Hindawi
Security and Communication Networks
Volume 2022, Article ID 7080367, 15 pages
https://doi.org/10.1155/2022/7080367

Research Article

SaaS Service Combinatorial Trustworthiness Measurement Method Based on Markov Theory and Cosine Similarity

Tilei Gao, Xiaohui Jia, Rong Jiang, Yuanyuan He, and Ming Yang

School of Information, Yunnan University of Finance and Economics, Kunming 650221, China

Correspondence should be addressed to Ming Yang; yangming@ynufe.edu.cn

Received 14 May 2022; Revised 7 August 2022; Accepted 20 August 2022; Published 16 September 2022

Academic Editor: Shah Nazir

With the rapid progress of information technology, cloud computing and cloud services are widely accepted and applied to all aspects of social life. In the cloud computing environment, SaaS (Software-as-a-Service) services have become the main form of software services. For SaaS services, evolutionary and iterative development methods have become the main methods of software system construction. For systems with high trustworthiness, the independent trustworthiness of each SaaS service has a great impact on the overall status. However, SaaS services with high independent trustworthiness do not always build highly trusted software systems. The combinatorial trustworthiness between SaaS services is as important as the independent trustworthiness of each SaaS service. This paper takes combinatorial trustworthiness between SaaS services as the research object. Combinatorial trustworthiness measurement method based on Markov and cosine similarity theory is proposed. The feasibility and effectiveness of the proposed method are verified through simulation experiments. Applicable scenarios, advantages, and disadvantages of the proposed method are shown through the comparison of different measurement methods. The proposed method provides theoretical and technical support for users to select SaaS services suitable for their application scenarios, build cloud service systems, and monitor the operation status of cloud service systems.

1. Introduction

With the progress of information technology, cloud computing has made great progress. Cloud services have been widely used in all aspects of social life, especially in today's highly competitive commercial enterprises. In order to reduce costs and improve organizational efficiency, it is the current trend for enterprises to adopt innovative services such as cloud services [1, 2]. Synergy research group released the global data center infrastructure revenue data in the first quarter of 2020. Data show that the global cloud computing (IaaS + PaaS) market revenue in the first quarter was US $29 billion, up 37% year-on-year. This is because the demand for e-commerce, streaming media, and telecommuting increased significantly during the epidemic. According to Flexera's 2020 cloud status report [3], 59% of enterprises expect cloud usage to exceed previous plans. It can be seen that the demand for cloud services in the global market is gradually increasing, and more and more institutions begin to choose cloud services to expand their applications. As the main form of software in the cloud computing environment, SaaS service has developed rapidly, especially favored by small and medium-sized enterprises [4].

In the cloud computing environment, the hierarchical structure of a software system is simpler and tends to be "flat" [5]. That is, each SaaS service represents a relatively independent function. Multiple SaaS services are combined into larger functional modules and the service system is formed through continuous superposition and combination between modules. For the trustworthiness of the SaaS service system, the trustworthiness of each SaaS service will directly affect the trustworthiness of the functional module and even the whole software system. However, the service system composed of multiple highly trusted SaaS services does not necessarily have high trustworthiness. The trustworthiness of a module or a system is affected not only by the independent trustworthiness of each service but also by the correlation between services. Therefore, the measurement of

SaaS service trustworthiness should not only measure the independent trustworthiness of each SaaS service but also measure the combinatorial trustworthiness of the SaaS services. Software testing in the cloud computing environment is mainly the evaluation results given by the software development organization (SDO) for the requirements and performance [6]. This activity is within the time period between the completion of the system and the delivery of the system. Different from testing, the measurement of service trustworthiness mainly focuses on the monitoring process of the operation process before and after the selection of services, and also pays more attention to the needs of other aspects other than system functions.

For users in specific scenarios, in addition to paying attention to the trustworthiness of the selected SaaS services, they pay more attention to the changes in the overall trustworthiness of the system after the selected services are integrated into the system. The objective of this paper is to design an overall trustworthiness measurement method to measure the overall trustworthiness of services after multiple services are integrated. In order to solve the above problems, this paper decomposes the overall trustworthiness into two parts: collaborative trustworthiness and local trustworthiness. In terms of collaborative trustworthiness, a collaborative trustworthiness measurement method based on Markov and cosine similarity theory is proposed. Local trustworthiness, according to its definition, is obtained by organic integration of relevant collaborative trustworthiness. The trustworthiness of the cloud service system is one of the most concerned issues when users use the cloud system. This problem directly determines the cost of users and the normal operation of normal business in the future. The method proposed in this paper provides theoretical and technical support for solving the most concerned problems of users in the cloud computing environment.

2. Literature Review

This paper focuses on the measurement method of SaaS service combinatorial trustworthiness. However, the research on SaaS service trustworthiness is relatively limited. SaaS service, as the main existing form in the new computing environment, is still the category of software service in essence. Therefore, this part combs the research progress of software service trustworthiness and related measurement and evaluation methods.

2.1. Concepts of Software Service Trustworthiness. In requirements engineering, nonfunctional requirements are usually regarded as the quality attributes of the software. Actually, it is the description of the software functional requirements, which depicts the degree of some attributes satisfied by the software system. The quality of the software is an objective evaluation of the software system, which will not change due to the different environment, personnel, and conditions [7]. Trustworthiness is an attribute with strong subjective preference, and its trustworthiness will show great differences in different application scenarios [8]. This difference is mainly reflected in the attention of users to the sub-attributes contained in software trustworthiness: some users pay attention to whether the functional requirements can be well realized, some users pay attention to whether the efficiency is high enough, and some users pay attention to whether the trustworthiness is better guaranteed. These are the categories of software trustworthiness. Therefore, some studies divide trustworthiness into broad trustworthiness and narrow trustworthiness [9]. Trustworthiness in a broad sense refers to comprehensive trustworthiness including function, structure, cost, and other nonfunctional demands and price; narrow trustworthiness only refers to the collection of nonfunctional attributes other than function and structure, that is, broad trustworthiness = functionality + structure + cost + narrow trustworthiness, and narrow trustworthiness = users' preference for quality attributes [9]. The main content of this project is trustworthiness in a narrow sense.

In terms of software service trustworthiness content, since the concept of trusted computing was put forward, the research on software trustworthiness [10] and trusted software [11] has gradually become one of the research hotspots in the field of software engineering, and its main research content is the construction and application of trustworthiness model. This paper summarizes the relevant research results of research institutions and scholars in various countries on the content of software trustworthiness since 2000. The results are shown in Table 1.

The existing research on trustworthiness content mainly focuses on the independent trustworthiness of software services. The purpose is to facilitate developers and publishers to find and describe the quality of their software service products and provide references and a basis for users to choose software services. For users in specific scenarios, the result of service independence and trustworthiness is only the first step in selecting services. Whether the selected services can play a satisfactory effect in their application scenarios is the direct factor determining whether users choose services.

2.2. Trustworthiness Measurement and Evaluation Method. Some research on software trustworthiness measurement methods mainly has two directions: one is the research on traditional measurement methods based on the software trustworthiness index model; the other is the research on general evaluation measurement methods.

The main work of the measurement and evaluation method based on the trustworthiness model is: in the face of a specific application scenario, through fine-tuning the selected quality model, setting a certain scoring standard, and using the way of artificial scoring to calculate and evaluate the trustworthiness of software services in this scenario. Such directions include demand-driven software trustworthiness evaluation and evolution model [25], software trustworthiness evaluation model based on evidence theory [26], runtime software trustworthiness evidence collection mechanism based on TPM [27], software service trustworthiness evaluation method based

Table 1: Software trustworthiness contents.

Sources	Trustworthiness contents	Time
Microsoft [12, 13]	Reliability, integrity, confidentiality, security	2002
Littlewood and Strigini [14], Schmidt [15]	Availability, reliability, risk prevention, safety, robustness	2003
DARPA's CHATS [16]	Reliability (fault tolerance), security (confidentiality, integrity, access control, authenticatability, auditability), survivability, performance (time efficiency, space efficiency)	2004
NSS2 [17]	Safety, risk prevention, reliability, survivability, performance	2005
TrustSoft [18]	Correctness, security, risk prevention, privacy, quality of service (performance, reliability, availability)	2005
COMPSAC [19]	Security, availability, reliability, confidentiality, recoverability, survivability, integrity	2006
Wang et al. [20]	Credible behavior, credible ability, and credible identity	2006
Safonov [21]	Confidentiality, ease of use, security, maintainability, reliability	2007
ICSP [22]	Functionality, risk prevention, reliability, safety, availability, maintainability, portability	2009
Feng et al. [23]	Transitive trust across system boundaries, independent trust within fixed boundaries	2014
Yang et al. [24]	Generalized reliability, generalized security, identity trust, capability trust, basic standard trust	2019

on subjective and objective comprehensive weighting [28], etc. Such methods rely too much on the selection of models, and their measurement and evaluation process completely depends on the personal quality of the scoring personnel and their familiarity with the scene. The stability and trustworthiness of the results have been greatly questioned. In addition, the process of this kind of measurement method involves a large number of activities participated by people, the degree of automation is very low, and the cost of economy, time, and manpower are huge.

General measurement and evaluation methods mainly include the AHP method and extension method based on AHP [29], the CMM/CMMI method proposed by Carnegie Mellon University [30, 31], the fault tree analysis method based on the decision tree and its deformation [32], method based on fuzzy set theory [33], method based on information entropy and entropy weight theory [34], state prediction and evaluation method based on Markov process [35], multi-attribute decision-making method based on the codified knowledge of field expert [36], etc. Most of the above measurement methods are general measurement methods, which can be used in different fields and different environments. The process is unified and the method is relatively simple. In the traditional field of system prediction and evaluation, it does solve many practical measurement and evaluation problems. However, in the face of the new computing environment, in addition to the problems of high human participation and strong subjectivity, this type of measurement method still has many deficiencies in other aspects: first, because it is a general measurement method, it is not targeted when facing virtual products such as software services, so it is still necessary to establish the measurement model of specific products first, The temporarily established measurement model is difficult to accurately describe all aspects of the trustworthiness requirements of software services in specific scenarios. Secondly, there is a separation between measurement methods and software services, and there is often a mismatch between measurement methods and models, which affects the stability and accuracy of measurement results. Thirdly, the measurement process still needs to invest a lot of time, manpower, and economic costs. It is difficult to realize real-time measurement and automatic measurement in the process of software system operation, and it is difficult to meet the requirements of users for rapid updates of the software system in the new environment.

Other methods include evaluation method based on fuzzy theory [37, 38], evaluation method based on D-S evidence theory [39, 40], evaluation method based on risk matrix [41, 42], trusted computing method based on trusted chain [43–45], prediction evaluation method based on Bayesian network [46, 47], etc. Although these single methods can effectively realize the quantitative evaluation of trustworthiness, they lack the evaluation of cloud service trustworthiness and its changes in practical application scenarios.

The existing measurement and evaluation methods, whether general methods or methods based on the software service trustworthiness model, mostly measure and evaluate the quality of software service individuals from the perspective of developers and suppliers, and lack the consideration of the adaptability of specific scenarios. For users, they pay more attention to whether the selected products can improve the trustworthy experience in the user's scene. For software service individuals, they only pay attention to their price and whether they can be integrated into their own environment. Therefore, based on the measurement and evaluation of individual quality and trustworthiness of software services, there is still a lack of a measurement and evaluation method for the trustworthiness of service composition in specific scenarios.

3. Concepts of SaaS Service Combinatorial Trustworthiness

Definition 1. (*SaaS service combinatorial trustworthiness*) SaaS service combinatorial trustworthiness includes two aspects: one is *TC* (trustworthiness of collaboration) between service pairs (SP) composed of two SaaS services; the second is the local trustworthiness *TL* (trustworthiness of local) of the whole (service block) composed of all SaaS service sets directly associated with the SaaS service to be examined.

The combinatorial trustworthiness of SaaS service is not only related to the trustworthiness of each service itself but also related to the trustworthiness of the composition mode of directly related service composition. In the following content, the article will introduce the related concepts of collaborative trustworthiness TC and local trustworthiness TL of software services respectively.

3.1. Trustworthiness of Collaboration (TC)

Definition 2. (*SaaS service collaborative trustworthiness TC*) Suppose $S = \{s_1, s_2, \ldots, s_n\}$ is a collection of all SaaS services in the software system on the cloud computing platform, $\forall s_i, s_j \in S$. if there is direct message communication between SaaS services s_i and s_j, then s_i and s_j are directly related. The trustworthiness between s_i and s_j is called the collaborative trustworthiness of s_i and s_j, which is recorded as TC_{ij}.

The external performance of a system affected by risk factors reflects the trustworthiness of the system. Under the action of the same risk factors, the smaller the change of system trustworthiness state, that is, it is less affected by risk, the stronger the ability of the system to resist risk, the higher the trustworthiness of the system, and vice versa. Therefore, this paper takes the fluctuation of SaaS services after all risk factors as collaborative trustworthiness. For a SaaS service pair, the trustworthiness state before being affected by all risk influencing factors (hereinafter referred to as influencing factors or factors) can be abstracted into a multi-dimensional vector, and the other trustworthiness state obtained after being affected by the influencing factor matrix can also be abstracted into a multi-dimensional vector. Using the cosine theorem, the distance between them can be calculated, that is, the fluctuation before and after the set of influencing factors, That is, the trustworthiness of SaaS services. The status before being affected by factors can be determined according to specific application scenarios.

In addition to the trustworthiness of the two SaaS services, the collaborative trustworthiness TC_{ij} of the service pair (s_i, s_j) is also affected by α_k $(k = 1, 2, \ldots, n)$, which is all the factors affecting the trustworthiness of the two services. When different α_k acts, the trusted state of the service to s_i-s_j can be transferred from q_t to q_{t+1} (that is, the occurrence of influencing factors is taken as the time in the traditional Markov chain). It is assumed that the effects of different factors on Collaborative trustworthiness TC_{ij} are independent of each other. Therefore, the change process of different influencing factors on the trusted state satisfies the nature of the Markov process.

The action of many different factors can be regarded as the observation sequence of multiple influencing factors, that is, the influence of multiple influencing factors on the collaborative trustworthiness of service pair (s_i, s_j) can be regarded as the order of each influencing factor α_k, which has one or more effects on the collaborative trustworthiness. Once for each factor, the impact on the collaborative trustworthiness is regarded as a step. Combined with the probability $p(\alpha_k)$ of occurrence of each influencing factor α_k, in a certain environment, the probability of each influencing factor is constant. According to the specified sequence of influencing factors, the impact of each influencing factor α_k on the trustworthiness can be calculated, so as to finally obtain the stable result $TC_{ij}^{(h)}$ of the collaborative trustworthiness of composite services, where h is the number of influencing factors that play a role, that is, the steps of Markov chain.

In essence, the collaborative trustworthiness TC of service pairs composed of two services is mainly determined by the sequence of all trustworthiness influencing factors of the two services. The occurrence of each influencing factor will change the result of collaborative trustworthiness. In the influencing factor sequence, the collaborative trustworthiness between the two services will become stable after all the influencing factors act in a certain order. The greater the influence of influencing factors on the initial state of service, the lower the trustworthiness of collaboration, and vice versa. Therefore, the essence of computing collaborative trustworthiness is to calculate the changes of the trust state before and after the affected factors. The cosine value of the vector is often used to calculate the similarity (difference) between multi-dimensional vectors in space. In this paper, the States H and H' before and after the affected factors are regarded as two vectors, and the cosine value of the two is calculated to represent the difference. Then the collaborative trustworthiness of a pair of SaaS service pairs can be expressed as

$$TC_{ij}^{\mathscr{R}} = \cos\theta = \frac{\mathbf{H} \cdot \mathbf{H}'}{\|\mathbf{H}\| \times \|\mathbf{H}'\|}. \tag{1}$$

In the formula,

$$\mathbf{H} \cdot \mathbf{H}' = \sum_{\mathbf{i}=1}^{\mathbf{m}} (t_i \times t_i'),$$
$$\|\mathbf{H}\| \times \|\mathbf{H}'\| = \sqrt{\sum_{\mathbf{i}=1}^{\mathbf{m}} (\mathbf{t_i})^2} \times \sqrt{\sum_{\mathbf{i}=1}^{\mathbf{m}} (\mathbf{t_i'})^2}. \tag{2}$$

(1) $\mathbf{H} = (t_1, t_2, \ldots, t_m)$, is all credible state vectors before the affected factors in collaborative trustworthiness take effect; $H' = (t_1', t_2', \ldots, t_m')$, is all credible state vectors after the influence of factors in collaborative trustworthiness.

(2) $\mathscr{R}$ is the occurrence sequence of influencing factors, that is, a sort of set $A = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ of all trustworthiness influencing factors.

(3) The closer the cosine value is to 1, the closer the angle between the two vectors is to 0 degrees, that is, the more similar the two vectors are, and the included angle is equal to 0, that is, the two vectors are equal. The greater the cosine value between the two state vectors H and H', the smaller the included angle and

the less affected by the outside world, the higher its trustworthiness $TC_{ij}^{\mathscr{R}}$.

According to the matrix multiplication calculation method, the final calculation results will be different due to the different ranking sequences of influencing factors. However, the monotonicity of the same initial state remains the same for different sequences, that is, for the same initial state and different influencing factor sequences, if the trustworthiness of the termination state of a sequence is greater than that of the initial state, the trustworthiness of the termination state of other sequences is also greater than that of the initial state. In other words, if the monotonicity of a sequence is known, the monotonicity of other sequences can be determined. In the process of actual SaaS system evolution or construction, collaborative trustworthiness is to judge the collaborative trustworthiness after evolution or construction and the change of size before evolution. It is a relative result. Just judge the trustworthiness before and after evolution to predict the success of this evolution or construction. Therefore, it is only necessary to calculate the same influencing factor sequence for the services before and after evolution or construction. It can judge the change of trustworthiness after evolution or construction without calculating the results of each sequence.

(4) If $(t_1, t_2, \ldots, t_m)$ represents the initial trustworthiness state vector of s_i and s_j combination and $(t_1', t_2', \ldots, t_m')$ represents the termination trustworthiness state vector of s_i and s_j combination after sequence $\mathscr{R}$, assuming that sequence $\mathscr{R}$ is a sequential sequence (the sequence is set according to the actual situation during actual calculation), then,

$$(t_1, t_2, \ldots, t_m) \xrightarrow{\alpha_1} \left(t_1^{(1)}, t_2^{(1)}, \ldots, t_m^{(1)}\right) \xrightarrow{\alpha_2} \cdots \xrightarrow{\alpha_k} \left(t_1^{(k)}, t_2^{(k)}, \ldots, t_m^{(k)}\right) \xrightarrow{\alpha_{k+1}} \cdots \xrightarrow{\alpha_n} (t_1', t_2', \ldots, t_m'). \tag{3}$$

$$\text{or } (t_1, t_2, \ldots, t_m) \xrightarrow{\mathscr{R}} (t_1', t_2', \ldots, t_m'). \tag{4}$$

According to the relevant theories of Markov chain and hidden Markov chain, each influencing factor in sequence $\mathscr{R}$ represents a time, then each influencing factor α_k is a state transition matrix $\mathbf{M_k}$, and the state transition matrix of each influencing factor is a square matrix of $m \times m$ according to the initial and termination state vectors. For example, the state transition matrix of the k-th influencing factor is as follows:

$$\mathbf{M_k} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \cdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mm} \end{bmatrix}. \tag{5}$$

In the matrix, i and j represent the status serial number, which is consistent with the status serial number in $(t_1, t_2, \ldots, t_m)$; $a_{ij}(i \neq j)$ represents the probability of the state transferring from i to j after the influencing factors act; $a_{ij}(i = j)$ represents the probability that the state will not transfer after the influencing factor acts.

Let the probability of occurrence of α_k be $p(\alpha_k)$, then,

$$(t_1', t_2', \ldots, t_m') = (t_1, t_2, \ldots, t_m) \times p(\alpha_1)\mathbf{M}_1 \times p(\alpha_2)\mathbf{M}_2 \times \cdots \times p(\alpha_n)\mathbf{M_n}. \tag{6}$$

(5) $TC_{ij}^{\mathscr{R}}$ represents the combination of s_i and s_i, corresponding to the fluctuation of influencing factor sequence R, that is, the collaborative trustworthiness of service pairs, which can also be abbreviated as TC_{ij}.

3.2. Trustworthiness of Local (TL)

Definition 3. (*SaaS service local block*$\mathbf{S_L^i}$) Suppose s_i is a SaaS service in the cloud computing software system, and the set $\mathbf{S_L^i} = \{s_1, s_2, \ldots, s_n\}$ of SaaS services with direct messaging mechanism with s_i is the SaaS service local block composed of service s_i, which is referred to as local block or service block for short. In Figure 1, the part in the dotted line box is the service block S_L^i centered on s_i. The service block only focuses on the collaborative trustworthiness between service pairs directly associated with s_i, and the collaborative trustworthiness between other services and SaaS services is not within the scope of this paper. Therefore, the collaborative trustworthiness of service pair (s_1, s_2) in the figure does not belong to the scope of this local block, and TC_{12} is not included in the trustworthiness calculation of local block S_L^i.

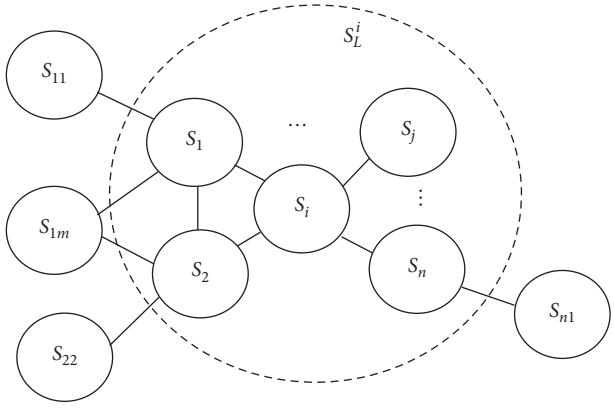

Figure 1: SaaS service local block S_L^i.

Definition 4. (*Local trustworthiness of SaaS service system TL*) Let $S_L^i = \{s_1, s_2, \ldots, s_n\}$ be a local block composed of service s_i, then the local trustworthiness TL_{s_i} of local block S_L^i is the weighted sum of all cooperative trustworthiness TC_{ij} related to s_i, i.e.,

$$TL_{s_i} = \sum_{j=1}^{n} \left(\omega_j \times TC_{ij}^{\mathscr{R}}\right). \tag{7}$$

In the formula, ω_j represents the importance of the collaboration trustworthiness of each pair of services related to s_i in the local block S_L^i, which is generally expressed by the association degree of other services with s_i.

For the evolution or construction of SaaS services, local trustworthiness only focuses on the services directly associated with the services to be evolved or constructed and has nothing to do with the services indirectly associated.

For instance, let s_e be one SaaS service in the system to be evolved. If there is a service composition sequence (s_e, s_i, s_{i+1}), s_e is directly associated with s_i, and s_e is indirectly associated with s_{i+1} through s_i, then the cooperative trustworthiness between s_e and s_i is independent of s_{i+1}. Because in the cloud computing environment, in order to realize the integration of products developed by different developers, SaaS services are highly encapsulated, SaaS services have a high degree of independence, and services interact only through public channels and message passing mechanisms. For the service s_e, the nondirectly related SaaS Service s_{i+1} is transparent. s_e does not know the existence of s_{i+1}. s_e only interacts with s_i. The collaborative trustworthiness between (s_e, s_i) has nothing to do with s_{i+1}. The cooperative trustworthiness between (s_e, s_{i+1}) is relatively independent and is not considered in this evolution. Therefore, the local trustworthiness *TL* of the software system in the cloud computing environment only calculates the collaborative trustworthiness between all SaaS services directly associated with the SaaS service s_e, and does not calculate the association trustworthiness or recommendation trustworthiness between other nondirectly associated services.

4. SaaS Service Combinatorial Trustworthiness Measurement Method

To solve the problem of SaaS service combinatorial trustworthiness, this paper proposes a method of SaaS service combinatorial trustworthiness measurement based on Markov chain and cosine similarity. The trustworthiness of service composition can be measured by calculating the changes in the trusted state of service composition in specific application scenarios. Markov chain theory is often used to express the state changes of things when they are affected by external factors. This paper studies the state changes of the trusted state of service composition when affected by different risk factors, which is consistent with Markov chain theory. The combined trustworthiness state of SaaS service is a spatial multi-dimensional vector. Calculating the combined trustworthiness state is to calculate the distance between spatial multi-dimensional vectors. Cosine similarity is often used to calculate the similarity or difference between spatial multi-dimensional vectors. Therefore, this paper selects Markov chain and cosine similarity as the basic theory and method of SaaS service combinatorial trustworthiness measurement.

4.1. Overall Framework of SaaS Service Combinatorial Trustworthiness Measurement. This paper proposes a calculation method of SaaS service collaborative trustworthiness based on Markov chain theory and a measurement method of SaaS service local trustworthiness based on Collaborative trustworthiness. The overall measurement model is shown in Figure 2.

The whole measurement model is divided into four levels. The bottom layer is the data source layer. The original data required for measurement are obtained or calculated by means of expert scoring or questionnaire survey. These data include the transition matrix $\mathbf{M_k}$ composed of the occurrence probability $p(\alpha_k)$ of SaaS service to each influencing factor α_k in (s_i, s_j) and the change probability of each service trustworthiness in the service pair under the action of each influencing factor. Above the data acquisition layer is the influencing factor layer, which analyzes and obtains all the influencing factor sets of SaaS services on Collaborative trustworthiness, and determines all the influencing factor sets $A = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ of SaaS services on (s_i, s_j). Above the influencing factors is the service-pair layer. This layer includes all SaaS service pairs that need to calculate collaborative trustworthiness. According to each service pair and trustworthiness influencing factors, the association relationship required for collaborative trustworthiness calculation is constructed. Suppose there are m States, $H = (t_1, t_2, \ldots, t_m)$ represents the trustworthiness state vector of a service pair in the initial state, and $H' = (t_1', t_2', \ldots, t_m')$ represents the trustworthiness state vector of the service pair after the sequence of influencing factors. The value of state vector can be calculated according to the transition matrix composed of all influencing factor sequences and the occurrence probability of each factor. The cosine value calculated for the two state vectors represents

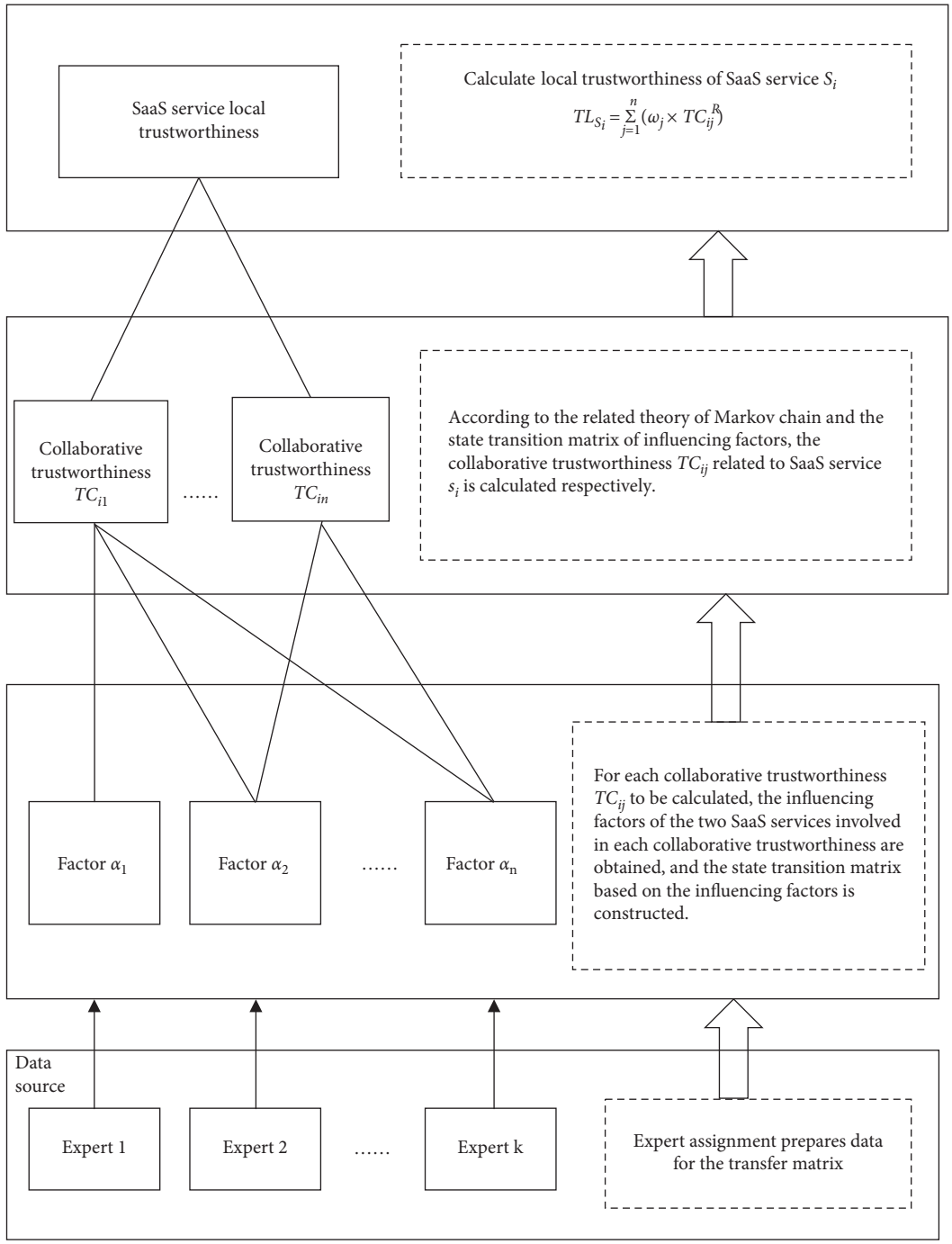

Figure 2: Overall framework of SaaS service combinatorial trustworthiness measurement.

the influence degree of all influencing factors on the combined collaborative trustworthiness, that is, collaborative trustworthiness. The top layer is SaaS service local trustworthiness. Combined with the weight of each service pair, the local trustworthiness TL_{s_i} centered on a service s_i can be calculated.

4.2. SaaS Service Combinatorial Trustworthiness Measurement Procedure. According to the measurement model framework, the local trustworthiness measurement method is shown in Figure 3.

Suppose $\mathbf{S_L^i}$ is a SaaS service local block composed of service s_i, $\mathbf{S_L^i} = \{s_1, s_2, \ldots, s_m\}$, its local trustworthiness $TL_{s_i}L$, and the set of all influencing factors of SaaS service on (s_i, s_j) is $\mathbf{A} = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$.

Inputs: (1) Occurrence probability p(αk) of all influencing factors of service pair; (2) the state matrix $\mathbf{M_k}$ composed of influence degrees of each influencing factor αk; (3) the trustworthiness weight c_r of each service pair in the service block.

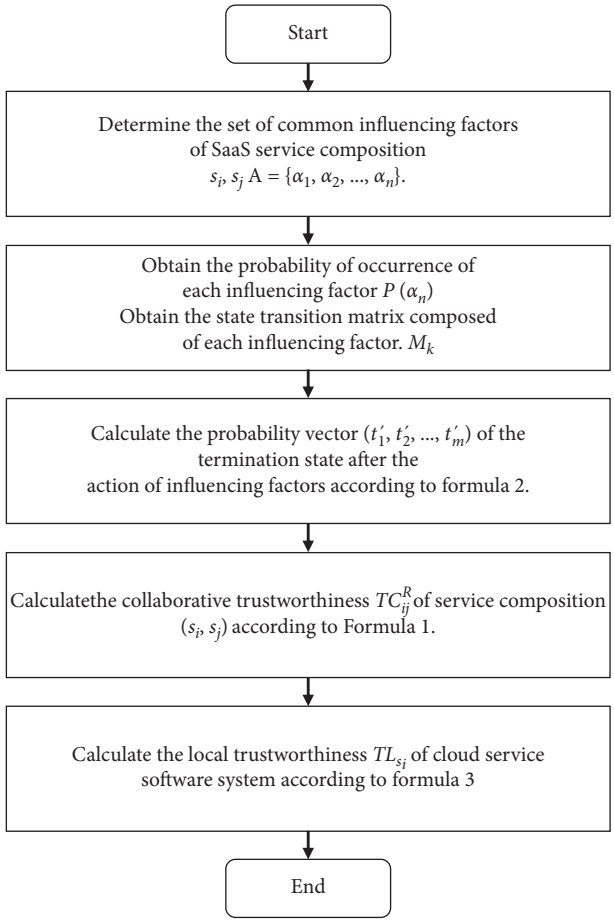

Figure 3: SaaS service combinatorial trustworthiness measurement procedure.

Outputs: Trustworthiness TL_{s_i} of s_i centered SaaS service block $\mathbf{S_L^i}$.

The specific calculation process is as follows:

Step 1: Obtain the initial state $p(\alpha k)$ of SaaS service trustworthiness and the probability of occurrence of all influencing factors.

Step 2: Obtain the influence $p_{ij}(\alpha_k)$ of each influencing factor αk on the trustworthiness of services s_i and s_j, and take this as the input to build the state transition matrix $\mathbf{M_k}$ of service trustworthiness

Step 3: Take the combined $(t_1, t_2, \ldots, t_m)$ composed of the trustworthiness of SaaS services s_i and s_j as the initial state of the cooperative trustworthiness state transition, and substitute it into formula (6) together with $\mathbf{M_k}$ to calculate the termination state $(t_1', t_2', \ldots, t_m')$ of the collaborative trustworthiness of SaaS services I and j after a given sequence.

Step 4: Substituting the initial trustworthiness state H and termination trustworthiness state H' of SaaS service pair (s_i, s_j) into formula (1) to calculate the collaborative trustworthiness $TC_{ij}^{\mathscr{R}}$ of the service to (s_i, s_j) in a given influencing factor sequence $\mathscr{R}$.

Step 5: Repeat steps 1–5 to calculate the collaboration trustworthiness of each service pair formed with s_i.

Step 6: In the s_i centered SaaS service block $\mathbf{S_L^i}$, the message passing times c_r of each other service s_r and s_i can be regarded as the importance of this service to (s_i, s_r) in this service block. Normalizing the message passing times within each service pair can obtain the collaborative trustworthiness weight ω_r of each service pair.

Step 7: Substituting the collaborative trustworthiness $TC_{ij}^{\mathscr{R}}$ of each service pair and its corresponding weight ω_j into formula (7) can finally calculate the trustworthiness TL_{s_i} of SaaS service block $\mathbf{S_L^i}$ centered on s_i, that is, the local trustworthiness of a service block in the cloud service software system.

In the specific calculation process, it is difficult to obtain the data of the initial state of service pair. Therefore, usually, the data we get through logs or tests are already in the state after being affected by influencing factors. In the service composition measurement method for evolution or system construction, what needs to be calculated is the relative trustworthiness of service composition, that is, the result of whether the trustworthiness is improved before and after evolution or construction. Therefore, the calculation process should be modified appropriately to meet the needs of evolution or system construction. In the process of specific evolution or system construction, it is assumed that the context before and after evolution or construction is consistent, that is, for different service pairs, under the same environment, the probability and effect of influencing factors before and after evolution or construction are the same, then the state transition matrix is the same. For the convenience of expression, this paper abstracts all the transition matrices into M, is the result of multiplying all state transition matrices, then formula (6) can be transformed into formula,

$$(t_1', t_2', \ldots, t_m') = (t_1, t_2, \ldots, t_m) \times \mathbf{M}. \quad (8)$$

Before evolution or system construction, the trustworthiness state of the service pair to be measured can be obtained, that is, the current trustworthiness state $H_0' = (t_1', t_2', \ldots, t_m')$, and the initial state data $H_0 = (t_1, t_2, \ldots, t_m)$ can be randomly generated, so that the value of the transition matrix $\mathbf{M}$ can be calculated, and the cosine values of H_0 and H_0' can be calculated as the collaborative trustworthiness TC_0 of the service pair before evolution, which can be substituted into formula (7) to calculate the local trustworthiness TL_0 before evolution or construction.

After evolution, a new service pair is formed after the evolved service is replaced by a new service. Through the data test, obtain the new current state data $H_1' = (t_1', t_2', \ldots, t_m')$, and substitute it into formula (8) together with the transfer matrix $\mathbf{M}$ to obtain the initial state $H_1 = (t_1, t_2, \ldots, t_m)$. Then, the cosine values of H_1 and H_1' are used as the trustworthiness TC_1 after evolution and substituted into formula (7) to calculate the local

trustworthiness TL_1 after evolution, so as to realize the comparison of trustworthiness before and after evolution and the comparison of trustworthiness between each service pair.

4.3. Algorithm of SaaS Service Combinatorial Trustworthiness Measurement. Algorithm of SaaS service combinatorial trustworthiness measurement is shown in Algorithm 1.

Time Complexity Analysis of Algorithm 1. The order of the state transition matrix and the order of the state vector in the algorithm are both m. The algorithm includes the multiplication of multiple matrices and the multiplication of vectors and matrices. It is generally considered that the time complexity of matrix multiplication is $O(n^3)$. The analysis here also counts the operation times of all matrix multiplication and vector-matrix multiplication as m^3 times.

The algorithm includes one loop and six summation operations. The first cycle (lines 2 to 8) has run for $(n-1)$ times to calculate the collaborative trustworthiness of each service pair and the third line in the loop is matrix operation, which is executed m^3 times. Line 4 contains two summations. The summation operation also includes m times of multiplication, and the total number of executions is $4m$. Lines 5 and 6 are used to find the inverse matrix, which is executed $2m^3$ times in total. Line 7 contains two summation operations. The summation operation also includes m times of multiplication, which is executed $4m$ times in total. Lines 9 and 10 each contain a summation operation, and the summation operation also includes m times of multiplication, which is executed for a total of $4m$ times. To sum up, the total execution times are $(n-1)*(m^3+4m+2m^3+4m+4m)$, m and n are variables, so the time complexity of the algorithm is a quartic function, that is $O(n^4)$.

In the algorithm, the variable n represents the number of services contained in the service block. The variable m represents the number of components contained in the state vector (also the order of the matrix). Generally, the service block contains services that have a direct communication relationship with the services to be evolved. In the cloud computing environment, the number is less than 10. The number of components in the state vector is artificially specified, and the number of trustworthiness states can be set according to the actual needs. Trustworthiness measurement is to calculate the improvement of trustworthiness before and after the performance. A large number of States will not have a significant impact on the quality of the results. Therefore, the state of trustworthiness is generally between 3 and 6. Although the time complexity of the algorithm is a quartic function, it is still within the acceptable range due to the limited amount of data and will not increase infinitely.

5. Simulation Experiment

In this paper, SaaS service combinatorial trustworthiness is divided into two types: collaborative trustworthiness within SaaS service pair and local trustworthiness of service blocks centered on some SaaS service. According to formula (7), the result of local trustworthiness is equal to the product of the collaborative trustworthiness of each service pair and its weight. In specific application scenarios, the collaborative trustworthiness weights are relatively stable within different service blocks. The weight can be calculated by the number of messages passed in the service block, or according to the actual situation and classical weight calculation methods, such as the AHP method and entropy weight coefficient method. The focus of this paper is the research on the measurement method of collaborative trustworthiness between SaaS service pairs. Therefore, to more clearly verify the combinatorial trustworthiness measurement method proposed in this paper, the experimental part is only to verify the collaborative trustworthiness of service pairs. In the simulation experiment, a service block composed of only two SaaS services is constructed as the experimental object. Select multiple SaaS services with similar functions and obvious trustworthiness differences to replace the SaaS services in the service block as the experimental process. The trustworthiness changes before and after the replacement of different SaaS services are calculated and matched with the expected results, so as to verify the feasibility and effectiveness of the method proposed in this paper.

5.1. Experimental Process

Step 1: Obtain the trustworthiness status data of the current SaaS service pair. In terms of the source of experimental data, first find a service with single function and strong scalability from GitHub, which is recorded as s_a. Randomly find a SaaS service s_b that can interact with it to form a service pair (s_a, s_b). Then, find three services s_c, s_d, and s_e that have similar functions to s_b and can be integrated with services. 10000 data are randomly generated. These data are used as the inputs of (s_a, s_b), (s_a, s_c), (s_a, s_d), and (s_a, s_e). Count the number of errors (i.e. untrusted state t_1, including the number of times when no results are obtained) and the number of normal times, respectively. In the normal times, it is divided into acceptable state t_2 and credible state t_3 according to the response time, and, respectively, calculate the untrusted probability $p'(t_1)$, acceptable probability $p'(t_2)$, and credible probability $p'(t_3)$. Then, $H' = (p'(t_1), p'(t_2), p'(t_2))$. The results are shown in Table 2.

Step 2: Calculate the Collaborative trustworthiness TC_{ab} of service pair (s_a, s_b). Randomly generated or specified (s_a, s_b) initial state data $H_a = (p_a(t_1), p_a(t_2), p_a(t_3)) = (0.3, 0.3, 0.4)$. Substituting H_a and H_a' into formula (1), $TC_{ab} = \cos(s_a, s_b) = 0.0093$. can be obtained.

Step 3: Calculate the state transition matrix **M**. According to formula (8), solve the matrix equation and obtain the result of a state transition matrix M,

Inputs: Service block $\mathbf{S_L^i} = \{s_1, s_2, \ldots, s_e, \ldots, s_n\}$, Service to be replaced s_e, the initial state vector set $H = \{H_1, H_2, \ldots, H_{n-1}\}$ of the service pair composed of s_e and other services in the service block, the current state vector set $H' = \{H_1', H_2', \ldots, H_{n-1}'\}$ of the service pair composed of s_e and other services in the service block, Service s (used for replacing s_e) , the current state vector set $H_s' = \{H_{s1}', H_{s2}', \ldots, H_{s(n-1)}'\}$ of the service pair composed of s and other services in the service block, each service pair weight $W = \{\omega_1, \omega_2, \ldots, \omega_{n-1}\}$.
Outputs: The local trustworthiness TL_0 of the pre-evolution service block and the local trustworthiness TL_1 of the post evolution service block.

(1) Begin:
(2) For each $i \in [1, n-1]$ do
(3) $H_i \times M_i = H_i' \cdot M_i$ // M_i is the state transition matrix of service pair (s_e, s_i).
(4) $TC_i \leftarrow H_i \times H_i' / \sqrt[2]{\sum_{j=1}^{m} (t_j)^2} \times \sqrt[2]{\sum_{j=1}^{m} (t_j')^2}$ // t_j and t_j' are the components of vectors H_i and// H_i', respectively.
(5) $M_i \leftarrow M_i^{-1}$ //Find the inverse matrix of M_i.
(6) $H_{si} \leftarrow H_{si}' \times M_i$
(7) $TC_i' \leftarrow H_{si} \times H_{si}' / \sqrt[2]{\sum_{j=1}^{m} (t_{sj})^2} \times \sqrt[2]{\sum_{j=1}^{m} (t_{sj}')^2}$ // t_{sj} and t_{sj}' are the components of vectors H_{si} //and H_{si}', respectively.
(8) End for
(9) $TL_0 \leftarrow \sum_{i=1}^{n-1} (\omega_i \times TC_i)$
(10) $TL_1 \leftarrow \sum_{i=1}^{n-1} (\omega_i \times TC_i')$
(11) End

Algorithm 1: SaaS service combinatorial trustworthiness measurement algorithm.

Table 2: Current status H' of four service pairs.

	(s_a, s_b)	(s_a, s_c)	(s_a, s_d)	(s_a, s_e)
$p'(t_1)$	0.0154	0.0190	0.0287	0.0227
$p'(t_2)$	0.1477	0.1938	0.1792	0.1455
$p'(t_3)$	0.8369	0.7872	0.7921	0.8318

$$\mathbf{M} = \begin{bmatrix} 0.0513 & 0 & 0 \\ 0 & 0.4923 & 0 \\ 0 & 0 & 2.0923 \end{bmatrix}. \quad (9)$$

Step 4: Calculate the initial state of other service pairs. Substitute the current state data of service pairs (s_a, s_c), (s_a, s_d), and (s_a, s_e) in the table together with the state transition matrix $\mathbf{M}$ into formula (8), and calculate the initial state results of the three service pairs. The results are shown in Table 3.

Step 5: Calculate the collaborative trustworthiness TC of other service pairs. Substitute the data in Tables 1 and 3 into formula (1), respectively, and calculate the cosine value of each service pair, that is, the collaborative trustworthiness. The results are shown in Table 4 and Figure 4.

Service pair (s_a, s_b) is the basis of calculation. The input data are randomly assigned to calculate the state transition matrix. For the other three service pairs, it is used to calculate their initial states according to the state transition matrix and their current states. Then, we can get the fluctuation of different service pairs under the same influencing factors, that is, collaborative trustworthiness. According to Table 4 and Figure 4, the service pair calculation results show that the collaborative trustworthiness of (s_a, s_c), (s_a, s_d), and (s_a, s_e) is higher than the value of the original service pair, and the value of (s_a, s_d) is the largest. Therefore, to implement evolution, s_d should be selected to implement evolution instead of service s_b.

Table 3: Initial states H of four service pairs.

	(s_a, s_b)	(s_a, s_c)	(s_a, s_d)	(s_a, s_e)
$p(t_1)$	0.3	0.3701	0.5591	0.4422
$p(t_2)$	0.3	0.3936	0.3640	0.2955
$p(t_3)$	0.4	0.3762	0.3786	0.3976

Table 4: Collaborative trustworthiness TC of four service pairs.

	(s_a, s_b)	(s_a, s_c)	(s_a, s_d)	(s_a, s_e)
TC	0.0093	0.0132	0.0257	0.0179

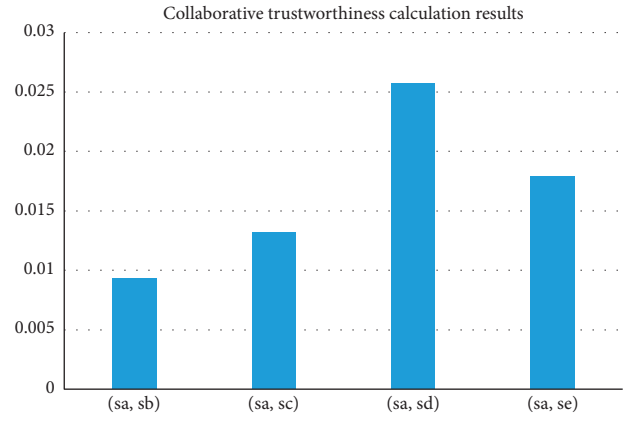

Figure 4: Collaborative trustworthiness calculation results.

5.2. Discussion and Analysis. According to the results summarized in section 2.2, most of the existing trustworthiness measurement methods are aimed at the measurement of a single service itself. Even a module or complex system composed of multiple services is measured as a "big service." This type of measurement method realizes the measurement of services by analyzing the constituent aspects and influencing factors of trustworthiness. For simple services, this method has high application value. However, for complex modules or systems, the overall trustworthiness is affected not only by a single service but also by the combination of services. At the same time, in the face of different application scenarios, the complexity of the factors affecting the trustworthiness of the system will increase exponentially. The existing trustworthiness measurement and evaluation methods are difficult to meet the challenges of complex systems. For specific users, their focus is often only on the overall or local trustworthiness of the system, and they are not sensitive to specific details or the trustworthiness of a single service.

Therefore, according to the specific requirements in this scenario, this paper designs a combined trustworthiness measurement method based on Markov theory and cosine similarity. This paper presents an effective solution to the problem that users are most concerned about, that is, the result of overall trustworthiness. Next, the paper shows the effectiveness of this method by comparing the results with those of classical methods. Through the comparison with other typical trustworthiness measurement methods, the advantages and disadvantages of this method are shown.

5.2.1. Effectiveness of the Proposed Method. The AHP method is a recognized, classical, and widely used measurement and evaluation method. In the face of a simple evaluation object, the results often have high trustworthiness. The purpose of the simulation experiment is to verify the effectiveness of the proposed method. The designed experiment is not complicated, and convincing results can be obtained by using the AHP method. Therefore, this paper compares the measurement results of the AHP method to verify the effectiveness of this method.

When using the AHP method to evaluate trustworthiness, it is necessary to establish an effective trustworthiness hierarchy model. We chose the trustworthiness model proposed by Tilei [48] et al. as the calculation model. The calculation process is implemented regarding literature [49]. For the experimental objects (s_a, s_b), (s_a, s_c), (s_a, s_d), and (s_a, s_e) in section 5.1, the trustworthiness results of each service pair are calculated, respectively. The calculation results are shown in Table 5.

The results calculated by the AHP method and the results calculated by this method are shown in Figure 5.

Trustworthiness is a subjective and objective evaluation standard. The calculation result of trustworthiness is a relative result, not an absolute value. The calculated trustworthiness value does not have practical significance but only reflects the high-level relationship between different objects. In other words, the result of trustworthiness calculation only reflects whether the trustworthiness of one service is higher than that of another, and has nothing to do with the specific value (whether it is 99 or 0.099). Therefore, for the same measurement object, the results calculated by different methods only obtain the relative relationship between different objects, and the actual value is not important.

Table 5: Trustworthiness of the AHP method.

	(s_a, s_b)	(s_a, s_c)	(s_a, s_d)	(s_a, s_e)
The AHP method	0.0196	0.0311	0.0493	0.0331
Our method	0.0093	0.0132	0.0257	0.0179

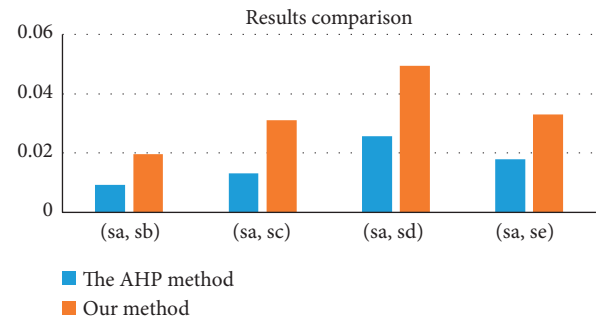

Figure 5: Results comparison.

According to the calculation results, $TC_{ad} > TC_{ae} > TC_{ac} > TC_{ab}$, the results are consistent with those calculated by our method. Therefore, it can be shown that the method proposed in this paper is effective in calculating the combinatorial trustworthiness in the current scenario.

5.2.2. Methods Comparison. The existing research results of software service measurement methods, whether from the perspective of products, processes, and services, or from the perspective of local optimization and global optimization, essentially analyze and calculate the independent trustworthiness of software services. Even if some methods decompose the process or product, its essence is to measure the decomposed parts as constituent elements. These belong to the research category of independent trustworthiness, and there are few measurement methods for combinatorial trustworthiness.

Next, this paper will compare the proposed trustworthiness measurement method with the AHP method and a recommended trust relationship model (RTRM) based on the recommended trust relationship model [50] from the five aspects of ease of use, objectivity, universality, functionality and costs, so as to illustrate the advantages and disadvantages of the proposed method.

AHP method is a classical measurement and evaluation method. When the hierarchical model is clear and the scoring data are reasonable, the results have high reference value. Its ease of use, objectivity, versatility, functionality, and costs are shown in Table 6.

When RTRM method calculates the service combinatorial trustworthiness, it not only calculates the trust

Table 6: AHP method.

AHP method	Descriptions
Ease of use	It has high requirements for the level clarity of the model and the rationality of expert scoring. For complex systems, it is difficult to obtain a correct, clear, and complete hierarchical model at one time. It is difficult to find multiple field experts at one time. Therefore, the ease of use is not high.
Objectivity	It is a subjective weighting method with strong interpretability and objective calculation process, but the input data is highly subjective. Therefore, its objectivity is not high.
Versatility	It is generally applicable to various measurement and evaluation scenarios, so it has strong universality.
Functionality	Each measurement and evaluation operation can only measure a certain model or aspect, so its functionality is weak.
Costs	It is necessary to establish an evaluation hierarchy model and hire domain experts, which has high time and economic cost.

Table 7: RTRM method.

RTRM method	Descriptions
Ease of use	It needs not only the trust relationship of the services directly associated with the tested service but also the information relationship of the indirectly associated services and the trust relationship between the indirectly associated services. The information relationship data are not easy to obtain, and the number of indirect recommenders is not easy to determine; unable to recommend new services for scenes where the complete recommender cannot be obtained. Therefore, the stability of the result is not high.
Objectivity	The objectivity of this method is mainly reflected in the input data. The trust relationship and recommendation trust relationship are calculated from the log data. Therefore, it has strong objectivity.
Versatility	This method is only available for scenarios where it is easy to obtain the directly and indirectly related service trust relationship of the tested composition. Therefore, its universality is not strong.
Functionality	This method has a single function and can only obtain the trustworthiness results of one service pair at a time.
Costs	The time cost of this method is large in the early stage. When the trust relationship and trust value between each service are obtained, the cost is reduced.

Table 8: Our method.

Our method	Descriptions
Ease of use	The state data of this method are easy to obtain, the influencing factors need to be analyzed in detail, and the rationality of the state transition matrix needs to be verified.
Objectivity	The input data of this method are the current running state of the service, which can be obtained from log data or test data, with strong objectivity.
Versatility	This method is applicable to the evolution scenario with stable operation environment factors, and has low universality.
Functionality	This method has a single function and can only judge the relationship with the trustworthiness of a given service composition.
Costs	The cost of this method is low, but its time complexity is high.

relationship between directly related services but also calculates the recommendation trust relationship of other services interacting with them. Theoretically, this calculation method is relatively complete in the trustworthiness calculation of the combination, and its ease of use, objectivity, universality, functionality, and costs are shown in Table 7.

Our method uses the influence degree of risk influencing factors on service pair as the trustworthiness of service pair. It is applicable to the scenario with stable context, that is, the scenario with stable probability of risk influencing factors and loss degree. Its ease of use, objectivity, versatility, functionality, and costs are shown in Table 8.

The overall comparison results of the three methods in five aspects are shown in Table 9 and Figure 6.

Through simulation experiments and method comparison, it can be seen that the proposed method has obvious advantages over the traditional methods in ease of use, objectivity, and cost consumption, and is insufficient in generality and function.

In terms of functionality, the measurement method proposed in this paper can only judge the trustworthiness of the established service composition but cannot determine the reasons for the low recognition and cannot point out the problems in the trustworthiness concerns of the service composition. Therefore, it is not convenient for users or developers to make specific adjustments for trustworthiness. To solve these problems, it needs to be combined with SaaS service independent trustworthiness measurement method. However, in the cloud computing environment, for users, users pay more attention to the overall performance of the software system in their application scenarios but are not sensitive to the specific situation inside the system or module. In the process of iterative and evolutionary development, users can test SaaS services with high independent trustworthiness one by one through replacement and trial, and can select SaaS services that can truly meet the improvement of the overall trustworthiness of the system at low cost. Therefore, the method proposed in this paper is

Table 9: Comparison results.

	Ease of use	Objectivity	Versatility	Functionality	Costs
Our method	High	High	Low	Low	Low
AHP	Medium	Low	High	High	High
RTRM	Low	Medium	Medium	Medium	Medium

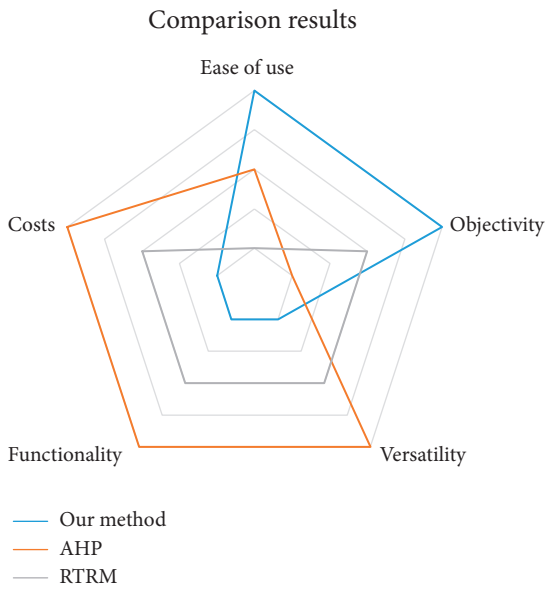

Figure 6: Comparison results.

highly targeted and solves the problem of the overall trustworthiness of the software system that users are most concerned about in specific scenarios. However, the function is relatively simple and needs to be combined with the SaaS service independent trustworthiness calculation method.

In terms of generality, compared with the other two methods, the method proposed in this paper only aims at the problem of SaaS service trustworthiness measurement, and is less applicable than other methods in terms of its application scenarios. Even in the field of software services, this method is more suitable for cloud computing environment, SaaS service environment with more high independent trustworthiness and more operation and evaluation data. However, with the further development of cloud computing, more and more users choose SaaS services, and the generality of the measurement method in this paper will be improved.

6. Conclusion

SaaS services with low independent trustworthiness can never build a service system with high trustworthiness. However, SaaS services with high independent trustworthiness cannot always build a high trusted service system, either. There are many other problems that will affect the overall trustworthiness. In the cloud computing environment, for users in specific scenarios, users pay more attention to the matching degree of the SaaS service system to their needs, rather than the trustworthiness of a single SaaS service. Therefore, it is far from enough to focus on the independent trustworthiness of a SaaS service. After a SaaS service enters the software system, the improvement of the combinatorial trustworthiness brought to the service system has more theoretical and practical significance. This paper takes the software service system under the cloud computing environment as the research object and takes the combinatorial trustworthiness measurement of the software service system as the research objective. A measurement method based on Markov theory and cosine similarity has been proposed. Through the simulation experiment and experimental analysis, the feasibility and effectiveness of the proposed method were verified. Through a comparison of different methods, the advantages and limitations of the proposed method were demonstrated.

The research result of this paper is the downstream research content of independent trustworthiness measurement method. The method proposed in this paper provides theoretical and technical support for solving the most concerned problems of users in the cloud computing environment. This method is more suitable for scenarios with a large number of services and service data. For a new cloud service, it is difficult to guarantee the stability of its trustworthiness results due to its lack of state data in its context. Due to the above limitations, further research on data acquisition of new services is needed in the future.

Data Availability

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

Ethical Approval

This article does not contain any studies with human participants or animals performed by any of the authors.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Authors' Contributions

Material preparation, data collection, and analysis were performed by Xiaohui Jia, Rong Jiang, Yuanyuan He, and Ming Yang. The first draft of the manuscript was written by Tilei Gao. All authors commented on the previous versions of the manuscript, contributed to the study conception and design, and read and approved the final manuscript.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Nos. 71972165 and 61763048), Science and Technology Foundation of Yunnan Province (No. 202001AS070031), Yunnan Fundamental Research Projects (Nos. 202201AT070142 and 202101AT070211), and Talent Introduction Projects of Yunnan University of Finance and Economics (No. 2021D16).

References

[1] S. Ali, N. Ullah, M. F. Abrar, Z. Yang, and J. Huang, "Fuzzy multicriteria decision-making approach for measuring the possibility of cloud adoption for software testing," *Scientific Programming*, vol. 2020, no. 1, 24 pages, Article ID 6597316, 2020.

[2] S. Ali, S. Baseer, I. A. Abbasi, B. Alouffi, W. Alosaimi, and J. Huang, "Analyzing the interactions among factors affecting cloud adoption for software testing: a two-stage ISM-ANN approach," *Soft Computing*, vol. 26, no. 16, pp. 8047–8075, 2022.

[3] Flexera, *State of the Cloud Report*, Flexera, Itasca, IL, USA, 2020.

[4] M. Raza, F. K. Hussain, O. K. Hussain, Z. U. Rehman, and M. Zhao, "Imputing sentiment intensity for SaaS service quality aspects using T-nearest neighbors with correlation-weighted Euclidean distance," *Knowledge and Information Systems*, vol. 63, no. 9, pp. 2541–2584, 2021.

[5] T. Gao, *Research on saas selection and evolution method based on trustworthiness measurement*, Yunnan University, Kunming, China, Ph.D, 2020.

[6] S. Ali and N. Ullah, "Critical influential factors for software testing-as-a-service adoption: preliminary findings from systematic literature review," in *Proceedings of the 2019 International Conference on Electrical, Communication, and Computer Engineering (ICECCE)*, pp. 1–6, Swat, Pakistan, July 2019.

[7] L. Li, Z. Chen, Y. Li, and J. Li, "Overview of software fault localization technology," *Computer Measurement & Control*, vol. 27, no. 5, pp. 1–4+47, 2019.

[8] H. Hu, D. Liu, and S. Wang, "Web ontology language OWL," *Computer Engineering and Design*, vol. 30, no. 12, pp. 1-2+47, 2004.

[9] H. Lin and Q. He, "Research on the model for video surveillance system based on web service," *Computer Application System*, vol. 33, no. 2, pp. 22–25, 2006.

[10] H. Chen, J. Wang, and W. Dong, "Highly trusted software engineering technology," *Journal of Electronics*, vol. 31, no. 12A, pp. 1933–1938, 2003.

[11] X. He, J. Tian, and F. Liu, "Survey on trusted cloud platform technology," *Journal on Communications*, vol. 40, no. 2, pp. 154–163, 2019.

[12] M. Howard and S. Lipner, *The Secure Development LIfe-Cycle*, Microsoft Press, Redmond, WA, USA, 2006.

[13] M. Howard and D. E. Leblanc, *Writing Secure Code*, Microsoft Press, Redmond, WA, USA, 2002.

[14] B. Littlewood and L. Strigini, "Software reliability and dependability: a roadmap," in *Proceedings of the The Conference on the Future of Software Engineering*, pp. 175–188, ACM Press, Limerick, Ireland, June 2000.

[15] H. W. Schmidt, "Trustworthy components-compositionality and prediction," *Journal of Systems and Software*, vol. 65, no. 3, pp. 215–225, 2003.

[16] P. Neumann, "Principled assuredly trustworthy composable architectures," Cdrl A0001 Final Report, Computer Science Laboratory, Menlo Park, CA USA, 2004.

[17] NSS2, "Software 2015: a national software strategy to ensure U.S. Security and competitiveness," 2005, http://www.cnsoftware.orgg/nss2report.

[18] L. Bernstein, *Trustworthy Systems through Quantitative Software Engineering*, Wiley-IEEE Computer Society Press, New York, NY, USA, 2005.

[19] A. Miller and J. D. Mclean, "COMPSAC panel session on trustworthy computing," in *Proceedings of the 30th Annual International Computer Software and Applications Conference (COMPSAC 2006)*, IEEE Computer Society, Chicago, IL, USA, September 2006.

[20] H. Wang, Y. Tang, and G. Yin, "Trust mechanism of Internet software," *Chinese Science: Technical Science*, vol. 36, no. 10, pp. 1156–1169, 2006.

[21] V. O. Safonov, *Using Aspect-Oriented Programming for Trustworthy Software Development*, Wiley & Sons, Hoboken, NJ, USA, 2007.

[22] Y. Yang, Q. Wang, and M. Li, "Process trustworthiness as a capability indicator for measuring and improving software trustworthiness," in *Proceedings of the International Conference on Software Process 2009*, vol. 5543, pp. 389–401, Vancouver, Canada, May 2009.

[23] D. Feng, Y. Qin, W. Feng, and J. Shao, "The theory and practice in the evolution of trusted computing," *Chinese Science Bulletin*, vol. 59, no. 32, pp. 4173–4189, 2014.

[24] X. Yang, P. Luo, and G. Jabeen, "A measurable SocialToTech software trust framework," *IOP Conference Series: Earth and Environmental Science*, vol. 234, Article ID 012073, 2019.

[25] S. Ding, F. Lu, S. Yang, and C. Xia, "A requirement-driven software trustworthiness evaluation and evolution model," *Journal of Computer Research and Development*, vol. 48, no. 4, pp. 647–655, 2011.

[26] S. Yang, S. Ding, and C. Fu, "Software trustworthiness evaluation model considering information source correlation," *Chinese Management Science*, vol. 17, no. 6, pp. 163–169, 2009.

[27] L. Gu, Y. Guo, H. Wang, Y. Z. Zou, B. Xie, and W. Z. Shao, "Runtime software trustworthiness evidence collection mechanism based on TPM," *Journal of Software*, vol. 21, no. 2, pp. 373–387, 2010.

[28] Y. Zhang, Y. Yuan, X. Liu, and M. Sun, "Evaluation method of software service trustworthiness of E-commerce website," *Computer Application Research*, vol. 37, no. s1, pp. 244–246+263, 2020.

[29] L. Shao, J. Zhang, Y. Wei, J. Zhao, B. Xie, and H. Mei, "Personalized QoS prediction forWeb services via collaborative filtering," in *Proceedings of the IEEE International Conference on Web Services (ICWS 2007)*, Salt Lake City, UT, USA, July 2007.

[30] W. Rong, K. Liu, and L. Liang, "Personalized Web Service Ranking via User Group Combining Association Rule," in *Proceedings of the 2009 IEEE International Conference on Web Services*, Los Angeles, CA, USA, July 2009.

[31] Y. Jiang, J. Liu, M. Tang, and x. Liu, "An Effective Web Service Recommendation Method Based on Personalized Collaborative Filtering," in *Proceedings of the 2011 IEEE International Conference on Web Services*, Washington, DC, USA, April 2011.

[32] L. Yao, Q. Z. Sheng, A. Segev, and J. Yu, "Recommending web services via combining collaborative filtering with content-based features," in *Proceedings of the 2013 IEEE 20th International Conference on Web Services*SantaClara, CA, USA, June-July 2013.

[33] Z. Liu, Z. Zhang, Y. Hai, S. Guo, and L. Yonli, "Research on active service recommendation method based on demand prediction," *Computer Engineering*, vol. 46, pp. 1–9, 2018.

[34] Y. Zhang, C. Yin, Z. Lu, D. Yan, M. Qiu, and Q. Tang, "Recurrent tensor factorization for time-aware service recommendation," *Applied Soft Computing*, vol. 85, no. 6, Article ID 105762, 2019.

[35] E. Çano, "Cloud-based Recommendation Systems: Applications and Solutions," *Computer Science*, vol. 2019, 2019.

[36] S. Ali and H. Li, "Moving Software Testing to the Cloud: An Adoption Assessment Model Based on Fuzzy Multi-Attribute Decision Making Algorithm," in *Proceedings of the 2019 IEEE 6th International Conference on Industrial Engineering and Applications*, pp. 382–386, Tokyo, Japan, April 2019.

[37] X. Hu, R. Jiang, M. Shi, and J. Shang, "A privacy protection model for health care big data based on trust evaluation access control in cloud service environment," *Journal of Intelligent and Fuzzy Systems*, vol. 38, no. 3, pp. 3167–3178, 2020.

[38] A. Mohsenzadeh, H. Motameni, and M. J. Er, "Retraction note to: a new trust evaluation algorithm between cloud entities based on fuzzy mathematics," *International Journal of Fuzzy Systems*, vol. 21, no. 6, p. 1988, 2019.

[39] D. Wang and Q. Wang, "Trustworthiness evidence supporting evaluation of software process trustworthiness," *Journal of Software*, vol. 29, no. 11, pp. 178–200, 2018.

[40] W. Liu, L. Zou, Y. Ba, G. Li, and Z. Zhang, "Association aware cloud service trust model based on convex function evidence theory," *Computer engineering and Science*, vol. 41, no. 1, pp. 47–55, 2019.

[41] R. C. Ratnayake and K. Antosz, "Development of a risk matrix and extending the risk-based maintenance analysis with fuzzy logic," *Procedia Engineering*, vol. 182, pp. 602–610, 2017.

[42] S. Albery, D. Borys, and S. Tepe, "Advantages for risk assessment: evaluating learnings from question sets inspired by the FRAM and the risk matrix in a manufacturing environment," *Safety Science*, vol. 89, pp. 180–189, 2016.

[43] W. Shang and X. Xing, "ICS software trust measurement method based on dynamic length trust chain," *Scientific Programming*, vol. 2021, no. 5, 11 pages, Article ID 691696, 2021.

[44] Z. Yang, C. Yin, Z. Fang, and N. Zhao, "Trust chain model and trustworthiness analysis in software systems," in *Proceedings of the 5th International Conference on Computer and Communication Systems*, Shanghai, China, May 2020.

[45] U. Jayasinghe, G. M. Lee, A. MacDermott, and W. S. Rhee, "TrustChain: a privacy preserving blockchain with edge computing," *Wireless, Communications and Mobile Computing*, vol. 2019, Article ID 2014697, 17 pages, 2019.

[46] Y. Song, Y. Wang, and D. Jin, "A bayesian approach based on bayes minimum risk decision for reliability assessment of web service composition," *Future Internet*, vol. 12, no. 12, p. 221, 2020.

[47] P. Chen, X. Wang, and D. Dang, "Construction of model based on petri net and reliability analysis based on bayes net of web service transaction," *Journal on Communications*, vol. 39, no. s1, pp. 99–104, 2018.

[48] G. Tilei, L. Tong, Y. Ming, and J. Rong, "Research on a trustworthiness measurement method of cloud service construction processes based on information entropy," *Entropy*, vol. 21, no. 5, pp. 462–482, 2019.

[49] T. Gao, T. Li, R. Jiang, R. Duan, R. Zhu, and M. Yang, "A research about trustworthiness metric method of SaaS services based on AHP," *Lecture Notes in Computer Science*, vol. 11063, pp. 207–218, 2018.

[50] Y. Wang, J. Lv, F. Xu, and L. Zhang, "A trust measurement and evolution model for internetware," *Journal of Software*, vol. 17, no. 4, pp. 682–690, 2006.