 2019.

[26] B.-H. Wang and Y.-X. Chen, "An approach of matching based on performance specification of components," in *Quantitative Logic and Soft Computing 2016*, T.-He Fan, S.-Li Chen, S.-M. Wang, and Y.-M. Li, Eds., pp. 171–180, Springer International Publishing, Midtown Manhattan, NY, USA, 2017.

[27] B. Meyer, "The grand challenge of trusted components," in *Proceedings of the 25th International Conference on Software Engineering*, pp. 660–667, Portland, OR, USA, May 2003.

[28] Y. Zheng, "Predicting trustworthiness for component software," in *Proceedings of the Third International Workshop on Security, Privacy and Trust in Pervasive and Ubiquitous Computing (SecPerU 2007)*, pp. 1–6, Istanbul, Turkey, July 2007.

[29] 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, no. 1, pp. 77–104, 2011.

[30] W. Zhou, Y. Ma, and H. Pan, "A novel trustworthiness measurement model based on weight and user feedback," *Chinese Journal of Electronics*, vol. 31, no. 4, pp. 612–625, 2022.

[31] X. Chen, Y. Fang, J. Chai, and Z. Xu, "Does intuitionistic fuzzy analytic hierarchy process work better than analytic hierarchy process?" *International Journal of Fuzzy Systems*, vol. 24, no. 2, pp. 909–924, 2021.

[32] L. Chen, "Research of the safety path of colleges and universities laboratory basing on the analysis of grey correlation degree," *Journal of Intelligent and Fuzzy Systems*, vol. 40, no. 4, pp. 7755–7762, 2021.

[33] S. M. Mubarak, *A Formal Component-Based Software Engineering Approach for Developing Trustworthy Systems*, PhD thesis, Concordia University, Montreal, Canada, 2009.

[34] Z. Wang and K.-K. Lau, "A taxonomy of software component models," in *Proceedings of the Euromicro conference*, pp. 88–95, Porto, Portugal, September 2005.

[35] T. Bures, P. Hnetynka, and F. Plasil, "Sofa 2.0: balancing advanced features in a hierarchical component model," in *Proceedings of the 4th International Conference on Software Engineering Research, Management and Applications (SERA'06)*, pp. 40–48, Seattle, WA, USA, August 2006.

[36] E. Bruneton, T. Coupaye, M. Leclercq, V. Quéma, and J. B. Stefani, "The FRACTAL component model and its support in java: experiences with auto-adaptive and reconfigurable systems," *Software: Practice and Experience*, vol. 36, no. 11-12, pp. 1257–1284, 2006.

[37] R. van Ommering, F. van der Linden, J. Kramer, and J. Magee, "The koala component model for consumer electronics software," *Computer*, vol. 33, no. 3, pp. 78–85, 2000.

[38] Y. Chen and H. Tao, *Software Trustworthiness Measurement and Enhancement Specification*, Science Press, Beijing, China, 2019.