Hindawi
Mathematical Problems in Engineering
Volume 2023, Article ID 2652066, 16 pages
https://doi.org/10.1155/2023/2652066

Research Article

The Weight Allocation Method of Component Based on the Mutual Influence

Zhixi Hu,[1] **Yanfang Ma**,[1,2] **and Xiaotong Gao**[2]

[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*

Correspondence should be addressed to Yanfang Ma; clmyf@163.com

Received 28 February 2023; Revised 21 July 2023; Accepted 7 August 2023; Published 24 August 2023

Academic Editor: Toqeer Mahmood

Components play a pivotal role in component-based software, with certain components being crucial in realizing the software's overall functionality. The importance of component is usually characterized using the weight value of component. How to get the weight value is a problem that needs to be studied. Traditionally, the weight value can be denoted by an expert's evaluation. However, the importance of components can vary depending on their interactions within different combination structures. Therefore, it becomes necessary to consider the mutual influence resulting from component interactions when determining component weights, in addition to expert evaluations. In this study, we first establish an influence value model that captures the interactions between components within various combinations. Subsequently, we propose a comprehensive weight allocation method for components by integrating expert evaluations with the influence values obtained from the component interactions, using an evidence-based information fusion approach. Finally, we validate the effectiveness of our method by implementing a scenic spot ticket purchase system. This approach is more reasonable than the single method and can provide software developers with a more comprehensive analysis of component importance, enabling them to make informed decisions.

1. Introduction

With the increasing complex of software systems, there is a significant need for software adaptability. Early software systems faced challenges in effectively adapting to dynamic environments and meeting user needs [1–4]. Consequently, there has been a significant increase in research interest in component-based software, which provides wide applicability and high reusability. The concept of component-based software engineering was first introduced by McIlroy [5] in 1968, advocating the division of complete and intricate software systems into smaller components to enhance system comprehensibility. In 2002, Szyperski et al. [6] explored the definition, current state, and research trends of components. However, the discussion at that time was limited by the immaturity of component thinking. Over the years, extensive research has been conducted in the field of component-based software engineering (CBSE), which has yielded a top-down approach to software architecture decomposition [7]. The development of CBSE has provided a systematic framework for organizing software systems into manageable components, enhancing adaptability and facilitating comprehension. By leveraging the benefits of component-based software engineering, software systems can better adapt to changing environments and meet the diverse needs of users. As software systems grow in scale to meet increasing user demands, the adoption of component-based software development methods becomes more prevalent [8]. A component-based software system comprises a collection of individual components, each with its internal structure and the ability to interact with other components in various combinations to fulfill software functionalities [9]. In complex systems, specific components play a vital role in ensuring the smooth operation of the software system [10, 11]. Therefore, it becomes essential to quantify the importance of these components.

Bertrand proposed the "ABCDE" quality classification model, which categorizes components into five classes, providing a basis for their classification and importance [12]. However, this classification only focuses on the class of components and does not distinguish the importance of the single component in great detail. To convey the importance of a component, it is necessary to assign weight to the component. Commonly, the analytic hierarchy process (AHP) [13] and entropy methods [14] are employed for weight allocation. In the classical AHP model, the importance between elements is denoted by a pairwise comparison matrix. The AHP method is subjective and heavily relies on expert's evaluations, leading to uncertain information. Wang et al. [15] used the AHP method for weight allocation in software components and developed a multivalued model for allocating software component development costs based on different software system structures. Zhou et al. [16] combined component weights with user feedback to establish trustworthiness measurement models based on component construction, using the Dempster–Shafer (D–S) evidence theory to fuse fuzzy and uncertain information obtained through the AHP method. Lozano-Tello and Gomez-Perez [17] utilized AHP to facilitate multicriteria decision-making for reused software components, assisting software engineers in evaluating suitable components for adoption. Comparing AHP with the weighted scoring method (WSM) and the hybrid knowledge-based system (HKBS), Jadhav and Sonar [18] found that HKBS performed more better than AHP and WSM in evaluating and selecting software packages. To simplify the AHP process in component quality evaluation and reduce subjectivity, Ren [19] proposed a software component quality evaluation model based on group decision-making. To characterize the uncertain information in more detail, D-AHP has been proposed, which considers the impact of information credibility [20]. To tackle the challenges arising from imprecise and incomplete information, van Laarhoven and Pedrycz [21] introduced a new method called fuzzy analytic hierarchy process (FAHP), which combines fuzzy set theory with AHP. FAHP builds upon AHP, aiming to align the judgment results with human thinking and capture the complexities of real-world scenarios. It offers significant advantages over traditional AHP in dealing with fuzzy and difficult-to-quantify problems, making it suitable for various nondeterministic situations where clear and systematic outcomes are desired. In FAHP, the use of fuzzy numbers for pairwise comparison accounts for the high degree of fuzziness and uncertainty inherent in human judgments in real-world contexts [22]. Recently, scholars have extended the application of fuzzy sets to T-spherical fuzzy sets and employed T-spherical fuzzy matrices to express expert's evaluations of risk factors. They obtained the weights of risk factors from risk assessment matrices [23, 24]. However, as the AHP, D-AHP, FAHP methods are prone to subjective bias and may not fully capture the internal information that reflects the importance, relying solely on expert evaluation may not be sufficient for weight allocation. Consequently, several objective weight allocation methods have been developed.

The entropy method is a widely used approach for weight allocation [14]. It is an objective method that calculates the weight based on the amount of information carried by the data, yielding a relatively objective index weight. Recently, the entropy method has been extended to the cross-entropy of discrete Z-numbers method [25], which describes the ambiguity and reliability of the information. However, it primarily applies to multiresponse systems. Certain software systems do not exist multiple response characteristics. For instance, some computational software simply requires the input to generate the output, without requiring user responses during the computation process. Hence, the entropy method is not suitable for weight allocation in general component-based software. Kim et al. [26] addressed the design optimization problem objectively and proposed the feasible improved weight allocation (FIWA) method. In FIWA, the significant input variables that violate constraints will obtained the more weight to achieve a feasible design. This method is more appropriate for multiresponse systems with multiple input variables. But, it may not be applicable to software systems that lack multiple input variables. CRiteria Importance Through Intercriteria Correlation (CRITIC) method is another objective method for weight allocation [27], which is based on the quantification of the contrast intensity and the conflicting character of the evaluation criteria. In the context of component-based software, different components have diverse functions and characteristics, and it is difficult to obtain the data of the conflict between components. Huang et al. [28] classified components into critical and noncritical components based on their varying importance and proposed a credibility measurement model considering different component combination modes. However, this model does not present a specific method for calculating the importance of weight.

To overcome the limitations of subjective and objective methods, combination weight allocation methods have been studied. For example, the entropy-AHP weighted method [29] was developed to compensate the disadvantage of single method and to boost their advantages. The software systems include many attributes, and the conflict often happened between these attributes. To allocate weights to attributes more reasonably, Gao et al. [30] proposed a novel weight allocation method for software attributes by combining the FAHP method with the CRITIC method. This approach considers the conflict between attributes by using the CRITIC method. In the context of component-based software systems, each component comprises multiple attributes, and the combination of components can influence trustworthiness. Recently, Ma et al. [31] employed the FAHP method and the correlation between trustworthy attributes and defects to determine the attribute weights for components.

The methods discussed above can provide a mean to analyze the weight or importance of a specific object. However, when dealing with complex component-based software systems, it is insufficient to determine the importance of components based solely on a single method, such as expert evaluation or component attributes [32–34]. The reason mainly arises there exists internal interaction information that objectively reflects the importance of components. For example, in a train ticket purchasing system, the "ticket" component must transmit data to the "payment" component. If the "ticket" component fails to execute successfully, then the performance of "payment" will be affected. Therefore, the "ticket" component is more important than the

Table 1: Integer scales meaning table.

Scale	Definition
1	C_i is as important as C_j
3	C_i is important than C_j
5	C_i is more important than C_j
7	C_i is very important than C_j
9	C_i is absolutely more important than C_j
2, 4, 6, and 8	The scale value corresponding to the intermediate state between judgments
Reciprocal	If C_i is important than C_j, and the judgment value is $a_{ij} = 3$, then $a_{ji} = 1/3$.

"payment," which also indicates that there exists a mutual influence on the importance of components. Therefore, it is essential to comprehensively consider expert evaluations and the mutual influence between components to evaluate component importance. Moreover, different combinations of components lead to varying interactions and levels of mutual influence. Therefore, in this paper, we establish an influence value model based on a directed diagram model and propose an objective weight allocation method. Additionally, a comprehensive weight allocation approach for components is proposed by combining the FAHP method with the objective weight allocation method.

The structure of this paper is as follows: Section 2 provides an overview of relevant work, including the FAHP method and evidence theory fusion method. In Section 3, we construct the influence value model based on combination structures. Section 4 presents the objective weight allocation method, considering the mutual influence between components. In Section 5, we propose a comprehensive weight allocation method for components. A case study is presented in Section 6. Finally, Section 7 concludes the paper.

2. Related Work

2.1. The Weight Allocation of Components Based on FAHP. Due to the importance of each component in the software system being different, experts need to compare each other according to the different importance of the components and give a relatively fuzzy and nondeterministic evaluation. Then, the weight of each component is determined by integrating the opinions of multiple experts. Additionally, the behavior of components can be influenced by various factors, such as the environment and hardware. Generally, experts cannot thoroughly consider all possible situations, leading to uncertainty and incomplete information. Thus, it is more appropriate to allocate the weight of components based on expert evaluations using the simplified FAHP method. The specific steps are given as follows, and detailed contents can be found in [21].

Step 1. *Suppose there are n components, and g experts compare each component pairwise to provide the corresponding fuzzy importance evaluation. These fuzzy evaluations are then converted into triangular fuzzy numbers, denoted as* $h_{ij}^1 = \left(l_{ij}^1, m_{ij}^1, u_{ij}^1\right), \ldots, h_{ij}^g = \left(l_{ij}^g, m_{ij}^g, u_{ij}^g\right), \quad i, j = 1, 2, \ldots, n,$ *where* l_{ij}^k, m_{ij}^k, *and* $u_{ij}^k (1 \le k \le g)$ *represent the lowest possible value, the most possible value, and the highest possible value of the relative importance of component* C_i *and* C_j *evaluated by the kth expert, respectively. These values are determined based on the integer scale from 1 to 9 adopted in the AHP method, as shown in Table 1. Therefore, the fuzzy judgment matrix* $H^k = \left[h_{ij}^k\right]_{n\times n}$ *is constructed based on the fuzzy evaluation given by the kth expert as follows, where n is the number of components:*

$$H^k = \begin{bmatrix} \left(l_{11}^k, m_{11}^k, u_{11}^k\right) & \cdots & \left(l_{n1}^k, m_{n1}^k, u_{n1}^k\right) \\ \vdots & \ddots & \vdots \\ \left(l_{1n}^k, m_{1n}^k, u_{1n}^k\right) & \cdots & \left(l_{nn}^k, m_{nn}^k, u_{nn}^k\right) \end{bmatrix}. \quad (1)$$

Step 2. *The fuzzy judgment matrices provided by g experts are fused to obtain a comprehensive fuzzy judgment matrix* H^{com}, *where the elements in the matrix are represented as comprehensive triangular fuzzy numbers* $h_{ij}^{\text{com}} = \left(l_{ij}^{\text{com}}, m_{ij}^{\text{com}}, u_{ij}^{\text{com}}\right)$. *The specific calculation process is given as follows:*

$$h_{ij}^{\text{com}} = \left(\frac{l_{ij}^1 + \cdots + l_{ij}^g}{g}, \frac{m_{ij}^1 + \cdots + m_{ij}^g}{g}, \frac{u_{ij}^1 + \cdots + u_{ij}^g}{g}\right). \quad (2)$$

Then, the comprehensive fuzzy judgment matrix H^{com} *is given as follows:*

$$H^{\text{com}} = \begin{bmatrix} \left(l_{11}^{\text{com}}, m_{11}^{\text{com}}, u_{11}^{\text{com}}\right) & \cdots & \left(l_{n1}^{\text{com}}, m_{n1}^{\text{com}}, u_{n1}^{\text{com}}\right) \\ \vdots & \ddots & \vdots \\ \left(l_{1n}^{\text{com}}, m_{1n}^{\text{com}}, u_{1n}^{\text{com}}\right) & \cdots & \left(l_{nn}^{\text{com}}, m_{nn}^{\text{com}}, u_{nn}^{\text{com}}\right) \end{bmatrix}. \quad (3)$$

Step 3. *Calculating the fuzzy weight* $\tilde{\alpha}_i$ *of the ith component:*

$$\begin{aligned} \tilde{\alpha}_i &= \left(\tilde{\alpha_i^l}, \tilde{\alpha_i^m}, \tilde{\alpha_i^u}\right) \\ &= \left(\frac{\sum_{j=1}^n l_{ij}^{\text{com}}}{\sum_{i=1}^n \sum_{j=1}^n l_{ij}^{\text{com}}}, \frac{\sum_{j=1}^n m_{ij}^{\text{com}}}{\sum_{i=1}^n \sum_{j=1}^n m_{ij}^{\text{com}}}, \frac{\sum_{j=1}^n u_{ij}^{\text{com}}}{\sum_{i=1}^n \sum_{j=1}^n u_{ij}^{\text{com}}}\right). \end{aligned} \quad (4)$$

Step 4. *According to the method proposed by Kaufmann and Gupta [35], the fuzzy weight obtained in Step 3 is defuzzified and the subjective weight of each component is calculated. The specific calculation process is given as follows:*

$$\alpha_i = \frac{\tilde{\alpha}_i^l + 2\tilde{\alpha}_i^m + \tilde{\alpha}_i^u}{4}, \tag{5}$$

where α_i expresses the subjective weight of the ith component, and the subjective weight of all components can be written as a n-dimensional vector, $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)$.

2.2. Evidence Theory Fusion Method. Evidence information fusion methods have distinct advantages in handling uncertain information and are commonly used to combine information from multiple independent sources [36, 37]. A recognition frame Θ is usually a nonempty finite set, where a subset of the recognition frame Θ is called a proposition Z. $m(Z)$ is the basic probability assignment function on the recognition frame Θ, which represents the probability that the evidence information supports the occurrence of proposition Z. Suppose that w independent evidences on the same recognition frame Θ have basic probability assignment functions of $m_1, \ldots, m_w$, and the D–S evidence fusion rule for fusing w evidences is defined as:

$$\begin{cases} (m_1 \oplus \cdots \oplus m_w)(Z_e) = \dfrac{m_1(Z_e) \cdots m_w(Z_e)}{1-K} \\ K = \sum\limits_{\bigcap_{e=1}^{w} Z_e = \emptyset} m_1(Z_1) \cdots m_w(Z_w), 1 \le e \le w, \end{cases} \tag{6}$$

where $1-K$ is a normalized coefficient, it is used to avoid assigning nonzero probabilities to the null set $\varnothing$, and the meaning of $K \in [0,1)$ is the conflict between evidence information. The closer the value of K is to 1, the greater the conflict between evidence information [38].

3. The Influence Value Model of Components Based on the Combination Structure

As the weight allocation of components based on FAHP mainly relies on expert evaluation, it may be too subjective to fully reflect the importance. In complex software systems, there is also objective information that can reflect the importance of components. During the component-based system, the component often interacts with other components through the various combination structures, such as sequence, branch, and more. If a critical function of a system can be realized by a component interacting with other components, then the corresponding component plays a pivotal role, and the weight of this component allocated is increased accordingly. At the same time, the weight of the interacted component should not be disregarded. Hence, the interaction information between components should be taken into consideration when determining the component's importance. Different combinations involve various interaction actions. For instance, in the sequence structure, the execution of a later component relies on the execution data of a previous component. This implies that the previous component influences the later component. The degree of influence between components also determines the component's importance. During this section, we will try to establish the influence value model based on the combination structure by using the directed diagram.

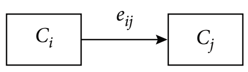

Figure 1: Component influence diagram.

Generally, component-based software consists of numerous components. There exists a mutual influence between components when the components are linked by some combination structure. Similar to the approach in [39], we consider the component as the node and the links between components as the directed edges. If the component C_i influences component C_j, then there exists a directed edge from component C_i to component C_j. In [39], the author uses unweighted directed edge to express the influence between components. Suppose there are n components, the influence matrix was established in [39]. If the component C_i influences component C_j, then the element in the i-th row and j-th column of the matrix is 1. However, component C_i may not only affect component C_j, but also have an impact on several other components. Therefore, the importance of component C_i should be comprehensively determined by considering all the outedges that originate from it and point to other components. To quantify the degree of influence between components conveniently and understand the influence relationship between components intuitively, we introduce an influence value e_{ij} for every directed edge of the component, as shown in Figure 1.

The value of e_{ij} belongs to the scale range proposed in [40], which is defined as $e = \{0, 1, 2, 3\}$. Here, elements 0, 1, 2, and 3 represent no impact, small impact, medium impact, and great impact, respectively.

The links between components are typically established through various combinations of components. The influence value e_{ij} should be analyzed according to the different combinations of components. Next, we will introduce seven basic component combinations that are frequently used in software system architecture research: sequence, branch, parallel, interrupt fault-tolerant, noninterrupt fault-tolerant, callback, and loop structure. According to the characterization of the combination structure, the influence value between components will be established.

3.1. Sequential Structure. Suppose the software system S is composed of n components through a sequential structure. According to the characterization of sequence structure, the later component can continue to be executed only when the previous component has been successfully executed. Therefore, the successful execution of the previous component has a significant impact on the execution of the later component. Moreover, the failure of any component in the sequential

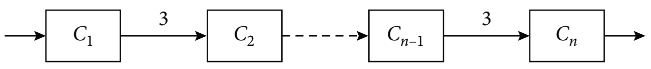

Figure 2: Sequence structure diagram with influence values.

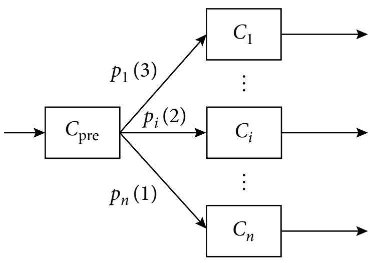

Figure 3: Branch structure diagram with influence values.

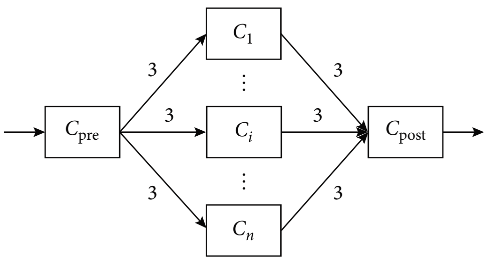

Figure 4: Parallel structure diagram with influence values.

structure will lead to the collapse of the whole software system, so that the corresponding task objectives cannot be performed. Therefore, each component of the software system in the sequential structure is critical and cannot be overlooked, and thus the influence values among each component in the sequential structure will be attached with 3 to express the importance. This is illustrated in Figure 2.

3.2. Branch Structure. Let the software system S be composed of n components through branch structure. Each component will be executed with a certain probability p_i $(\sum_{i=1}^{n} p_i = 1)$, and the influence value among each component under the branch structure will be determined by the probability p_i. Therefore, the component with the highest probability p_1 is assigned an influence value of 3, the component with the lowest probability p_n is assigned an influence value of 1, and the other component influence values are assigned an influence value of 2, as shown in Figure 3.

3.3. Parallel Structure. In a parallel structure, components are executed simultaneously. When all components in the parallel structure run successfully, the entire parallel structure can run successfully. Under this structure, all components are required to run successfully, and the failure of any component will lead to subsequent processes failing. Therefore, the influence values among each component in the parallel structure will be attached with 3, as shown in Figure 4.

3.4. Interrupt Fault-Tolerant Structure. Suppose the software system S comprises n components through an interrupt fault-tolerant structure. If the main component fails due to external factors (e.g., virus attacks, changes in the experimental environment, etc.) or internal conflicts (e.g., code conflicts, etc.), the system will immediately stop the operation of the main components. Then, the software system switches to redundant backup components, allowing the system to recover from errors and ensure the normal operation of the software system. The system is considered to have failed only if both the main component and its redundant backup components have failed. In an interrupt fault-tolerant structure, only one of the n components needs to execute successfully. Therefore, the influence value of the precomponent on each component will be assigned a value of 3, and the influence value of each component on the postcomponent will be assigned a value of 1, as shown in Figure 5.

Where the solid part of the arrows represents the main operational components, while the dashed part represents the redundant backup components.

3.5. Noninterrupt Fault-Tolerant Structure. In a noninterrupt fault-tolerant structure, the same function in the software system is simultaneously performed by multiple identical components. The successfully executed component will be used as the redundant backup of the unsuccessfully executed component and will be executed at any time. Generally speaking, as long as at least $l = [n/2] + 1$ components are successfully executed, the software system is considered to have no failures. Therefore, the influence value of the precomponent on each component will be assigned a value of 3, and the influence value on each component on the postcomponent will be assigned a value of 2, as shown in Figure 6.

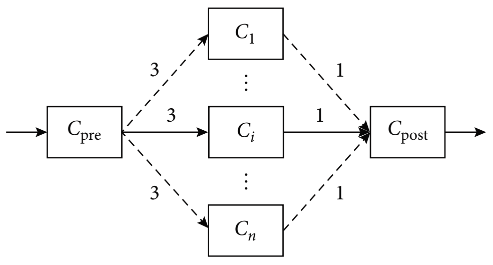

Figure 5: Interrupt fault-tolerant structure diagram with influence values.

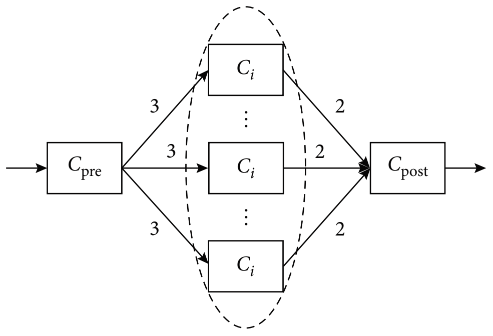

Figure 6: Noninterrupt fault-tolerant structure diagram with influence values.

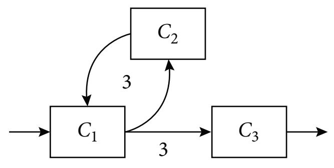

Figure 7: Callback structure diagram with influence values.

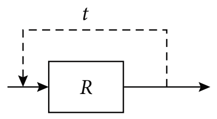

Figure 8: Loop structure diagram with influence values.

3.6. Callback Structure. When the currently running component cannot perform a specific task itself, it is possible to call other components to assist in its completion. This situation is regarded as a callback structure. In this structure, the software system can only be regarded as successfully executed when the called component is successfully executed, and the control right is called back to the postcomponent. Therefore, the influence values among each component in the callback structure will be assigned a value of 3, as shown in Figure 7.

3.7. Loop Structure. The loop structure is composed of a loop body that completes the task assigned by the software system through repeated operations. Suppose there is a loop body R. When the loop condition is satisfied, the loop is repeated t times until the loop condition is no longer satisfied, at which point the loop is exited. The internal structure of the loop body can be sequence, branch, interrupt fault-tolerant, and so on. Therefore, the influence value among each component in this structure will be determined by the structure of the loop body, and will not be described in detail here, as shown in Figure 8.

4. The Objective Weight Allocation Method of Component Based on Mutual Influence

Section 3 discusses the existence of mutual influence between components. This mutual influence information reflects the importance of the component from an objective view. Next, we will try to analyze the component importance from the objective view based on the mutual influence information.

Suppose that there are n components. Based on the mutual influence information discussed in Section 3, we can obtain the influence value between each pair of components. Therefore, the influence relationship between components can be represented as a matrix, denoted as the direct influence matrix $G = \left[g_{ij}\right]_{n\times n}$, where $g_{ij} = e_{ij}$ indicates an influence from component C_i to C_j, and the influence value is e_{ij}. If there is no effect between C_i and C_j, then $g_{ij} = 0$. Additionally, $g_{ii} = 0$ is defined. It is important to note that the influence between components can be transitive. For example, during the sequence structure, the precomponent C_i has an impact on the component C_j, and C_j has an impact on the component C_k, then C_i can indirectly influence C_k to some content, even though C_i cannot directly influence C_k. Therefore, the indirect influence between components should be considered to quantify the influence between components. To do this, the indirect influence between the components can be calculated using the self-multiplication of the direct influence matrix, based on the relevant definition of the reachability matrix. Subsequently, the comprehensive influence matrix between the components can be obtained by combining the direct influence matrix and the indirect influence matrix. Finally, the objective weight allocation of components can be determined following these specific steps:

Step 5. *Normalizing the direct influence matrix G by dividing the maximum value of the sum of the elements in the row of the matrix and the normalized influence matrix is obtained, written as G_{sta}, i.e.,*

$$G_{\text{sta}} = \frac{1}{A} G, \tag{7}$$

where $A = \max_{i\in\{1,\cdots,n\}} \sum_{j=1}^{n} g_{ij}$.

Step 6. *Adding the normalized influence matrix's power to obtain the comprehensive influence matrix, denoted as $G_t = \left[g_{ij}^t\right]_{n\times n}$, where the power exponent is from 1 to n, the formula is given as follows:*

$$G_t = G_{\text{sta}} + G_{\text{sta}}^2 + \cdots + G_{\text{sta}}^n. \tag{8}$$

Step 7. *Analyzing the comprehensive influence matrix G_t, the influence degree of the ith component C_i is defined as the sum of all elements in the ith row of matrix G_t, i.e.,*

$$\text{In}f_i = \sum_{j=1}^{n} g_{ij}^t, 1 \le i \le n. \tag{9}$$

And the influenced degree of the ith component C_i is defined as the sum of all elements in the ith column of matrix G_t, i.e.,

$$\text{In fed}_i = \sum_{j=1}^{n} g_{ji}^t, 1 \le i \le n. \tag{10}$$

Step 8. *The comprehensive influence degree of the ith component, denoted as $\text{Inf}_i^{\text{com}}$, can be obtained by adding the influence degree and the influenced degree, i.e.,*

$$\text{Inf}_i^{\text{com}} = \text{In}f_i + \text{In fed}_i, 1 \le i \le n. \tag{11}$$

Step 9. *The objective weight allocation of the ith component C_i is obtained by normalizing the comprehensive influence degree of the ith component, i.e.,*

$$\beta_i = \frac{\text{In} f_i^{\text{com}}}{\sum_{i=1}^{n} \text{in} f_i^{\text{com}}}, 1 \le i \le n, \tag{12}$$

where β_i expresses the objective weight of the ith component and the objective weight of all components can be written as a n-dimensional vector, $\beta = (\beta_1, \beta_2, \cdots, \beta_n)$.

5. The Comprehensive Weight Allocation of Components

To comprehensively consider the weight allocation of components from different perspectives, it is necessary to fuse the subjective and objective weights to obtain a comprehensive weight allocation. This approach effectively reduces subjective arbitrariness and objective ideality, making the component weight allocation results more aligned with the actual situation of software system development. The evidence theory method has distinct advantages in dealing with the fusion of uncertain information. Therefore, the evidence fusion method is used to make the final result of fusion information more realistic, which is beneficial to improve the evaluation personnel decision accuracy.

Since the allocated weight values sum up to 1, and the weight allocation of each component is independent of the others, we can define a basic probability assignment function (i.e., BPA function) to allocate the weight to each component. Based on the subjective and objective weight allocation methods proposed in the above two sections, we can represent two BPA functions as follows: $m_1(C_i) = \alpha_i$, $m_2(C_i) = \beta_i$ for every $i = 1, \cdots, n$, where $\sum_{i=1}^{n} \alpha_i = 1, \sum_{i=1}^{n} \beta_i = 1$. These satisfy the definition in D–S (in short) evidence theory, where the sum of the basic probability assignments must be 1. Therefore, we can use the D–S evidence information combination rule to fuse the subjective and objective weights, and the formula is given as follows:

$$\begin{cases} (m_1 \oplus m_2)(C_i) = \dfrac{m_1(C_i) \times m_2(C_i)}{1 - K}, \\ 1 - K = 1 - \sum_{i \ne j} m_1(C_i) \times m_2(C_j), \end{cases} \tag{13}$$

where $1 - K$ is the normalization factor and the meaning of $K \in [0, 1)$ is the conflict between evidence information. The closer the value of K is to 1, the greater the conflict between evidence information; the closer K is to 0, the smaller the conflict between evidence information.

However, the allocation method of the objective weight proposed above may lead to the objective weight allocated to a component is 0 (i.e., 0 trust conflict). The traditional DS evidence theory fusion method cannot effectively eliminate the impact of 0 trust conflict between evidence bodies. Therefore, this section adopts a fusion method proposed by Murphy [41] to eliminate the "bad value" between the evidence, which can obtain the comprehensive weight of the component based on the average reliability. The specific steps are shown as follows:

Step 10. *The two different BPA functions $m_1(C_i)$, $m_2(C_i)$ are averaged to obtain the comprehensive BPA function $m(C_i)$.*

$$m(C_i) = \frac{1}{2}[m_1(C_i) + m_2(C_i)], 1 \le i \le n. \tag{14}$$

Step 11. *The comprehensive weight of the ith component, denoted as γ_i, can be obtained by combining the comprehensive BPA function $m(C_i)$ according to the DS combination rule, i.e.,*

$$\gamma_i = \frac{m(C_i)^2}{1 - K}, 1 \le i \le n, \tag{15}$$

where the comprehensive weight of all components can be written as a n-dimensional vector, $\gamma = (\gamma_1, \gamma_2, \cdots, \gamma_n)$.

To state our method more clearly, the framework to compute the comprehensive weight is designed, as shown in Figure 9. From the framework, it is easy to understand the comprehensive weight of the component is achieved by considering the subjective and objective weight of components, which balanced the imperfection of the single method.

6. Case Study

Suppose there is the scenic spot ticket purchase system S, including login, personal authentication, ticket query, ticket booking, ticket cancellation, payment, and exit seven functions. The personal authentication function can be further divided into certificate authentication, health code authentication, and mobile phone authentication. The system consists of 12 components C_1, C_2, C_3, C_4, C_5, C_6, C_7, C_8, C_9, C_{10}, C_{11}, and C_{12}, where component C_1 performs the login function; components C_2, C_3, and C_4 perform the personal authentication function through a combination of noninterrupt fault-tolerant structure; component C_5 performs the ticket query function; component C_6 performs the ticket booking function; components C_7 and C_{11} perform the ticket cancellation function through a combination of callback structure; components C_8, C_9, and C_{10} perform the payment function through a combination of interrupt fault-tolerant structure; and component C_{12} performs the exit function. Through the proposed method of attaching influence values between components based on software architecture, the influence values attached between 12 components under this system are shown in Figure 10.

6.1. Calculating the Subjective Weights α of Components

(1) Suppose two experts have fuzzy evaluations on the 12 components in the ticketing software, and the transformed fuzzy values are chosen from the [1, 9] integer scales mentioned in Subsection 2.1. Then, the fuzzy judgment matrix H^1, H^2 given by the two experts is shown as follows:

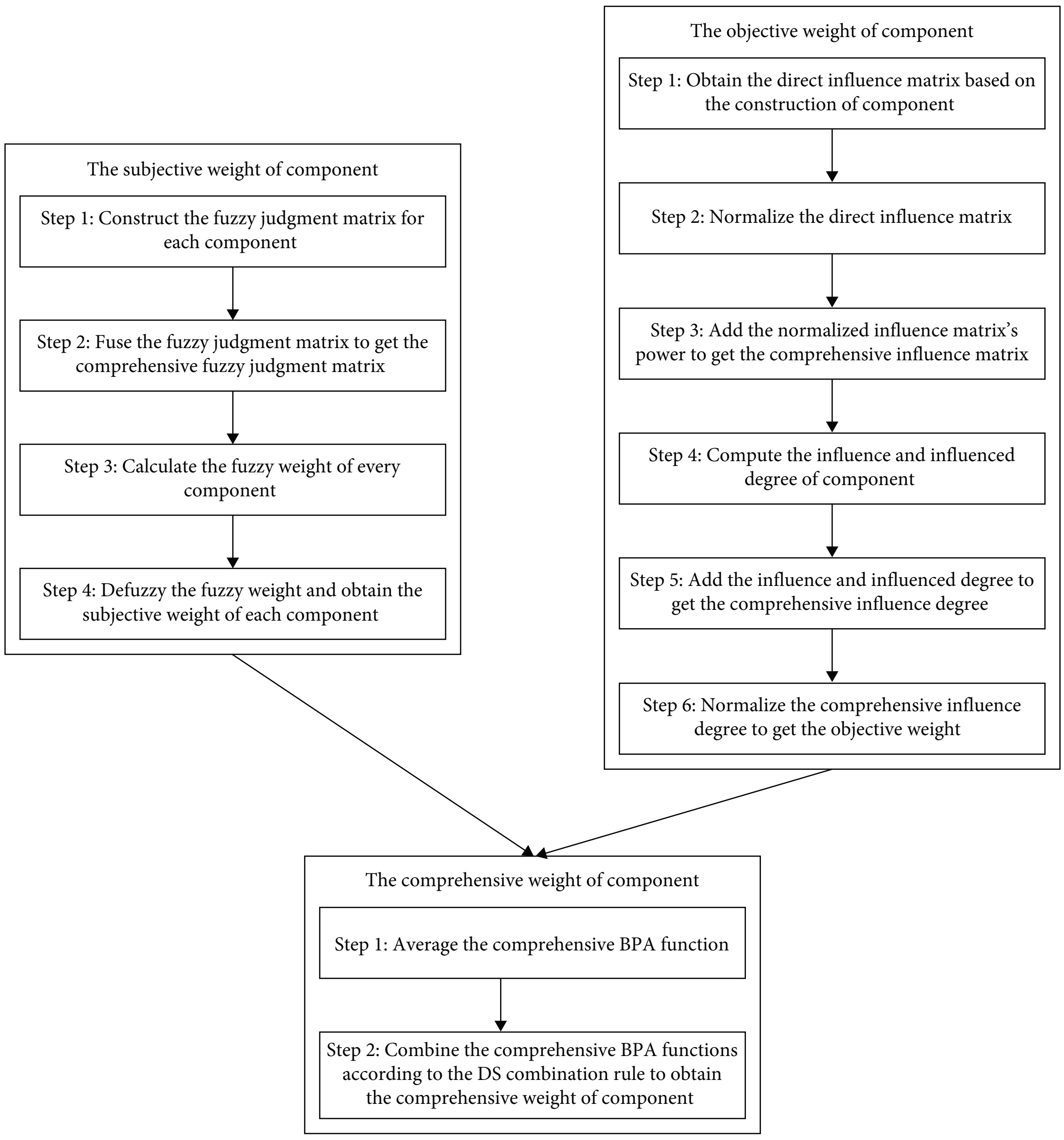
The subjective weight of component
Step 1: Construct the fuzzy judgment matrix for each component
Step 2: Fuse the fuzzy judgment matrix to get the comprehensive fuzzy judgment matrix
Step 3: Calculate the fuzzy weight of every component
Step 4: Defuzzy the fuzzy weight and obtain the subjective weight of each component
The objective weight of component
Step 1: Obtain the direct influence matrix based on the construction of component
Step 2: Normalize the direct influence matrix
Step 3: Add the normalized influence matrix's power to get the comprehensive influence matrix
Step 4: Compute the influence and influenced degree of component
Step 5: Add the influence and influenced degree to get the comprehensive influence degree
Step 6: Normalize the comprehensive influence degree to get the objective weight
The comprehensive weight of component
Step 1: Average the comprehensive BPA function
Step 2: Combine the comprehensive BPA functions according to the DS combination rule to obtain the comprehensive weight of component

FIGURE 9: The computation framework of comprehensive weight.

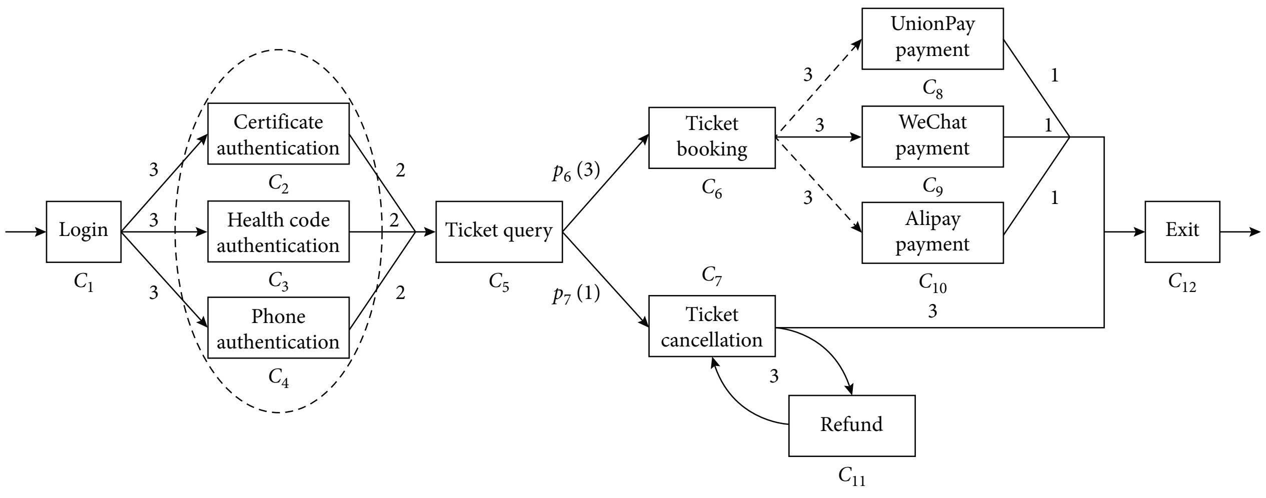

FIGURE 10: Mutual influence diagram of components in ticketing system.

$$H^1 = \left[\begin{array}{cccccccccccc}
(1,1,1) & (1/4,1/3,1/2) & (1/4,1/3,1/2) & (1/4,1/3,1/2) & (1,2,3) & (1,2,3) & (1,2,3) & (1/4,1/3,1/2) & (1/4,1/3,1/2) & (1/4,1/3,1/2) & (1,1,1) & (1,1,1) \\
(2,3,4) & (1,1,1) & (1,1,1) & (1,1,1) & (2,3,4) & (2,3,4) & (2,3,4) & (1,1,1) & (1,1,1) & (1,1,1) & (1,2,3) & (1,2,3) \\
(2,3,4) & (1,1,1) & (1,1,1) & (1,1,1) & (2,3,4) & (2,3,4) & (2,3,4) & (1,1,1) & (1,1,1) & (1,1,1) & (1,2,3) & (1,2,3) \\
(2,3,4) & (1,1,1) & (1,1,1) & (1,1,1) & (2,3,4) & (2,3,4) & (2,3,4) & (1,1,1) & (1,1,1) & (1,1,1) & (1,2,3) & (1,2,3) \\
(1/3,1/2,1) & (1/4,1/3,1/2) & (1/4,1/3,1/2) & (1/4,1/3,1/2) & (1,1,1) & (1,1,1) & (1,1,1) & (1/4,1/3,1/2) & (1/4,1/3,1/2) & (1/4,1/3,1/2) & (1/3,1/2,1) & (1/3,1/2,1) \\
(1/3,1/2,1) & (1/4,1/3,1/2) & (1/4,1/3,1/2) & (1/4,1/3,1/2) & (1,1,1) & (1,1,1) & (1,1,1) & (1/4,1/3,1/2) & (1/4,1/3,1/2) & (1/4,1/3,1/2) & (1/3,1/2,1) & (1/3,1/2,1) \\
(1/3,1/2,1) & (1/4,1/3,1/2) & (1/4,1/3,1/2) & (1/4,1/3,1/2) & (1,1,1) & (1,1,1) & (1,1,1) & (1/4,1/3,1/2) & (1/4,1/3,1/2) & (1/4,1/3,1/2) & (1/3,1/2,1) & (1/3,1/2,1) \\
(2,3,4) & (1,1,1) & (1,1,1) & (1,1,1) & (2,3,4) & (2,3,4) & (2,3,4) & (1,1,1) & (1,1,1) & (1,1,1) & (1,2,3) & (1,2,3) \\
(2,3,4) & (1,1,1) & (1,1,1) & (1,1,1) & (2,3,4) & (2,3,4) & (2,3,4) & (1,1,1) & (1,1,1) & (1,1,1) & (1,2,3) & (1,2,3) \\
(2,3,4) & (1,1,1) & (1,1,1) & (1,1,1) & (2,3,4) & (2,3,4) & (2,3,4) & (1,1,1) & (1,1,1) & (1,1,1) & (1,2,3) & (1,2,3) \\
(1,1,1) & (1/3,1/2,1) & (1/3,1/2,1) & (1/3,1/2,1) & (1,2,3) & (1,2,3) & (1,2,3) & (1/3,1/2,1) & (1/3,1/2,1) & (1/3,1/2,1) & (1,1,1) & (1,1,1) \\
(1,1,1) & (1/3,1/2,1) & (1/3,1/2,1) & (1/3,1/2,1) & (1,2,3) & (1,2,3) & (1,2,3) & (1/3,1/2,1) & (1/3,1/2,1) & (1/3,1/2,1) & (1,1,1) & (1,1,1)
\end{array}\right]. \tag{16}$$

$$H^2 = \left[\begin{array}{cccccccccccc}
(1,1,1) & (1/3,1/2,1) & (1/3,1/2,1) & (1/3,1/2,1) & (1,2,3) & (1,2,3) & (1,2,3) & (1/3,1/2,1) & (1/3,1/2,1) & (1/3,1/2,1) & (1,1,1) & (1,1,1) \\
(1,2,3) & (1,1,1) & (1,1,1) & (1,1,1) & (1,2,3) & (1,2,3) & (1,2,3) & (1,1,1) & (1,1,1) & (1,1,1) & (1,2,3) & (1,2,3) \\
(1,2,3) & (1,1,1) & (1,1,1) & (1,1,1) & (1,2,3) & (1,2,3) & (1,2,3) & (1,1,1) & (1,1,1) & (1,1,1) & (1,2,3) & (1,2,3) \\
(1,2,3) & (1,1,1) & (1,1,1) & (1,1,1) & (1,2,3) & (1,2,3) & (1,2,3) & (1,1,1) & (1,1,1) & (1,1,1) & (1,2,3) & (1,2,3) \\
(1/3,1/2,1) & (1/3,1/2,1) & (1/3,1/2,1) & (1/3,1/2,1) & (1,1,1) & (1,1,1) & (1,1,1) & (1/3,1/2,1) & (1/3,1/2,1) & (1/3,1/2,1) & (1/3,1/2,1) & (1/3,1/2,1) \\
(1/3,1/2,1) & (1/3,1/2,1) & (1/3,1/2,1) & (1/3,1/2,1) & (1,1,1) & (1,1,1) & (1,1,1) & (1/3,1/2,1) & (1/3,1/2,1) & (1/3,1/2,1) & (1/3,1/2,1) & (1/3,1/2,1) \\
(1/3,1/2,1) & (1/3,1/2,1) & (1/3,1/2,1) & (1/3,1/2,1) & (1,1,1) & (1,1,1) & (1,1,1) & (1/3,1/2,1) & (1/3,1/2,1) & (1/3,1/2,1) & (1/3,1/2,1) & (1/3,1/2,1) \\
(1,2,3) & (1,1,1) & (1,1,1) & (1,1,1) & (1,2,3) & (1,2,3) & (1,2,3) & (1,1,1) & (1,1,1) & (1,1,1) & (1,2,3) & (1,2,3) \\
(1,2,3) & (1,1,1) & (1,1,1) & (1,1,1) & (1,2,3) & (1,2,3) & (1,2,3) & (1,1,1) & (1,1,1) & (1,1,1) & (1,2,3) & (1,2,3) \\
(1,2,3) & (1,1,1) & (1,1,1) & (1,1,1) & (1,2,3) & (1,2,3) & (1,2,3) & (1,1,1) & (1,1,1) & (1,1,1) & (1,2,3) & (1,2,3) \\
(1,1,1) & (1/3,1/2,1) & (1/3,1/2,1) & (1/3,1/2,1) & (1,2,3) & (1,2,3) & (1,2,3) & (1/3,1/2,1) & (1/3,1/2,1) & (1/3,1/2,1) & (1,1,1) & (1,1,1) \\
(1,1,1) & (1/3,1/2,1) & (1/3,1/2,1) & (1/3,1/2,1) & (1,2,3) & (1,2,3) & (1,2,3) & (1/3,1/2,1) & (1/3,1/2,1) & (1/3,1/2,1) & (1,1,1) & (1,1,1)
\end{array}\right]. \tag{17}$$

(2) The fuzzy judgment matrix H^1 and H^2 of the two experts are fused through Equation (2) to obtain the comprehensive fuzzy judgment matrix H^{com}, as follows:

$$H^{\text{com}} = \left[\begin{array}{cccccccccccc}
(1,1,1) & (7/24,5/12,3/4) & (7/24,5/12,3/4) & (7/24,5/12,3/4) & (1,2,3) & (1,2,3) & (1,2,3) & (7/24,5/12,3/4) & (7/24,5/12,3/4) & (7/24,5/12,3/4) & (1,1,1) & (1,1,1) \\
(3/2,5/2,7/2) & (1,1,1) & (1,1,1) & (1,1,1) & (3/2,5/2,7/2) & (3/2,5/2,7/2) & (3/2,5/2,7/2) & (1,1,1) & (1,1,1) & (1,1,1) & (1,2,3) & (1,2,3) \\
(3/2,5/2,7/2) & (1,1,1) & (1,1,1) & (1,1,1) & (3/2,5/2,7/2) & (3/2,5/2,7/2) & (3/2,5/2,7/2) & (1,1,1) & (1,1,1) & (1,1,1) & (1,2,3) & (1,2,3) \\
(3/2,5/2,7/2) & (1,1,1) & (1,1,1) & (1,1,1) & (3/2,5/2,7/2) & (3/2,5/2,7/2) & (3/2,5/2,7/2) & (1,1,1) & (1,1,1) & (1,1,1) & (1,2,3) & (1,2,3) \\
(1/3,1/2,1) & (7/24,5/12,3/4) & (7/24,5/12,3/4) & (7/24,5/12,3/4) & (1,1,1) & (1,1,1) & (1,1,1) & (7/24,5/12,3/4) & (7/24,5/12,3/4) & (7/24,5/12,3/4) & (1/3,1/2,1) & (1/3,1/2,1) \\
(1/3,1/2,1) & (7/24,5/12,3/4) & (7/24,5/12,3/4) & (7/24,5/12,3/4) & (1,1,1) & (1,1,1) & (1,1,1) & (7/24,5/12,3/4) & (7/24,5/12,3/4) & (7/24,5/12,3/4) & (1/3,1/2,1) & (1/3,1/2,1) \\
(1/3,1/2,1) & (7/24,5/12,3/4) & (7/24,5/12,3/4) & (7/24,5/12,3/4) & (1,1,1) & (1,1,1) & (1,1,1) & (7/24,5/12,3/4) & (7/24,5/12,3/4) & (7/24,5/12,3/4) & (1/3,1/2,1) & (1/3,1/2,1) \\
(3/2,5/2,7/2) & (1,1,1) & (1,1,1) & (1,1,1) & (3/2,5/2,7/2) & (3/2,5/2,7/2) & (3/2,5/2,7/2) & (1,1,1) & (1,1,1) & (1,1,1) & (1,2,3) & (1,2,3) \\
(3/2,5/2,7/2) & (1,1,1) & (1,1,1) & (1,1,1) & (3/2,5/2,7/2) & (3/2,5/2,7/2) & (3/2,5/2,7/2) & (1,1,1) & (1,1,1) & (1,1,1) & (1,2,3) & (1,2,3) \\
(3/2,5/2,7/2) & (1,1,1) & (1,1,1) & (1,1,1) & (3/2,5/2,7/2) & (3/2,5/2,7/2) & (3/2,5/2,7/2) & (1,1,1) & (1,1,1) & (1,1,1) & (1,2,3) & (1,2,3) \\
(1,1,1) & (1/3,1/2,1) & (1/3,1/2,1) & (1/3,1/2,1) & (1,2,3) & (1,2,3) & (1,2,3) & (1/3,1/2,1) & (1/3,1/2,1) & (1/3,1/2,1) & (1,1,1) & (1,1,1) \\
(1,1,1) & (1/3,1/2,1) & (1/3,1/2,1) & (1/3,1/2,1) & (1,2,3) & (1,2,3) & (1,2,3) & (1/3,1/2,1) & (1/3,1/2,1) & (1/3,1/2,1) & (1,1,1) & (1,1,1)
\end{array}\right]. \tag{18}$$

(3) The fuzzy weight $\tilde{\alpha}$ of each component is calculated using Equation (4) based on the comprehensive fuzzy matrix H^{com}, and the calculation result is accurate to three decimal places. The specific calculation procedure is shown as follows:

(i) The fuzzy weight of component C_1:

$$\tilde{\alpha}_1 = \left(\frac{\sum_{j=1}^{12} l_{1j}^{\text{com}}}{\sum_{i=1}^{12}\sum_{j=1}^{12} l_{ij}^{\text{com}}}, \frac{\sum_{j=1}^{12} m_{1j}^{\text{com}}}{\sum_{i=1}^{12}\sum_{j=1}^{12} m_{ij}^{\text{com}}}, \frac{\sum_{j=1}^{12} u_{1j}^{\text{com}}}{\sum_{i=1}^{12}\sum_{j=1}^{12} u_{ij}^{\text{com}}} \right) = (0.063, 0.065, 0.069). \tag{19}$$

(ii) The fuzzy weight of component C_2:

$$\tilde{\alpha_2} = \left(\frac{\sum_{j=1}^{12} l_{2i}^{\text{com}}}{\sum_{i=1}^{12}\sum_{j=1}^{12} l_{ij}^{com}}, \frac{\sum_{j=1}^{12} m_{2i}^{\text{com}}}{\sum_{i=1}^{12}\sum_{j=1}^{12} m_{ij}^{\text{com}}}, \frac{\sum_{j=1}^{12} u_{2i}^{\text{com}}}{\sum_{i=1}^{12}\sum_{j=1}^{12} u_{ij}^{\text{com}}} \right) = (0.114, 0.113, 0.108). \tag{20}$$

(iii) The fuzzy weight of component C_3:

$$\tilde{\alpha_3} = \left(\frac{\sum_{j=1}^{12} l_{3i}^{\text{com}}}{\sum_{i=1}^{12}\sum_{j=1}^{12} l_{ij}^{\text{com}}}, \frac{\sum_{j=1}^{12} m_{3i}^{\text{com}}}{\sum_{i=1}^{12}\sum_{j=1}^{12} m_{ij}^{\text{com}}}, \frac{\sum_{j=1}^{12} u_{3i}^{\text{com}}}{\sum_{i=1}^{12}\sum_{j=1}^{12} u_{ij}^{\text{com}}} \right) = (0.114, 0.113, 0.108). \tag{21}$$

(iv) The fuzzy weight of component C_4:

$$\tilde{\alpha_4} = \left(\frac{\sum_{j=1}^{12} l_{4i}^{\text{com}}}{\sum_{i=1}^{12}\sum_{j=1}^{12} l_{ij}^{\text{com}}}, \frac{\sum_{j=1}^{12} m_{4i}^{\text{com}}}{\sum_{i=1}^{12}\sum_{j=1}^{12} m_{ij}^{\text{com}}}, \frac{\sum_{j=1}^{12} u_{4i}^{\text{com}}}{\sum_{i=1}^{12}\sum_{j=1}^{12} u_{ij}^{\text{com}}} \right) = (0.114, 0.113, 0.108). \tag{22}$$

(v) The fuzzy weight of component C_5:

$$\tilde{\alpha_5} = \left(\frac{\sum_{j=1}^{12} l_{5i}^{\text{com}}}{\sum_{i=1}^{12}\sum_{j=1}^{12} l_{ij}^{\text{com}}}, \frac{\sum_{j=1}^{12} m_{5i}^{\text{com}}}{\sum_{i=1}^{12}\sum_{j=1}^{12} m_{ij}^{\text{com}}}, \frac{\sum_{j=1}^{12} u_{5i}^{\text{com}}}{\sum_{i=1}^{12}\sum_{j=1}^{12} u_{ij}^{\text{com}}} \right) = (0.047, 0.040, 0.044). \tag{23}$$

(vi) The fuzzy weight of component C_6:

$$\tilde{\alpha_6} = \left(\frac{\sum_{j=1}^{12} l_{6i}^{\text{com}}}{\sum_{i=1}^{12}\sum_{j=1}^{12} l_{ij}^{\text{com}}}, \frac{\sum_{j=1}^{12} m_{6i}^{\text{com}}}{\sum_{i=1}^{12}\sum_{j=1}^{12} m_{ij}^{\text{com}}}, \frac{\sum_{j=1}^{12} u_{6i}^{\text{com}}}{\sum_{i=1}^{12}\sum_{j=1}^{12} u_{ij}^{\text{com}}} \right) = (0.047, 0.040, 0.044). \tag{24}$$

(vii) The fuzzy weight of component C_7:

$$\tilde{\alpha_7} = \left(\frac{\sum_{j=1}^{12} l_{7i}^{\text{com}}}{\sum_{i=1}^{12}\sum_{j=1}^{12} l_{ij}^{\text{com}}}, \frac{\sum_{j=1}^{12} m_{7i}^{\text{com}}}{\sum_{i=1}^{12}\sum_{j=1}^{12} m_{ij}^{\text{com}}}, \frac{\sum_{j=1}^{12} u_{7i}^{\text{com}}}{\sum_{i=1}^{12}\sum_{j=1}^{12} u_{ij}^{\text{com}}} \right) = (0.047, 0.040, 0.044). \tag{25}$$

(viii) The fuzzy weight of component C_8:

$$\tilde{\alpha_8} = \left(\frac{\sum_{j=1}^{12} l_{8i}^{\text{com}}}{\sum_{i=1}^{12}\sum_{j=1}^{12} l_{ij}^{\text{com}}}, \frac{\sum_{j=1}^{12} m_{8i}^{\text{com}}}{\sum_{i=1}^{12}\sum_{j=1}^{12} m_{ij}^{\text{com}}}, \frac{\sum_{j=1}^{12} u_{8i}^{\text{com}}}{\sum_{i=1}^{12}\sum_{j=1}^{12} u_{ij}^{\text{com}}} \right) = (0.114, 0.113, 0.108). \tag{26}$$

(ix) The fuzzy weight of component C_9:

$$\tilde{\alpha_9} = \left(\frac{\sum_{j=1}^{12} l_{9i}^{\text{com}}}{\sum_{i=1}^{12}\sum_{j=1}^{12} l_{ij}^{\text{com}}}, \frac{\sum_{j=1}^{12} m_{9i}^{\text{com}}}{\sum_{i=1}^{12}\sum_{j=1}^{12} m_{ij}^{\text{com}}}, \frac{\sum_{j=1}^{12} u_{9i}^{\text{com}}}{\sum_{i=1}^{12}\sum_{j=1}^{12} u_{ij}^{\text{com}}} \right) = (0.114, 0.113, 0.108). \tag{27}$$

(x) The fuzzy weight of component C_{10}:

$$\tilde{\alpha_{10}} = \left(\frac{\sum_{j=1}^{12} l_{10i}^{\text{com}}}{\sum_{i=1}^{12}\sum_{j=1}^{12} l_{ij}^{\text{com}}}, \frac{\sum_{j=1}^{12} m_{10i}^{\text{com}}}{\sum_{i=1}^{12}\sum_{j=1}^{12} m_{ij}^{\text{com}}}, \frac{\sum_{j=1}^{12} u_{10i}^{\text{com}}}{\sum_{i=1}^{12}\sum_{j=1}^{12} u_{ij}^{\text{com}}} \right) = (0.114, 0.113, 0.108). \tag{28}$$

(xi) The fuzzy weight of component C_{11}:

$$\tilde{\alpha_{11}} = \left(\frac{\sum_{j=1}^{12} l_{11i}^{\text{com}}}{\sum_{i=1}^{12}\sum_{j=1}^{12} l_{ij}^{\text{com}}}, \frac{\sum_{j=1}^{12} m_{11i}^{\text{com}}}{\sum_{i=1}^{12}\sum_{j=1}^{12} m_{ij}^{\text{com}}}, \frac{\sum_{j=1}^{12} u_{11i}^{\text{com}}}{\sum_{i=1}^{12}\sum_{j=1}^{12} u_{ij}^{\text{com}}} \right) = (0.057, 0.068, 0.075). \tag{29}$$

Table 2: The subjective weight of components.

Subjective weight	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}	C_{11}	C_{12}
α_i	0.065	0.112	0.112	0.112	0.043	0.043	0.043	0.112	0.112	0.112	0.067	0.067

(xii) The fuzzy weight of component C_{12}:

$$\tilde{\alpha_{12}} = \left(\frac{\sum_{j=1}^{12} l_{12j}^{\text{com}}}{\sum_{i=1}^{12}\sum_{j=1}^{12} l_{ij}^{\text{com}}}, \frac{\sum_{j=1}^{12} m_{12j}^{\text{com}}}{\sum_{i=1}^{12}\sum_{j=1}^{12} m_{ij}^{\text{com}}}, \frac{\sum_{j=1}^{12} u_{12j}^{\text{com}}}{\sum_{i=1}^{12}\sum_{j=1}^{12} u_{ij}^{\text{com}}} \right) = (0.057, 0.068, 0.075). \tag{30}$$

(4) The fuzzy weight of each component is defuzzified by Equation (5), and the subjective weight α_i of each component is calculated, as shown in Table 2.

6.2. Calculating the Objective Weights β of Components

(1) According to Figure 10, the direct influence matrix G between the components can be obtained, and the matrix G divides the maximum value of the sum of the elements of each row in the matrix, $A = 9$, to get the normalized influence matrix G_{sta}.

$$G = \begin{bmatrix} 0 & 3 & 3 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 3 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{31}$$

$$G_{\text{sta}} = \begin{bmatrix} 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{2}{9} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{2}{9} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{2}{9} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{3} & \frac{1}{9} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{3} & \frac{1}{3} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{9} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{9} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{9} \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{3} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{32}$$

(2) The normalized influence matrix G_{sta} is added to the indirect influence matrix $G_{\text{sta}}^2, \ldots, G_{\text{sta}}^{12}$, and the comprehensive influence matrix G_t is calculated by Equation (8). The calculation process is given as follows:

$$G_t = G_{\text{sta}} + G_{\text{sta}}^2 + \cdots G_{\text{sta}}^{12}. \tag{33}$$

Due to a large amount of calculation, the matrix power addition software is used to automatically calculate the result, as shown in matrix G_t.

Table 3: The influence degree and the influenced degree of each component.

Degree	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}	C_{11}	C_{12}
Influence degree	1.424	0.425	0.425	0.425	0.912	1.110	0.875	0.111	0.111	0.111	0.625	0
Influenced degree	0	0.333	0.333	0.333	0.888	0.629	0.737	0.544	0.544	0.544	0.578	1.091

Table 4: The objective weight of components.

Objective weight	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}	C_{11}	C_{12}
β_i	0.109	0.058	0.058	0.058	0.137	0.132	0.123	0.050	0.050	0.050	0.092	0.083

$$G_t = \begin{bmatrix}
0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & \frac{2}{9} & \frac{2}{27} & \frac{14762}{531441} & \frac{2}{81} & \frac{2}{81} & \frac{2}{81} & \frac{14762}{1594323} & \frac{27884}{1594323} \\
0 & 0 & 0 & 0 & \frac{2}{9} & \frac{2}{27} & \frac{132860}{4782969} & \frac{2}{81} & \frac{2}{81} & \frac{2}{81} & \frac{14762}{1594323} & \frac{27884}{1594323} \\
0 & 0 & 0 & 0 & \frac{2}{9} & \frac{2}{27} & \frac{132860}{4782969} & \frac{2}{81} & \frac{2}{81} & \frac{2}{81} & \frac{14762}{1594323} & \frac{27884}{1594323} \\
0 & 0 & 0 & 0 & \frac{2}{9} & \frac{2}{27} & \frac{132860}{4782969} & \frac{2}{81} & \frac{2}{81} & \frac{2}{81} & \frac{14762}{1594323} & \frac{27884}{1594323} \\
0 & 0 & 0 & 0 & 0 & \frac{1}{3} & \frac{66430}{531441} & \frac{1}{9} & \frac{1}{9} & \frac{1}{9} & \frac{66430}{1594323} & \frac{125479}{1594323} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 & \frac{1}{9} \\
0 & 0 & 0 & 0 & 0 & 0 & \frac{66430}{531441} & 0 & 0 & 0 & \frac{66430}{177147} & \frac{66430}{177147} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{9} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{9} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{9} \\
0 & 0 & 0 & 0 & 0 & 0 & \frac{66430}{177147} & 0 & 0 & 0 & \frac{66430}{531441} & \frac{66430}{531441} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}. \quad (34)$$

(3) According to the comprehensive influence matrix G_t, the influence degree and the influenced degree of each component can be obtained according to Equations (7) and (8), and the calculation result is accurate to three decimal places. The specific calculation process is given as follows.

The influence degree of component C_1 is computed by:

$$Inf_1 = \sum_{j=1}^{12} g_{1j}^t = 1.424. \quad (35)$$

The influenced degree of component C_1 is computed by:

$$Infed_1 = \sum_{j=1}^{12} g_{j1}^t = 0. \quad (36)$$

By using the similar method, the influence and influenced degree of C_i, $i = 2, \ldots, 12$ can be obtained, as shown in Table 3.

(4) The comprehensive influence $\mathrm{In}f^{\mathrm{com}}$ of the component can be obtained based on Equation (11), and then the normalization operation is performed to obtain the objective weight β_i of all component by Equation (12); the weight value of C_i is shown in Table 4.

6.3. Calculating the Comprehensive Weights γ of Components

(1) Considering the above calculated subjective and objective weights α, β as two mutually independent

Table 5: The comprehensive BPA function.

CBPA function	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}	C_{11}	C_{12}
$m(C_i)$	0.087	0.085	0.085	0.085	0.090	0.088	0.083	0.081	0.081	0.081	0.079	0.075

Table 6: The comprehensive weight of components.

Comprehensive weight	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}	C_{11}	C_{12}
γ_i	0.090	0.087	0.087	0.087	0.097	0.092	0.082	0.079	0.079	0.079	0.074	0.067

Table 7: The comparison between comprehensive weight and FAHP.

Method	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}	C_{11}	C_{12}
FAHP	0.065	0.112	0.112	0.112	0.043	0.043	0.043	0.112	0.112	0.112	0.067	0.067
Comprehensive weight	0.090	0.087	0.087	0.087	0.097	0.092	0.082	0.079	0.079	0.079	0.074	0.067

bodies of evidence, and the corresponding BPA functions are defined as $m_1(C_i)$, $m_2(C_i)$, $i = 1, 2, \ldots, 12$. Then, the comprehensive BPA (for short CBPA) function $m(C_i)$ is obtained by averaging the functions $m_1(C_i)$ and $m_2(C_i)$ according to the Equation (14) for $i = 1, 2, \ldots, n$, the value of comprehensive BPA function is obtained, as shown in Table 5.

(2) The comprehensive weight γ of components is calculated by combining the comprehensive BPA function $m(C)$ based on the D–S combination rule by Equation (15). The specific calculation process is given as follows:

$$1 - K = 1 - \sum_{i \neq j} m(C_i) \times m(C_j) = 0.084. \tag{37}$$

$$\begin{aligned}
\gamma_1 &= \frac{\{m(C_1)\}^2}{1-K} = 0.090, & \gamma_2 &= \frac{\{m(C_2)\}^2}{1-K} = 0.087,\\
\gamma_3 &= \frac{\{m(C_3)\}^2}{1-K} = 0.087, & \gamma_4 &= \frac{\{m(C_4)\}^2}{1-K} = 0.087,\\
\gamma_5 &= \frac{\{m(C_5)\}^2}{1-K} = 0.097, & \gamma_6 &= \frac{\{m(C_6)\}^2}{1-K} = 0.092,\\
\gamma_7 &= \frac{\{m(C_7)\}^2}{1-K} = 0.082, & \gamma_8 &= \frac{\{m(C_8)\}^2}{1-K} = 0.079,\\
\gamma_9 &= \frac{\{m(C_7)\}^2}{1-K} = 0.079, & \gamma_{10} &= \frac{\{m(C_{10})\}^2}{1-K} = 0.079,\\
\gamma_{11} &= \frac{\{m(C_{11})\}^2}{1-K} = 0.074, & \gamma_{12} &= \frac{\{m(C_{12})\}^2}{1-K} = 0.067.
\end{aligned} \tag{38}$$

Therefore, the comprehensive weight γ_i of the 12 components in this ticketing system is achieved, as shown in Table 6.

6.4. Comparison. We observed that the component's weight can be determined using either the FAHP method based on experts' evaluation or the objective influence value model. However, the experts' evaluation exists the arbitrariness and the influence value model tends to be overly idealized. In contrast, our proposed method strikes a balance between these approaches, resulting in a component weight allocation that better reflects the actual situation of software system development.

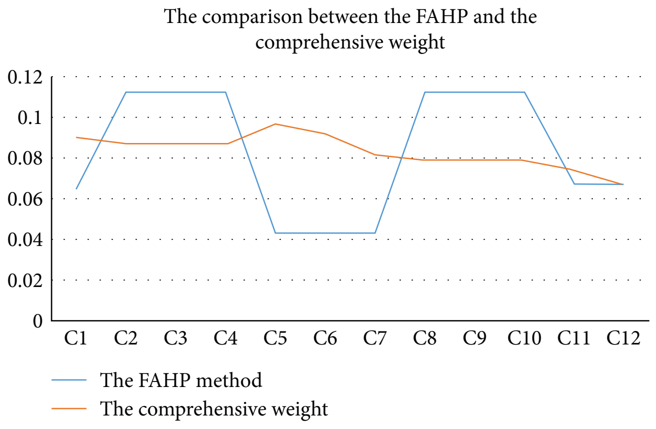

Figure 11: The comparison between comprehensive weight and FAHP.

Compared to FAHP, the results presented in Table 7 indicate that the weights of components C_2, C_3, C_4,C_8, C_9, and C_{10} are 0.112, which represents the highest weight obtained using the FAHP method among all components. Conversely, the weights of components C_{11}, C_{12} are 0.067, and the weight of C_1 is 0.065. Furthermore, the weights of C_5, C_6, C_7 are 0.043, representing the lowest weights achieved by the FAHP method. Notably, C_5 is the successor component of C_2, C_3, C_4, and the precursor component of C_6, C_7, as shown in Figure 10. Consequently, the execution of C_2, C_3, C_4 can significantly affect the execution of C_5, the execution of C_5 can affect the performance of C_6, meaning that the weight difference between C_2, C_3, C_4, and C_5 should not be excessively large. In the comprehensive weight model, the weights of C_2, C_3, C_4 are 0.087, while the weight of C_5 is 0.097, resulting in a reduced weight difference between C_2, C_3, C_4, and C_5. Additionally, as shown in Figure 11, the

Table 8: The comparison between comprehensive weight and influence value method.

Method	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}	C_{11}	C_{12}
Influence value method	0.109	0.058	0.058	0.058	0.137	0.132	0.123	0.050	0.050	0.050	0.092	0.083
Comprehensive weight	0.090	0.087	0.087	0.087	0.097	0.092	0.082	0.079	0.079	0.079	0.074	0.067

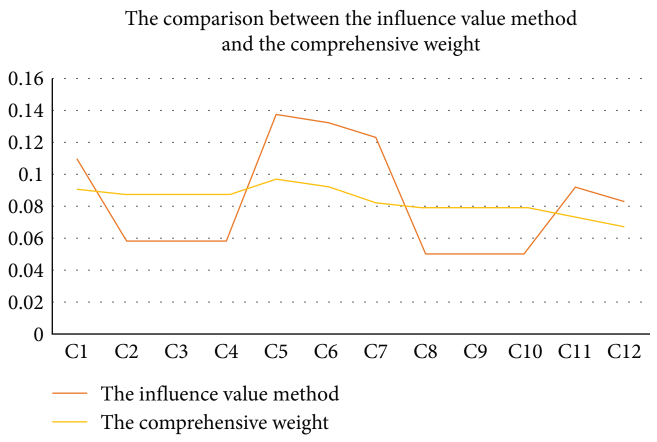

Figure 12: The comparison between comprehensive weight and influence value.

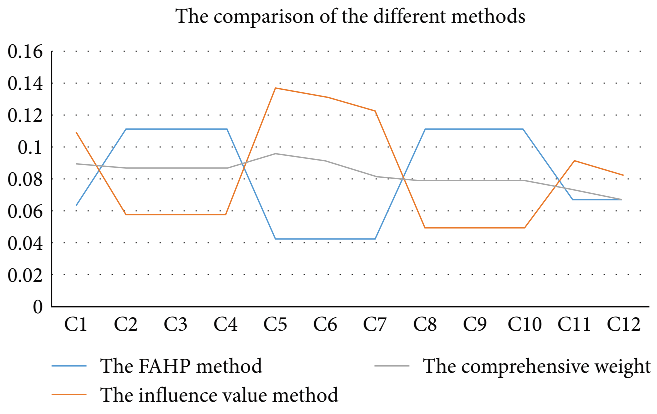

Figure 13: The comparison of the different methods.

FAHP method exhibits relatively significant fluctuations on the weight of component, which also showed the arbitrariness of the experts' evaluation.

In comparison with the objective weight allocation method that we proposed based on the influence value, we notice that the weights of C_8, C_9, C_{10} are 0.050, which is the least weight among all components. These components, referred to as the "payment" components, as shown in Figure 10, are crucial for the software as their failure can influence the entire system. Therefore, their weights should not be too small. In the comprehensive weight model, the weights of C_8, C_9, C_{10} have been improved to 0.079, as shown in Table 8. Additionally, Figure 12 demonstrates that the objective weight allocation method based on the influence value exhibits a relatively large fluctuation.

The detailed comparison is presented in Figure 13. As depicted in Figure 13, the comprehensive weight shows a low fluctuation, while the FAHP and influence value methods exhibit relatively large fluctuations. The main reason behind this observation is their deficiency in accurately capturing component weights. Through our computation, we noticed that the comprehensive weight allocation effectively reduces subjective arbitrariness and objective ideality, resulting in a component weight allocation that better aligns with the actual situation of software system development.

Compared to the method introduced in [30], to obtain the weight of component, the volatility of the component attribute should be calculated according to the standard deviation of attributes. Compared to the method in [31], to compute the weight of the component, the attribute value and defect data should be collected. But, in our case, we only have the data of influence value, lacking the data relating to the attribute of component. In fact, we can also use many methods to obtain the weight of components, but the different method depends on the different data set. In our case, our aim is to show the comprehensive weight method of combining the FAHP and influence value is more reasonable than the single method. Therefore, we only have the comparison with FAHP and influence value model.

7. Conclusions and Future Works

This study focused on the problem of allocating weight for the components to evaluate the importance of components during component-based software. The method of allocation weight proposed in this paper not only considers the expert's evaluation of the importance of components but also the interaction relationship between components. The framework of the weight allocation method is designed by the flow diagram. Our method can help software developers to produce a reasonable evaluation of component importance and develop highly trustworthy software. The contributions of this paper are shown as follows:

(i) For the complexity software, there exist many interactions between components. The mutual influence between components can reflect the importance of the component. In this paper, based on the combination structure of components, the influence value model between the components was proposed to characterize the degree of mutual influence between components.

(ii) For the weight of the researched object, the popular method is to get the value by the expert's evaluation. Though there exist some objective weight computation methods, the mutual influence between component didn't be considered. In this paper, the objective weight allocation method of components is presented to reflect the importance of the component based on mutual influence. During this method, the direct influence and indirect influence between components is established based on the matrix's power.

(iii) To obtain the importance of components more reasonably, it is necessary to consider the importance from the subjective and objective views to achieve the weight. In this paper, a new weight allocation method of components is built by combining the experts' evaluation with the influence value between components based on the evidence information fusion method. By comparing with the other methods, it is more reasonable to consider the mutual influence.

The method proposed in this paper can effectively allocate the weight of components, but the expert's evaluation is based on the triangular fuzzy set to express the evaluation information. For the triangular fuzzy set, there are some deficiencies in many situations. Generally, the fuzzy value of evaluation depends on experts' subjective perceptions and preferences. There exist several types of uncertainty, such as interpersonal perception ambiguousness, personal judgment reliability, interpersonal preference inconsistency [42, 43], and so on. However, previous studies, such as the triangular fuzzy set, didn't consider these uncertainties comprehensively, usually just considering one of the various uncertainties, which may affect their effectiveness. In [42], the authors developed an integrated design alternative assessment model by fully considering the various uncertainties. In the future, we will try to combine the method in [42] with the mutual influence model and find a more effective weight allocation method of the component.

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 (no. 62162014, 62077029), and the Abroad Visiting of Excellent Young Talents of Universities in Anhui Province (no. GXGWFX2019022).

References

[1] W. B. Frakes and P. B. Gandel, "Representing reusable software," *Information and Software Technology*, vol. 32, no. 10, pp. 653–664, 1990.

[2] W. Behutiye, P. Karhapää, L. López et al., "Management of quality requirements in agile and rapid software development: a systematic mapping study," *Information and Software Technology*, vol. 123, Article ID 106225, 2020.

[3] W. Behutiye, P. Rodríguez, M. Oivo, S. Aaramaa, J. Partanen, and A. Abhervé, "Towards optimal quality requirement documentation in agile software development: a multiple case study," *Journal of Systems and Software*, vol. 183, Article ID 111112, 2022.

[4] I. Atoum, M. K. Baklizi, I. Alsmadi et al., "Challenges of software requirements quality assurance and validation: a systematic literature review," *IEEE Access*, vol. 9, pp. 137613–137634, 2021.

[5] M. D. McIlroy, *Mass-Produced Software Components, Software Engineering Concepts and Techniques (1968 NATO Conference on Software Engineering)*, Van Nostrand Reinhold, New York, 1976.

[6] C. Szyperski, D. Gruntz, and S. Murer, *Component Software: Beyond Object-Oriented Programming*, Pearson Education, 2002.

[7] T. Vale, I. Crnkovic, E. S. de Almeida, P. A. da Mota Silveira Neto, Y. C. Cavalcanti, and S. R. de Lemos Meira, "Twenty-eight years of component-based software engineering," *Journal of Systems and Software*, vol. 111, pp. 128–148, 2016.

[8] Z. Wang, Z. Chen, and J. Zhang, "Component and component-based software development," in *Proceedings of the International Conference on Information Engineering and Applications, Lecture Notes in Electrical Engineering*, vol. 216, pp. 719–724, 2012.

[9] T. Skalniak, A. Rot, and W. Gryncewicz, "Software quality improvement by application of a component-based software meta-architecture," in *Computational Collective Intelligence*, N. Nguyen, R. Chbeir, E. Exposito, P. Aniorté, and B. Trawinski, Eds., pp. 428–439, Springer, Cham, 2019.

[10] B. Wang, Y. Chen, S. Zhang, and H. Wu, "Updating model of software component trustworthiness based on users feedback," *IEEE Access*, vol. 7, pp. 60199–60205, 2019.

[11] R. Sehgal, D. Mehrotra, and R. Nagpal, "Complexity metrics for component-based software system," in *Soft Computing: Theories and Applications*, K. Ray, T. Sharma, S. Rawat, R. Saini, and A. Bandyopadhyay, Eds., pp. 13–22, Springer, Singapore, 2019.

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

[13] T. L. Saaty, "A scaling method for priorities in hierarchical structures," *Journal of Mathematical Psychology*, vol. 15, no. 3, pp. 234–281, 1977.

[14] C. E. Shannon, "A mathematical theory of communication," *ACM SIGCOMM Computer Communication Review*, vol. 5, no. 1, pp. 3–55, 2001.

[15] 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.

[16] 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.

[17] A. Lozano-Tello and A. Gómez-Pérez, "BAREMO: how to choose the appropriate software component using the analytic hierarchy process," in *Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering*, pp. 781–788, Association for Computing Machinery, New York, NY, USA, 2002.

[18] A. Jadhav and R. Sonar, "Analytic hierarchy process (AHP), weighted scoring method (WSM), and hybrid knowledge based system (HKBS) for software selection: a comparative

study," in *2009 Second International Conference on Emerging Trends in Engineering & Technology*, pp. 991–997, IEEE, Nagpur, India, 2009.

[19] Y. P. Ren, "Research on evaluation model of software component quality based on group decision-making," *Computer Engineering and Design*, vol. 31, no. 21, pp. 4639–4641, 2010.

[20] X. Deng and Y. Deng, "D-AHP method with different credibility of information," *Soft Computing*, vol. 23, pp. 683–691, 2019.

[21] P. J. M. van Laarhoven and W. Pedrycz, "A fuzzy extension of Saaty's priority theory," *Fuzzy Sets and Systems*, vol. 11, no. 1–3, pp. 229–241, 1983.

[22] F. Ahmed and K. Kilic, "Fuzzy analytic hierarchy process: a performance analysis of various algorithms," *Fuzzy Sets and Systems*, vol. 362, pp. 110–128, 2019.

[23] G. Huang, L. Xiao, W. Pedrycz, G. Zhang, and L. Martinez, "Failure mode and effect analysis using T-spherical fuzzy maximizing deviation and combined comparison solution methods," *IEEE Transactions on Reliability*, vol. 72, no. 2, pp. 552–573, 2023.

[24] L. Xiao, G. Huang, W. Pedrycz, D. Pamucar, L. Martínez, and G. Zhang, "A q-rung orthopair fuzzy decision-making model with new score function and best-worst method for manufacturer selection," *Information Sciences*, vol. 608, pp. 153–177, 2022.

[25] D. Qiao, X.-K. Wang, J.-Q. Wang, and K. Chen, "Cross entropy for Discrete Z-numbers and its application in multi-criteria decision-making," *International Journal of Fuzzy Systems*, vol. 21, pp. 1786–1800, 2019.

[26] H. Kim, T. H. Lee, and T. Kwon, "Normalized neighborhood component feature selection and feasible-improved weight allocation for input variable selection," *Knowledge-Based Systems*, vol. 218, Article ID 106855, 2021.

[27] D. Diakoulaki, G. Mavrotas, and L. Papayannakis, "Determining objective weights in multiple criteria problems: the critic method," *Computers & Operations Research*, vol. 22, no. 7, pp. 763–770, 1995.

[28] D. Huang, Y. Ma, H. Pan, and M. Wang, "Software trustworthiness metric model based on component weight," *International Journal of Performability Engineering*, vol. 14, pp. 1985–1996, 2018.

[29] C.-H. Chen, "A novel multi-criteria decision-making model for building material supplier selection based on entropy-AHP weighted TOPSIS," *Entropy*, vol. 22, no. 2, Article ID 259, 2020.

[30] X. Gao, Y. Ma, and W. Zhou, "Analysis of software trustworthiness based on FAHP-CRITIC method," *Journal of Shanghai Jiaotong University (Science)*, 2022.

[31] Y. Ma, X. Gao, and W. Zhou, "The trustworthiness measurement model of component-based software based on combination weight," in *Artificial Intelligence Logic and Applications*, Y. Chen and S. Zhang, Eds., Springer, Singapore, 1657.

[32] A. U. Alrubaee, D. Cetinkaya, G. Liebchen, and H. Dogan, "A process model for component-based model-driven software development," *Information*, vol. 11, no. 6, Article ID 302, 2020.

[33] U. K. Tiwari and S. Kumar, "Actual interactions for component-based software using straight and circuitous links," *International Journal of Innovative Technology and Exploring Engineering*, vol. 9, no. 3, pp. 2495–2504, 2020.

[34] C. Rathfelder, *Modelling Event-Based Interactions in Component-Based Architectures for Quantitative System Evaluation*, KIT Scientific Publishing, 2012.

[35] A. Kaufmann and M. M. Gupta, *Fuzzy Mathematical Models in Engineering and Management Science*, Elsevier Science Inc, 1988.

[36] A. P. Dempster, "Upper and lower probabilities induced by a multivalued mapping," *The Annals of Mathematical Statistics*, vol. 38, no. 2, pp. 325–339, 1967.

[37] G. Shafer, *A Mathematical Theory of Evidence*, Princeton University Press, 1976.

[38] J. Y. Xiao, "*Study on evidence theory and its application to the water inrush prediction in mine*," Ph.D. Thesis, China University of Mining and Technology, Xuzhou, 2012.

[39] L. Cheng, Y. Lin, T. Li, X. Wang, J. Zheng, and X. Li, "A method of component importance measurement based on node betweenness in software evolution environment," *Computer Applications and Software*, vol. 34, no. 10, pp. 29–34, 2017.

[40] W.-W. Wu, "Choosing knowledge management strategies by using a combined ANP and DEMATEL approach," *Expert Systems with Applications*, vol. 35, no. 3, pp. 828–835, 2008.

[41] C. K. Murphy, "Combining belief functions when evidence conflicts," *Decision Support Systems*, vol. 29, no. 1, pp. 1–9, 2000.

[42] G. Huang, L. Xiao, W. Pedrycz, D. Pamucar, G. Zhang, and L. Martínez, "Design alternative assessment and selection: a novel Z-cloud rough number-based BWM-MABAC model," *Information Sciences*, vol. 603, pp. 149–189, 2022.

[43] G. Huang, L. Xiao, and G. Zhang, "Decision-making model of machine tool remanufacturing alternatives based on dual interval rough number clouds," *Engineering Applications of Artificial Intelligence*, vol. 104, Article ID 104392, 2021.