Received June 9, 2020, accepted June 29, 2020, date of publication July 6, 2020, date of current version July 17, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3007158

Multi-Value Models for Allocation of Software Component Development Costs Based on Trustworthiness

MENGYUE WANG[1], YANFANG MA[1], GUANRU LI[1], WEI ZHOU[1], AND LIANG CHEN[2]
[1]School of Computer Science and Technology, Huaibei Normal University, Huaibei 235000, China
[2]School of Mathematics Science, Huaibei Normal University, Huaibei 235000, China

Corresponding author: Yanfang Ma (clmyf@163.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61672023, in part by the Provincial Projects of Natural Science for Anhui Universities under Grant KJ2019A0604 and Grant KJ2019A0952, in part by the Doctoral Research Fund of Zhengzhou University of Light Industry under Grant 13501050045, and in part by the Science and Technology Project of Henan Province under Grant 182102210617.

ABSTRACT The trustworthiness of software is an important attribute. The cost of software development increases with its improvement by software trustworthiness. As one of main methods of software development, component-based software development can reduce development costs to a certain extent. However, it is important to study how to allocate the given development costs to each component so that software trustworthiness can be optimized. First, multi-value models for allocation of software component development costs are established based on different structures of software system. Second, algorithms for allocation of software component development costs can be designed by using dynamic programming. The proposed allocation algorithms can allocate development costs to each component to optimize software trustworthiness. Furthermore, in order to allocate development costs to each component automatically, a web-based software tool for allocating development costs to each component is developed. Finally, a case study of a self-service ticketing system is provided to show the feasibility of the proposed allocation algorithms.

INDEX TERMS Allocation algorithm, component, development costs, dynamic programming, software trustworthiness.

I. INTRODUCTION

A. BACKGROUND

Software is an integral part of fields such as energy, communications, manufacturing, finance, government, as well as our everyday lives [1]. However, software is not always trustworthy [2]. The faults and defects of software system bring loss to users directly or indirectly, sometimes even threaten people's life and safety [3], [4]. Therefore, more and more attention has been paid to software trustworthiness, and software trustworthiness has become one of the important research topics in the field of software engineering [5], [6].

Software trustworthiness is the ability of software to satisfy users' expectations with its behaviors and results and to still provide continuous services during a software disturbance [4]. It can be characterized by many software attributes, which are called trustworthy attributes [7]. Research performed on the topics of software trustworthiness has primarily included measurement, evaluation, and allocation of software trustworthiness. The measurement of software trustworthiness can provide an important reference to software trustworthiness [8], [9]. Then, the evaluation of software trustworthiness can not only help users choose trustworthy software, but it can also provide evidence for increasing the trustworthiness of the design and implementation of software [10]. In order to ensure software trustworthiness, analysts of software requirements need to determine the trustworthiness degree of each unit or specific attributes. Therefore, allocation of software trustworthiness becomes the overarching problem. The allocation of software trustworthiness has a practical significance to software development and software applications.

There has been so much research on the issue of software trustworthiness measurement and evaluation [11]–[14].

The associate editor coordinating the review of this manuscript and approving it for publication was Abdullah Iliyasu.

However, the allocation of software trustworthiness has developed in recent years. There is a scarcity of literature on the allocation of software trustworthiness. Software trustworthiness can be characterized by trustworthy attributes. Some researchers have studied the allocation of software trustworthiness in terms of trustworthy attributes. In 2015, Ma *et al.* presented an attribute-based trustworthiness allocation model and a corresponding algorithm, which allocates software trustworthiness to trustworthy attributes in order to attain the goal of software trustworthiness [14]. In 2019, Tao *et al.* proposed a mathematical programming approach to allocate the trustworthiness degree to each sub-attribute of some trustworthy attributes appropriately and then to make the trustworthiness degree of this attribute maximize under some constraint conditions [7]. Moreover, a polynomial allocation algorithm is given for computing the optimal solution of mathematical programming. Based on [7], they also established a reallocation approach for modeling software trustworthiness [15]. Component-based software development (CBSD) is the mainstream technology of developing software. CBSD avoids duplication of effort, reduces development costs and improves productivity [16]. By reusing software component, CBSD avoids repeated emergence of errors and improves software trustworthiness. In 2019, Wang *et al.* proposed an updated model of software component trustworthiness based on users' feedback and the number of users. At the same time, the values of the weights were allocated by the positive reciprocal matrix [17].

In the process of software development, there are some problems in the allocation of development costs, such as unbalanced development costs allocation of components and inaccurate cost estimation of components, which will bring great difficulties to the development of software system and the trustworthiness of software system cannot be guaranteed. However, [7], [14], and [17] did not consider development costs. Few researchers today pay attention to the allocation of development costs when studying software trustworthiness. More attentions need to be paid to the problem of allocating development costs to optimize software trustworthiness. Therefore, the optimization of software trustworthiness and development costs allocation is an important research field in software engineering.

At present, the research on development costs is mainly divided into two aspects. On one hand, the development costs of software system can be minimized while the goal of software trustworthiness can be ensured. On the other hand, software trustworthiness can be maximized when the development costs are given by the user. In 2019, Huang *et al.* studied how to allocate the trustworthiness of each component to minimize development costs of the overall software system when users gave software trustworthiness as the goal [18]. In software development, software developers hope to find a reasonable way to allocate development costs given by users, so as to improve software trustworthiness. In 2020, Wang *et al.* established two-value models for allocation of development costs under different structures of software system, and corresponding allocation algorithms were designed by using dynamic programming [19]. For each component, the maximum trustworthiness and minimum trustworthiness were given when the development costs were allocated to it. In [19], an array of 0/1 elements $x = [x_1, x_2, \cdots, x_n]$ was used to represent whether the component was developed or not. The value 1 presented that the development costs $e_k (k = 1, 2, \cdots, n)$ was allocated to the k^{th} component. The trustworthiness of the k^{th} component arrived the maximum trustworthiness $T_{k.\max}$. The value 0 presented that the development costs e_k was not allocated to the k^{th} component. The trustworthiness of the k^{th} component arrived the initial trustworthiness $T_{k.\min}$. However, generally, each component has different trustworthiness when we allocate different development costs to the component. The development costs are normally at too high a level to be allocated directly in [19], and the development costs can be further subdivided into multi-value costs allocation. In order to improve the trustworthiness of software system, the multi-value models for allocation of software component development costs are established based on different structures of software system. Within the development costs given by users, the allocation algorithms are designed in order to optimize the trustworthiness of software system. The main contributions of this paper are as follows.

1) The multi-value allocation models are proposed based on different component structures.
2) The allocation algorithms of software component development costs are designed to obtain the optimal trustworthiness of software system in different structures.
3) A web-based software component development costs allocation tool is developed to perform the automatic development costs allocation of each component.
4) A case study of a self-service ticketing system is included to show the feasibility of the proposed allocation algorithms.

B. RELATED WORKS

To date, the research on the allocation method of software trustworthiness mainly focused on two topics. One topic is the allocation of trustworthiness on the attributes. Another topic pays more attention to the allocation of weight value for each attribute.

For example, in [14], based on software trustworthiness goal, an attribute-based allocation model of software trustworthiness was given. In this model, first, user gave software trustworthiness goal T^*. The lowest weight of all attributes was also given. The specified lowest trustworthiness degree that each attribute must reach was given. Then, according to the equation $y_i = y^* + m(\alpha_i - \alpha_{\min})$, the trustworthiness degree of the i^{th} attribute could be obtained, where m was the trustworthiness growth rate of attributes, and α_i was the weight of the i^{th} attribute. Finally, in order to make the trustworthiness of software satisfies the trustworthiness goal, the author used the following equation to get the suitable parameter m.

$$\begin{cases} \prod_{i=1}^{n}(y^* + m(\alpha_i - \alpha_{\min}))^{\alpha_i} \geq T^* & y^* \leq y_i \leq 1 \\ \sum_{i=1}^{n} \alpha_i = 1 & 0 \leq \alpha_i \leq 1 \\ \min m & 0 \leq m \leq \dfrac{1-y^*}{\alpha_{\max} - \alpha_{\min}}. \end{cases} \tag{1}$$

When the suitable parameter was chosen, the value of every attribute could be obtained. Specially, the allocation algorithm was designed to automatically perform the allocation.

Furthermore, in 2019, based on the metric model of trustworthiness in [7], Tao *et al.* proposed a mathematical programming approach to allocate the trustworthiness degree to each sub-attribute of some software attributes appropriately. A polynomial allocation algorithm was given for computing the optimal solution of mathematical programming. There are some differences between the allocation in [7] and [14]. In [14], the range of each attribute value belongs to the interval [0,1]. The allocation of software trustworthiness described the process of determining the trustworthiness degree of each software attribute within the given trustworthiness degree of software. However, in [7], the trustworthy degrees of the sub-attributes came from the set $\{0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1\}$. The allocation of software attribute trustworthiness in [7] described how to get the trustworthiness degree of each software sub-attribute according to the given software attribute trustworthiness. On the other hand, it is enough when a feasible solution was found in this model in [14]. However, in [7], it needed to find the optimal solution of the allocation of software attribute trustworthiness. In order to obtain more optimal solutions, Tao *et al.* reallocated the sub-attributes in [15].

However, we notice that the proposed methods in [7], [14], and [15] only considered the allocation of the trustworthiness degree of the attributes. They did not consider the allocation of development costs.

In [17], according to users' feedback and the number of users, the attribute weights of the components were allocated, and the updated software trustworthiness could be calculated. The detailed model was showed as follows:

$$\begin{cases} T_n = \omega_0 T_0 + \omega_u(n) T_u, \\ \omega_0 + \omega_u(n) = 1, \end{cases} \tag{2}$$

where T_n was the updated trustworthiness degree of software components; T_0 was original trustworthiness degree of software; T_u was the trustworthiness degree of users' feedback; n was the number of users; ω_0 was the weight given to the original trustworthiness degree; and $\omega_u(n)$ was the weight of users' feedback. Usually, the values of the weights were assigned by the experts. However, it is too subjective to reflect the real situation of the weights. In [17], the authors used the positive reciprocal matrix to compute the values of weights such that the weights could be allocated and the trustworthiness could be obtained.

In [18], the author proposed the metric model of trustworthiness based different component structures. The values of the weights were computed by using the analytic hierarchy process. There are difference between [17] and [18]. One the one hand, the measurement model is different and the meaning of the weights is different. On the other hand, the positive reciprocal matrix is different from the analytic hierarchy process.

For the allocation of development costs, there has been some research. To date, we find the following models are related to development costs.

In [18], the author studied how to allocate the trustworthiness degree of each subsystem to reach the goal of software trustworthiness, in order to minimize development costs. The author proposed a function to establish the relation between trustworthiness of software and the complexity degree of software. The model was as follows:

$$\begin{aligned} &\min E = \sum_{k=1}^{n} E(T_k), \\ &s.\,t.\ g(T_k) \geq T, \end{aligned} \tag{3}$$

where E was development costs; T_k was the trustworthiness degree of the k^{th} subsystem; $E(T_k)$ was the development costs of the k^{th} subsystem; g was the function of development costs allocation of the software system in different structures; and T was software trustworthiness goal. From this model, we can know that the aim of this model is to allocate the trustworthiness degree to each component such that the development costs are the minimum. However, the aim of this paper is to allocate development costs to each component such that the trustworthiness degree can be the maximum. The goal is different.

In [19], in order to improve the trustworthiness degree of software, the additional development costs are required. There are many components in the component-based software. It is necessary to allocate development costs to the components. In [19], the author used an array $x = [x_1, x_2, \cdots, x_k, \cdots, x_n]$ to describe development costs allocation of the components. When $x_k = 1$, it expressed that development costs were allocated to the k^{th} component, otherwise, $x_k = 0$. At the same time, if additional development costs were allocated to the component, the trustworthiness degree of this component would achieve the maximum trustworthiness $T_{k.\max}$. If additional development costs were not allocated to the component, the trustworthiness degree of this component only had the minimum trustworthiness $T_{k.\min}$. The detailed model was showed as follows:

$$\begin{aligned} &T = \max L(T_1, T_2, \cdots, T_n), \\ &s.\,t.\ \sum_{k=1}^{n} (e_k x_k) \leq e, \end{aligned} \tag{4}$$

where T_k was the trustworthiness degree of the k^{th} component; when $x_k = 0, T_k = T_{k.\min}$; $x_k = 1, T_k = T_{k.\max}$. e_k was development costs which could make the component achieves the maximum trustworthiness; e was additional development costs. L was the allocation function in different component structures.

However, the component can have different trustworthiness degrees under different development costs. In [19], there is only one case of development costs allocation. The component can have the maximum trustworthiness when the corresponding development costs can be allocated to it, or the component can only have the initial trustworthiness when there are no development costs allocated to it. It is too absolute to reflect the real relation between the trustworthiness degree of component and development costs. Therefore, we generalize the allocation models of development costs in this paper.

The differences between the proposed models and the models in [19] are showed as follows:

1) The function to express the relation between the development costs and the trustworthiness degree is different. In this paper, we prove the function satisfies the properties of the cost function proposed by Dale and Winterbottom in 1986 [20].
2) When the component is allocated development costs, the trustworthiness degree of the component can be computed according to the function of development costs and the trustworthiness degree. However, in [19], the trustworthiness degree of the component is assigned as the maximum.
3) For every component structure, there are different recursive equations and algorithms to achieve the optimal solution.

The rest of the paper is organized as follows. Section II shows the model for development costs estimation of a component. Section III establishes the multi-value models for allocation of software component development costs under different structures of software system. In addition, the corresponding allocation algorithms are proposed by using dynamic programming. In Section IV, the development of a web-based software component development costs allocation tool is described. At the same time, the proposed allocation algorithms are applied to the self-service ticketing system in Section V. And, the comparative experiment is showed. Section VI offers our conclusions and future research.

II. MODEL FOR DEVELOPMENT COSTS ESTIMATION OF A COMPONENT

Development costs are required to improve component trustworthiness. The research on the relationship between component trustworthiness and development costs is the basis of optimizing trustworthiness of software system.

First, the development costs of a component are closely related to the complexity of a component. If the complexity of a component is higher, the development costs will be higher. Second, if the requirement of component trustworthiness is higher, the development costs will be higher. Therefore, the model for development costs estimation of a component is given as follows:

$$C(T_k) = -f_k / \ln T_k, \tag{5}$$

where f_k denotes the complexity of the k^{th} component where $f_k \in (0, 1)$; T_k denotes the trustworthiness of the k^{th} component. $C(T_k)$ denotes the needed development costs of the k^{th} component if the trustworthiness of the k^{th} component is T_k.

Equation (5) satisfies the relationship between the component trustworthiness and development costs in practice. If the structure of a component is simple, the development costs are low. Otherwise, the structure of component is complex and the development is difficult, then the development costs are relatively higher. When the trustworthiness of one component is closer to the maximum trustworthiness, the development costs will increase rapidly to infinity.

Equation (5) can satisfy the properties of the cost function proposed by Dale and Winterbottom in 1986 [20], and the properties are as follows:

Proposition 1: Suppose a software system is composed of n components. Let T_k be the trustworthiness of the k^{th} component. $C(T_k') - C(T_k)$ expresses the cost of improving the trustworthiness degree from T_k to T_k', where $0 < T_k \leq T_k' < 1$. Then, there are the following properties:

1) $C(T_k') - C(T_k) \geq 0$.
2) $\lim\limits_{T_k \to 1} C(T_k) \to \infty$.
3) $C(T_k'') - C(T_k) = C(T_k') - C(T_k) + C(T_k'') - C(T_k')$, where $0 < T_k \leq T_k' \leq T_k'' < 1$.
4) $\frac{dC(T_k)}{dT_k} > 0$.
5) $\frac{d^2C(T_k)}{dT_k^2} > 0$.

Proof: It is easy to get the proof according to the definition of $C(T_k)$ and the meaning of first derivative and second derivative of $C(T_k)$.

III. MODEL FOR ALLOCATION OF SOFTWARE COMPONENT DEVELOPMENT COSTS BASED ON DIFFERENT STRUCTURES

In practice, users will explain their requirements and functions to developers or a software development company in the phase of requirement analysis. Generally, based on previous development experience, developers or a software development company can estimate basic development costs that will be entailed to incorporate specific functionalities. However, basic development costs are only estimates, not exact costs. The developers can design and develop software based on the given basic costs. If users are not satisfied with software quality, more development is needed, and more development costs are incurred. Generally, a software system consists of different components, and the components are designed by the different teams. Therefore, it is necessary to allocate development costs to different components when the additional costs are given to improve software trustworthiness. The different costs can lead to different trustworthiness of each component. So, in order to optimize software trustworthiness, how to allocate the development costs given by users to each component is a key problem.

We suppose that a software system consists of n components: $C_1, C_2, \cdots, C_n$, and the development costs given by users are denoted by e, which refers to additional development costs of improving the trustworthiness of some components (all components by default), in addition to basic development costs. For the sake of research convenience, we suppose that the value of development costs is a non-negative integer, $x = 0, 1, \cdots, e$. Generally, if we spend more money on component development, the component trustworthiness will be higher. According to (5), we suppose that $T_k(x)$ represents the $k^{\text{th}}(k = 1, 2, \cdots, n)$ component trustworthiness when we allocate additional development costs x to develop the k^{th} component, as shown in Table 1.

TABLE 1. Component trustworthiness under different development costs.

x	0	1	2	$\cdots$	e
$T_1(x)$	$T_1(0)$	$T_1(1)$	$T_1(2)$	$\cdots$	$T_1(e)$
$T_2(x)$	$T_2(0)$	$T_2(1)$	$T_2(2)$	$\cdots$	$T_2(e)$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
$T_n(x)$	$T_n(0)$	$T_n(1)$	$T_n(2)$	$\cdots$	$T_n(e)$

In CBSD, developers will use basic development costs to develop each component, so that each component has initial trustworthiness. Overall software trustworthiness depends on the trustworthiness of each component. On the other hand, a software system has structures, such as sequence structure, parallel structure, branch structure, and loop structure, which also have a strong effect on software trustworthiness [21].

We suppose that the initial trustworthiness of the $k^{\text{th}}(k = 1, 2, \cdots, n)$ component is denoted by $T_{k.\min}$, so $T_k(0) = T_{k.\min}$. The trustworthiness of the k^{th} component is denoted by T_k, $0 < T_{k.\min} \leq T_k < 1$. The development costs of the k^{th} component are denoted by e_k. In CBSD, the function of development costs allocation in different structures is denoted by G. The multi-value model for the allocation of development costs is defined as follows:

$$T = \max G(T_1, T_2, \cdots, T_n), \quad s.t. \sum_{k=1}^{n} e_k \leq e, \tag{6}$$

where T presents the optimal trustworthiness of software system pertaining to its different structures.

In the next section of this paper, we will look at the models for allocation of software component development costs in sequence structure, branch structure, parallel structure, and loop structure.

A. MODEL FOR ALLOCATION OF SOFTWARE COMPONENT DEVELOPMENT COSTS IN SEQUENCE STRUCTURE

In sequence structure, each component is independent of each other and components run in turn [21], as shown in Fig. 1.

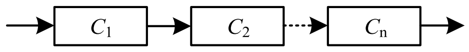


FIGURE 1. Software system in sequence structure.

In sequence structure, the failure of a single component will lead to the failure of overall system. Therefore, lower component trustworthiness contributes to lower software trustworthiness. As a result, the trustworthiness of a software system with a sequence structure can be obtained by multiplying the trustworthiness of each component. Accordingly, the multi-value model for allocation of software component development costs in sequence structure is defined as follows:

$$T_S = \max \prod_{k=1}^{n} T_k, \quad s.t. \sum_{k=1}^{n} e_k \leq e, \tag{7}$$

where T_S presents the optimal software trustworthiness in sequence structure.

Next, the allocation algorithm is designed by using dynamic programming. According to the number of components n, The development costs allocation stage is divided into n stages. First, the development costs are allocated to the first component. Then, development costs are allocated to the first two components until the development costs allocation of n components is taken into consideration. We suppose that $F_k^S(x)_{\max}(k = 1, 2, \cdots, n, x = 0, 1, \cdots, e)$ represents the optimal trustworthiness of a software system composed of the top k components when we allocate development costs x to develop the top k components in sequence structure. First, we allocate $x_k(0 \leq x_k \leq x)$ from x to develop the k^{th} component, and the rest of development costs $x - x_k$ will be used to develop the top $k - 1$ components. Therefore, recursive equations and boundary conditions are defined as follows:

$$F_k^S(x)_{\max} = \max_{0 \leq x_k \leq x} \left\{ T_k(x_k) * F_{k-1}^S(x - x_k)_{\max} \right\}, k = 2, 3, \cdots, n,$$
$$F_1^S(x)_{\max} = T_1(x),$$
$$F_k^S(0)_{\max} = T_{1.\min} T_{2.\min} \cdots T_{k.\min}, \quad k = 1, 2, \cdots, n.$$

$F_1^S(x)_{\max}$ represents that a software system consists of just one component. Therefore, the optimal trustworthiness of a software system is the component trustworthiness when we allocate development costs x to develop the component. $F_k^S(0)_{\max}$ represents that the trustworthiness of the top k components remains the initial trustworthiness $T_{i.\min}(i = 1, 2, \cdots, k)$ when development costs are 0. Therefore, the optimal trustworthiness of a software system composed of the top k components is $F_k^S(0)_{\max} = T_{1.\min} T_{2.\min} \cdots T_{k.\min}$, $k = 1, 2, \cdots, n$. The algorithm for allocating development costs in sequence structure is shown below as Algorithm 1. The symbols interpretation in Algorithm 1 is as follows.

Double *Ori_T*$[n][e]$: It is used to store the value of $T_k(x)$, $k = 1, 2, \cdots, n, x = 0, 1, \cdots, e$.

Double *Max_T*$[n][e]$: It is used to store the optimal trustworthiness of the software system composed of the top $k(k = 1, 2, \cdots, n)$ components when we allocate development costs $x(x = 0, 1, \cdots, e)$ to develop the top k components. Element *Max_T*$[n][e]$ is the optimal trustworthiness of the software system.

Int $Max\_M[n][e]$: it is used to record development costs of the $k^{th}(k = 1, 2, \cdots, n)$ component when we allocate development costs $x(x = 0, 1, \cdots, e)$ to develop the k^{th} component.

Int $Gain[n]$: It is used to store development costs of each component.

Algorithm 1 Algorithm for Allocating Development Costs in Sequence Structure

Input: development costs e, number of components n, array $Ori\_T[n][e]$

Output: optimal trustworthiness of software system $Max\_T[n][e]$, array $Gain[n]$

```
 1. Initialize all element values of array $Ori\_T[n][e]$ to 0
 2. for $x \leftarrow 0$ to $e$ do
 3.   $Max\_T[1][x] \leftarrow Ori\_T[1][x]$; $Max\_M[1][x] \leftarrow x$;
 4. end for
 5. for $k \leftarrow 2$ to $n$ do
 6.   for $x \leftarrow 0$ to $e$ do
 7.     max $\leftarrow 0$;
 8.     for $s \leftarrow 0$ to $x$ do
 9.       if $Ori\_T[k][s] * Max\_T[k-1][x-s] >$ max
10.       then max $\leftarrow Ori\_T[k][s] * Max\_T[k-1][x-s]$;
              $Max\_M[k][x] \leftarrow s$;
11.       else continue
12.       end if
13.     end for
14.     $Max\_T[k][x] \leftarrow$ max;
15.   end for
16. end for
17. for $k \leftarrow n$ to 1 step $\leftarrow -1$ do
18.   $Gain[k] \leftarrow Max\_M[k][e]$; $e \leftarrow e - Gain[k]$;
19. end for
```

The $Gain[k](k = 1, 2, \cdots, n)$ can be obtained by tracing the element value of $Max\_M[n][e]$. For example, first, we check the element value of $Max\_M[n][e]$. If $Max\_M[n][e] = x_n$, $Gain[n] = x_n$. It means we allocate development costs x_n to develop the n^{th} component, and the rest of development costs $e - x_n$ will be used to develop the top $n - 1$ components. Next, we check the element value of $Max\_M[n-1][e-x_n]$. If $Max\_M[n-1][e-x_n] = x_{n-1}$, $Gain[n-1] = x_{n-1}$. It means we allocate development costs x_{n-1} to develop the $(n-1)^{th}$ component, and the rest of development costs $e - x_n - x_{n-1}$ will be used to develop the top $n - 2$ components. Accordingly, the element values of array $Gain[n]$ can be obtained by tracing element values of array $Max\_M[n][e]$.

Algorithm efficiency analysis: Algorithm 1 includes five for loops. The time complexity of the first for loop is $O(e)$. The third for loop is nested in the second for loop. The fourth for loop is nested in the third for loop. When $k = 2, x = 0$, it only needs one multiplication and one comparison to get the value of $Max\_T[2][0]$. When $k = 2, x = 1$, it needs two multiplication and two comparison to obtain the value of $Max\_T[2][1]$. For $x = 0, 1, \cdots, e$, $1 + 2 + \cdots + (e + 1) = \sum_{x=0}^{e}(x + 1)$ times multiplication and comparison are both needed to get the values of $Max\_T[2][x](x = 0, 1, \cdots, e)$. On the other hand, when k is $2, 3, \cdots, n$, the times of multiplication and comparison are both $\sum_{k=2}^{n}\sum_{x=0}^{e}(x + 1) = (n - 1)(e + 1)(e + 2)/2$. The time complexity of the fifth for loop is $O(n)$. Therefore, The time complexity of the whole system is $O(e + ne^2 + n) = O(ne^2)$. The space complexity is the storage space required by Algorithm 1, including one-dimensional array $Gain[n]$ and two-dimensional array $Max\_T[n][e]$ and $Max\_M[n][e]$. The space complexity of the Algorithm 1 is $O(n + ne + ne) = O(ne)$.

B. MODEL FOR ALLOCATION OF SOFTWARE COMPONENT DEVELOPMENT COSTS IN BRANCH STRUCTURE

Each branch runs with a certain probability [21]. The software system in branch structure is shown in Fig. 2.

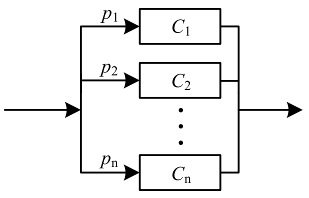


FIGURE 2. Software system in branch structure.

Each component runs with a certain probability. Let $p_k \in (0, 1)$ be the probability of the k^{th} component execution, which represents the average execution times of the k^{th} component. Then, $p_k T_k (k = 1, 2, \cdots, n)$ reflects the impact of each component in branch structure on software trustworthiness. Therefore, the multi-value model for allocation of software component development costs in branch structure is defined as follows:

$$T_B = \max \sum_{k=1}^{n} p_k T_k, \quad s.t. \sum_{k=1}^{n} e_k \leq e, \tag{8}$$

where T_B presents the optimal software trustworthiness in branch structure, and $p_k \in (0, 1)$ denotes the probability of the k^{th} component running, $\sum_{k=1}^{n} p_k = 1$. Therefore, the recursive equations and boundary conditions are defined as follows:

$$F_k^B(x)_{\max} = \max_{0 \leq x_k \leq x}\left\{p_k * T_k(x_k) + F_{k-1}^B(x - x_k)_{\max}\right\}, k = 2, 3, \cdots, n,$$

$$F_1^B(x)_{\max} = p_1 T_1(x),$$

$$F_k^B(0)_{\max} = p_1 T_{1.\min} + p_2 T_{2.\min} + \cdots + p_k T_{k.\min}, k = 1, 2, \cdots, n.$$

$F_1^B(x)_{\max}$ represents that the software system consists of just one component. Theoretically, the optimal trustworthiness of a software system is component trustworthiness when we allocate development costs x to develop the component. However, the development costs need to be allocated to n components. For allocation to the rest of components, we assume that $F_1^B(x)_{\max} = p_1 T_1(x)$. $F_k^B(0)_{\max}$ represents that the trustworthiness of the top k components remains the initial trustworthiness $T_{i.\min}(i = 1, 2, \cdots, k)$ when the development costs are 0. Therefore, $F_k^B(0)_{\max} = p_1 T_{1.\min} + p_2 T_{2.\min} + \cdots + p_k T_{k.\min}$, $k = 1, 2, \cdots, n$. The algorithm for allocating development costs in branch structure is shown as Algorithm 2.

Let **Int** $P[n]$ restore the probability of the component running. The other symbols have the same meaning in Algorithm 1.

Algorithm 2 Algorithm for Allocating Development Costs in Branch Structure

Input: development costs e, number of components n, array $Ori\_T[n][e]$, array $P[n]$

Output: optimal trustworthiness of software system $Max\_T[n][e]$, array $Gain[n]$

```
 1. Initialize all element values of array $Max\_T[n][e]$ to 0
 2. for $x \leftarrow 0$ to $e$ do
 3.   $Max\_TM[1][x] \leftarrow Ori\_T[1][x]$; $Max\_T[1][x] \leftarrow x$;
 4. end for
 5. for $k \leftarrow 2$ to $n$ do
 6.   for $x \leftarrow 0$ to $e$ do
 7.     max $\leftarrow 0$;
 8.       for $s \leftarrow 0$ to $x$ do
 9.         if $P[k]*Ori\_T[k][s]+Max\_T[k-1][x-s] >$ max
10.           then max $\leftarrow P[k]*Ori\_T[k][s]+Max\_T[k-1][x-s]$; $Max\_M[k][x] \leftarrow s$;
11.           else continue
12.           end if
13.       end for
14.     $Max\_T[k][x] \leftarrow$ max;
15.   end for
16. end for
17. for $k \leftarrow n$ to 1 step $\leftarrow -1$ do
18.   $Gain[k] \leftarrow Max\_M[k][e]$; $e \leftarrow e - Gain[k]$;
19. end for
```

Algorithm efficiency analysis: Algorithm 2 includes five for loops. The time complexity of the first for loop is $O(e)$. The third for loop is nested in the second for loop. The fourth for loop is nested in the third for loop. When $k = 2, x = 0$, it only needs one addition and one comparison to get the value of $Max\_T[2][0]$. When $k = 2, x = 1$, it needs two addition and two comparison to obtain the value of $Max\_T[2][1]$. So, when $x = 0, 1, \cdots, e$, $1+2+\cdots+(e+1) = \sum_{x=0}^{e}(x+1)$ times addition and comparison are both needed to get the values of $Max\_T[2][x](x = 0, 1, \cdots, e)$. On the other hand, when k is $2, 3, \cdots, n$, the times of addition and comparison are both $\sum_{k=2}^{n}\sum_{x=0}^{e}(x+1) = (n-1)(e+1)(e+2)/2$. The time complexity of the fifth for loop is $O(n)$. Therefore, The time complexity of the whole system is $O(e + ne^2 + n) = O(ne^2)$. The space complexity is the storage space required by Algorithm 2, including one-dimensional array $Gain[n]$ and two-dimensional array $Max\_T[n][e]$ and $Max\_M[n][e]$. The space complexity of the Algorithm 2 is $O(n+ne+ne) = O(ne)$.

C. MODEL FOR ALLOCATION OF SOFTWARE COMPONENT DEVELOPMENT COSTS IN PARALLEL STRUCTURE

There are two basic types of parallel structure [21]: "and parallel" and "or parallel," as shown in Fig. 3. A software system runs successfully only when all components run successfully. The multi-value model for allocation of software component development costs in "and parallel" structure is the same as that in sequence structure. In "or parallel" structure, the software system runs successfully as long as one component runs successfully.

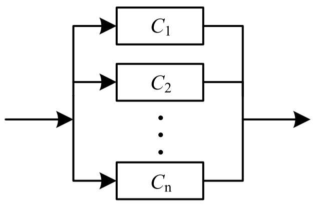


FIGURE 3. Software system in parallel structure.

When the trustworthiness of the k^{th} component is T_k, so the k^{th} component untrustworthiness is $1 - T_k$. Therefore, software untrustworthiness can be presented as $\prod_{k=1}^{n}(1-T_k)$, and software trustworthiness is presented as $1 - \prod_{k=1}^{n}(1-T_k)$. It can be seen that if components have higher trustworthiness, software trustworthiness will be higher. Accordingly, the multi-value model for allocation software component development costs in "or parallel" structure is defined as follows:

$$T_P = \max[1 - \prod_{k=1}^{n}(1-T_k)], \tag{9}$$
$$s.t. \sum_{k=1}^{n} e_k \le e,$$

where T_P presents the optimal software trustworthiness in parallel structure.

Based on (9), the optimal trustworthiness of a software system can be obtained indirectly by $\min \prod_{k=1}^{n}(1-T_k)$. Let $F_k^P(x)_{\min}$ represents the minimum untrustworthiness of the software system when development costs x are allocated to develop the top k components in parallel structure. Therefore, the optimal trustworthiness of the software system is $1 - F_k^P(x)_{\min}$. First, we allocate $x_k(0 \le x_k \le x)$ from x to develop the k^{th} component, and the rest of development costs $x - x_k$ will be used to develop the top $k-1$ components. The optimal allocation method has been attained when we allocate development costs to the top $(k-1)^{\text{th}}$ component. Therefore, recursive equations and boundary conditions are defined as follows:

$$F_k^P(x)_{\min} = \min_{0\le x_k \le x}\left\{[1-T_k(x_k)]*F_{k-1}(x-x_k)_{\min}\right\}, k=2,3,\cdots,n,$$
$$F_1^P(x)_{\min} = 1 - T_1(x),$$
$$F_k^P(0)_{\min} = (1-T_{1.\min})(1-T_{2.\min})\cdots(1-T_{k.\min}), k=1,2,\cdots,n.$$

$F_1^P(x)_{\min}$ represents that the software system consists of just one component. Therefore, the minimum untrustworthiness of software system is 1 minus component trustworthiness when we allocate development costs x to develop the component. $F_k(0)_{\min}$ represents that the untrustworthiness of the top k components remains the initial trustworthiness $T_{i.\min}(i = 1, 2, \cdots, k)$ when the development costs are 0. Therefore, $F_k^P(0)_{\min} = (1 - T_{1.\min})(1 - T_{2.\min}) \cdots (1 - T_{k.\min})$, $k = 1, 2, \cdots, n$. At this time, the optimal trustworthiness of software system is $1 - F_k^P(0)_{\min}$. The algorithm for allocating development costs in parallel structure is shown as Algorithm 3.

Double $Min\_T[n][e]$: It is used to store the minimum untrustworthiness of the software system composed of the top $k(k = 1, 2, \cdots, n)$ components when we allocate development costs $x(x = 0, 1, \cdots, e)$ to develop the top k components. $1 - Min\_T[n][e]$ is the optimal trustworthiness of the software system.

Int $Min\_M[n][e]$: it is used to record development costs of the kth$(k = 1, 2, \cdots, n)$ component when we allocate development costs $x(x = 0, 1, \cdots, e)$ to develop the kth component.

Int $Min\_T[n][e]$: it is used to record the value of the minimum untrustworthiness of the software system. The other symbols have the same meaning in Algorithm 1.

Algorithm 3 Model for Allocating Development Costs in Parallel Structure

Input: development costs e, number of components n, array $Ori\_T[n][e]$

Output: optimal trustworthiness of software system $1 - Min\_T[n][e]$, array $Gain[n]$

```
1.  Initialize all element values of array $Min\_T[n][e]$ to 0
2.  for $x \leftarrow 0$ to $e$ do
3.    $Min\_T[1][x] \leftarrow Ori\_T[1][x]$; $Min\_M[1][x] \leftarrow x$;
4.  end for
5.  for $k \leftarrow 2$ to $n$ do
6.    for $x \leftarrow 0$ to $e$ do
7.      min $\leftarrow 1$;
8.      for $s \leftarrow 0$ to $x$ do
9.        if $(1-Ori\_T[k][s]) * Min\_T[k-1][x-s] <$ min
10.         then min $\leftarrow (1 - Ori\_T[k][s]) * Min\_T[k-1][x-s]$; $Min\_M[k][x] \leftarrow s$;
11.         else continue
12.       end if
13.     end for
14.     $Min\_T[k][x] \leftarrow$ min;
15.   end for
16. end for
17. for $k \leftarrow n$ to 1 step $\leftarrow -1$ do
18.   $Gain[k] \leftarrow Min\_M[k][e]$; $e \leftarrow e - Gain[k]$;
19. end for
```

Algorithm efficiency analysis: Algorithm 3 includes five for loops. The time complexity of the first for loop is $O(e)$. The third for loop is nested in the second for loop. The fourth for loop is nested in the third for loop. When $k = 2$, $x = 0$, it only needs one multiplication and one comparison to get the value of $Min\_T[2][0]$. When $k = 2$, $x = 1$, it needs two multiplication and two comparison to obtain the value of $Min\_T[2][1]$. Therefore, for $x = 0, 1, \cdots, e$, $1 + 2 + \cdots + (e + 1) = \sum_{x=0}^{e}(x + 1)$ times multiplication and comparison are both needed to get the values of $Min\_T[2][x](x = 0, 1, \cdots, e)$. On the other hand, when k is $2, 3, \cdots, n$, the times of multiplication and comparison are both $\sum_{k=2}^{n}\sum_{x=0}^{e}(x + 1) = (n - 1)(e + 1)(e + 2)/2$. The time complexity of the fifth for loop is $O(n)$. Therefore, The time complexity of the whole system is $O(e + ne^2 + n) = O(ne^2)$. The space complexity is the storage space required by Algorithm 3, including one-dimensional array $Gain[n]$ and two-dimensional array $Min\_T[n][e]$ and $Min\_M[n][e]$. The space complexity of the Algorithm 3 is $O(n + ne + ne) = O(ne)$.

D. MODEL FOR ALLOCATION OF SOFTWARE COMPONENT DEVELOPMENT COSTS IN LOOP STRUCTURE

When one component or some components run repeatedly, a loop body will be formed [21], as shown in Fig. 4. We assume that a loop body is denoted by A, and loop times are denoted by t, $t \geq 1$.

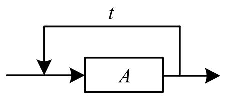


FIGURE 4. Software system in loop structure.

We assume that loop body A is composed of n components: $C_1, C_2, \cdots, C_n$. The structures of a loop body may be sequence structure, parallel structure, or branch structure. The trustworthiness of a loop body A can be obtained by using (7)–(9). We denote the trustworthiness of a loop body by T_L. If the structure of a loop body is a sequence structure, $T_L = T_S$. If the structure of a loop body is a branch structure, $T_L = T_B$. If the structure of a loop body is a parallel structure, $T_L = T_P$. Therefore, the multi-value model for allocation of software component development costs in loop structure is defined as follows:

$$T_L = \max T_A^t, \quad s.t. \sum_{k=1}^{n} e_k \leq e, \tag{10}$$

where T_L presents the optimal software trustworthiness in loop structure.

IV. SOFTWARE COMPONENT DEVELOPMENT COSTS ALLOCATION TOOL

In order to implement the automatic allocation of development costs, it is necessary to develop a tool. Therefore, the web-based software component development costs allocation tool is developed based on Algorithms 1–3. The main function of the tool is to obtain the optimal software trustworthiness and development costs allocation of each component. The user interface of the tool is shown in Fig. 5.

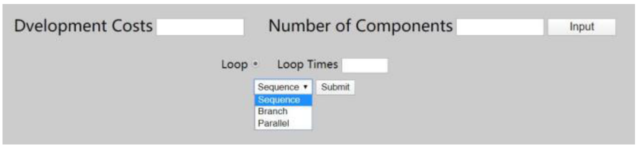
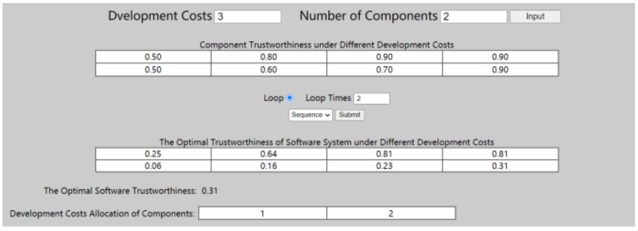


FIGURE 5. **User interface of software component development costs tool.**

First, we need to input development costs e and the number of components n. Next, we click the input button, and an $n \times (e+1)$ empty table will be automatically created and presented on the screen according to the values of e and n. We need to input the different trustworthiness of each component under different development costs in the $n\times(e+1)$ table: $T_1(0), T_1(1), \cdots, T_1(e); \cdots; T_n(0), T_n(1), \cdots, T_n(e)$.

Next, we need to determine whether the structure of the software system is a loop structure. If it is not a loop structure, we can directly select whether the structure is sequence structure, branch structure, or parallel structure. If it is a loop structure, we choose the loop button, and then the loop times will be presented. We also need to select the corresponding structure of the loop body. Finally, we click the submit button, and the optimal trustworthiness of software system under different development costs is presented by an $n \times (e+1)$ table, and the optimal trustworthiness of software system and development costs allocation of each component can be obtained. For example, development costs are 3, and number of components is 2. The structure of the software system is a loop structure. The structure of the loop body is sequence structure and the loop times are 2, as shown in Fig. 6.



Dvelopment Costs 3

Number of Components 2

Component Trustworthiness under Different Development Costs

0.50	0.80	0.90	0.90
0.50	0.60	0.70	0.90

Loop Times 2

The Optimal Trustworthiness of Software System under Different Development Costs

0.25	0.64	0.81	0.81
0.06	0.16	0.23	0.31

The Optimal Software Trustworthiness: 0.31

Development Costs Allocation of Components:

1	2

FIGURE 6. **An example of software component development costs allocation tool.**

V. CASE STUDY

The self-service ticketing system used in our case study consists of seven components, and each component has its own specific function, as shown in Fig. 7. The logon function is implemented by the components C_1 and C_2. Component C_1 requires users to log on through a phone number and a password, and component C_2 requires users to log on through a third-party platform, such as Alipay, WeChat, or QQ. The function of booking a single ticket is implemented by the component C_3. The function of booking group tickets is implemented by the component C_4. The payment function is

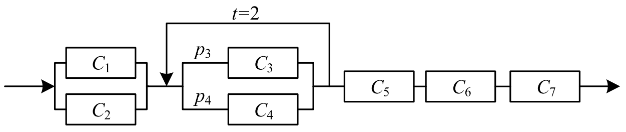


FIGURE 7. **A self-service ticketing system.**

implemented by the component C_5. The function of printing tickets is implemented by the component C_6. The exit function is implemented by the component C_7.

If component C_1 or C_2 runs successfully, users can log on the system. Therefore, the structure between component C_1 and C_2 is "or parallel," and we denote component C_1 and C_2 by component $C_{1,2}$. Once the user logs on the system, there are two options: booking a single ticket, booking group tickets. Thus, components C_3 and C_4 are connected in branch structure and form a loop body together. We denote components C_3 and C_4 by component $C_{3,4}$, and p_k denotes the probability of components $C_k (k = 3, 4)$ running. Components C_5, C_6, and C_7 are connected in sequence structure. Therefore, the overall structure of the self-service ticketing system is sequence structure. The optimal trustworthiness of the system is given as follows:

$$T_S = \max[1 - (1 - T_1)(1 - T_2)](p_3T_3 + p_4T_4)^2T_5T_6T_7.$$

In case study, we assume that the development costs are 10. The complexity, initial trustworthiness, maximum trustworthiness, and corresponding development costs of each component are shown in Table 2.

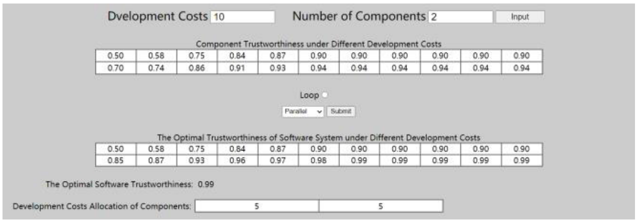
TABLE 2. **The complexity, initial trustworthiness, maximum trustworthiness, and corresponding development costs of each component.**

C_i	f_k	$T_{k.\min}$	$T_{k.\max}$	e_k
C_1	0.54	0.50	0.90	5
C_2	0.30	0.70	0.94	5
C_3	0.20	0.80	0.96	5
C_4	0.10	0.90	0.98	5
C_5	0.15	0.80	0.95	6
C_6	0.40	0.90	0.95	3
C_7	0.10	0.90	0.98	4

The optimal trustworthiness of the system can be obtained by the following four steps:

Step 1): According to the trustworthiness of components C_1 and C_2 under different development costs, the optimal trustworthiness of component $C_{1,2}$ under different development costs can be obtained by using Algorithm 3, as shown in Fig. 8.

Step 2): According to the use probability of each component, as calculated by the research team, $p_3 = 0.2, p_4 = 0.8$. We suppose that $t = 2$. Combined with the trustworthiness of components C_3 and C_4 under different development costs,



Dvelopment Costs 10 Number of Components 2 Input

Component Trustworthiness under Different Development Costs

0.50	0.58	0.75	0.84	0.87	0.90	0.90	0.90	0.90	0.90	0.90
0.70	0.74	0.86	0.91	0.93	0.94	0.94	0.94	0.94	0.94	0.94

Loop

Parallel Submit

The Optimal Trustworthiness of Software System under Different Development Costs

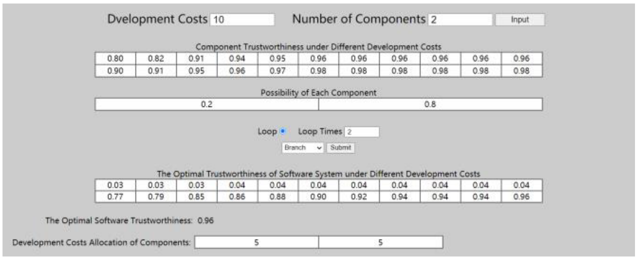
0.50	0.58	0.75	0.84	0.87	0.90	0.90	0.90	0.90	0.90	0.90
0.85	0.87	0.93	0.96	0.97	0.98	0.99	0.99	0.99	0.99	0.99

The Optimal Software Trustworthiness: 0.99

Development Costs Allocation of Components: 5 5

FIGURE 8. Development costs allocation of component $C_{1,2}$.

the optimal trustworthiness of component $C_{3,4}$ under different development costs can be obtained by using Algorithm 2, as shown in Fig. 9.



Dvelopment Costs 10 Number of Components 2 Input

Component Trustworthiness under Different Development Costs

0.80	0.82	0.91	0.94	0.95	0.96	0.96	0.96	0.96	0.96	0.96
0.90	0.91	0.95	0.96	0.97	0.98	0.98	0.98	0.98	0.98	0.98

Possibility of Each Component

0.2	0.8

Loop ● Loop Times 2

Branch Submit

The Optimal Trustworthiness of Software System under Different Development Costs

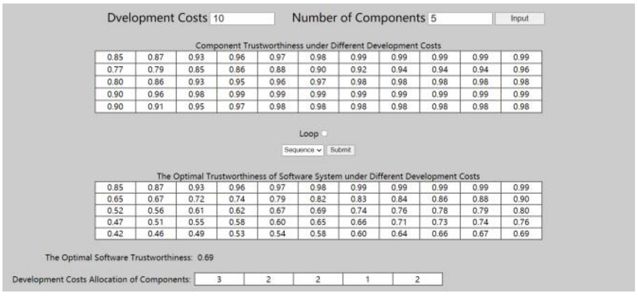
0.03	0.03	0.03	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04
0.77	0.79	0.85	0.86	0.88	0.90	0.92	0.94	0.94	0.94	0.96

The Optimal Software Trustworthiness: 0.96

Development Costs Allocation of Components: 5 5

FIGURE 9. Development costs allocation of component $C_{3,4}$.

Step 3): The optimal trustworthiness of components $C_{1,2}$ and $C_{3,4}$ under different development costs are obtained. Combined with the trustworthiness of components C_5, C_6, and C_7 under different development costs, the optimal trustworthiness of the system and development costs allocation of components $C_{1,2}$, $C_{3,4}$, C_5, C_6, and C_7 can be obtained by using Algorithm 1, as shown in Fig. 10.



Dvelopment Costs 10 Number of Components 5 Input

Component Trustworthiness under Different Development Costs

0.85	0.87	0.93	0.96	0.97	0.98	0.99	0.99	0.99	0.99	0.99
0.77	0.79	0.85	0.86	0.88	0.90	0.92	0.94	0.94	0.94	0.96
0.80	0.86	0.93	0.95	0.96	0.97	0.98	0.98	0.98	0.98	0.98
0.90	0.96	0.98	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
0.90	0.91	0.95	0.97	0.98	0.98	0.98	0.98	0.98	0.98	0.98

Loop

Sequence Submit

The Optimal Trustworthiness of Software System under Different Development Costs

0.85	0.87	0.93	0.96	0.97	0.98	0.99	0.99	0.99	0.99	0.99
0.65	0.67	0.72	0.74	0.79	0.82	0.83	0.84	0.86	0.88	0.90
0.52	0.56	0.61	0.62	0.67	0.69	0.74	0.76	0.78	0.79	0.80
0.47	0.51	0.55	0.58	0.60	0.65	0.66	0.71	0.73	0.74	0.76
0.42	0.46	0.49	0.53	0.54	0.58	0.60	0.64	0.66	0.67	0.69

The Optimal Software Trustworthiness: 0.69

Development Costs Allocation of Components: 3 2 2 1 2

FIGURE 10. Development costs allocation of component $C_{1,2}$, $C_{3,4}$, C_5, C_6, and C_7.

Step 4): As shown in Fig. 10, we need to allocate development costs 3, 2, 2, 1, and 2 to components $C_{1,2}$, $C_{3,4}$, C_5, C_6, and C_7 respectively. According to the trustworthiness of components C_1 and C_2 with development costs 3, the development costs allocation of components C_1 and C_2 can be obtained. In the same way, according to the trustworthiness of components C_3 and C_4 with development costs 2, the development costs allocation of components C_3 and C_4 can be obtained.

Therefore, the optimal software trustworthiness is 0.69. The development costs allocation for each component is shown in Table 3.

TABLE 3. Development costs allocation of each component in multi-value allocation models.

Component	C_1	C_2	C_3	C_4	C_5	C_6	C_7
Development Costs	0	3	0	2	2	1	2

Developers will invest resources to make a detail development plan to improve the component trustworthiness if a component needs to be developed. If the requirement of the component trustworthiness is not high, there is no need in wasting resources to make a development plan, and the component could maintain the initial trustworthiness. Therefore, in [19], the two-value allocation of development costs was proposed. In two-value allocation of development costs, the authors use the array of $[x_1, x_2, \cdots, x_k, \cdots, x_n](x_k \in \{0, 1\}, k = 1, 2, \cdots, n)$ to express the situation of component. 1 means that the component is allocated development costs to achieve the maximum trustworthiness. 0 means there are no development costs allocated to the component. Correspondingly, there are algorithms to allocate development costs to the components according to the different structures of software system.

In this paper, in order to compare our methods with two-value allocation in [19], we made the same inputs in two- value methods for this case. We use two-value allocation algorithms to allocate development costs for the case study in this paper, and the optimal trustworthiness of the software system is 0.55. The development costs allocation for each component is shown in Table 4. The software trustworthiness is 0.69 by using the multi-value allocation algorithms, which can improve (0.69-0.55)/0.55=25% of software trustworthiness.

TABLE 4. Development costs allocation of each component in two-value allocation models.

Component	C_1	C_2	C_3	C_4	C_5	C_6	C_7
Development Costs	0	0	0	0	6	0	4

To sum up, if users have low requirements for software quality, then within the given development costs, using the two-value allocation algorithms can save a lot of resources and improve development efficiency to a certain extent. If users have higher requirements on software trustworthiness, they can consider using the multi-value allocation algorithms of software component development costs. Although we need to spend more resources to make development plans for each component, software trustworthiness can be guaranteed to a certain extent.

VI. CONCLUSION

Software trustworthiness is an important criterion for measuring software quality. In order to improve software trustworthiness and solve the allocation problem of development costs, multi-values model for allocation of software component development costs are established according to four different structures of software system: sequence structure, parallel structure, branch structure, and loop structure. Combined with dynamic programming, the corresponding allocation algorithms are proposed. Moreover, a web-based software component development costs allocation tool is developed to implement automatic development costs allocation of each component. In this paper, the application of a self-service ticketing system indicates that the development costs can be allocated to each component to optimize software trustworthiness. The proposed models can be used not only in simple software system, but also in large complex software system. It can be seen that the proposed algorithms can guide and control the allocation of development costs, which has reference value for software development costs management and software quality management.

In the allocation of development costs, there are lots of challenges, for example, changing development costs for one component may changes development costs for another component. At the same time, we will continue improving the proposed models. The validity of the proposed models needs to be verified in practical projects.

REFERENCES

[1] T. A. Ahanger and A. Aljumah, "Internet of Things: A comprehensive study of security issues and defense mechanisms," *IEEE Access*, vol. 7, pp. 11020–11028, 2019.

[2] P. G. Neumann, "Trustworthiness and truthfulness are essential," *Commun. ACM*, vol. 60, no. 6, pp. 26–28, May 2017.

[3] F. Schneider, *Trust in Cyberspace*, Washington, DC, USA: National Academy Press. 1998.

[4] J. He, Z. Shan, J. Wang, G. Y. Pu; Fang, K. Liu, R. Zhao, and Z. Zhang, "Review of the achievements of major research plan of trustworthy software," (in Chinese), *Bull. Natl. Nat. Sci. Found.*, vol. 32, no. 3, pp. 291–296, 2018.

[5] W. Hasselbring and R. Reussner, "Toward trustworthy software systems," (in Chinese), *Computer*, vol. 39, no. 4, pp. 91–92, Apr. 2006.

[6] H. Tao and Y. Chen, "A new metric model for trustworthiness of software," (in Chinese), *Telecommun. Syst.*, vol. 51, pp. 95–105, Nov. 2010.

[7] H. Tao, H. Wu, and Y. Chen, "An approach of trustworthy measurement allocation based on sub-attributes of software," (in Chinese), *Mathematics*, vol. 7, p. 237, Mar. 2019.

[8] H. Tao, Y. Chen, and J. Pang, "Axiomatic approaches based software trustworthiness measure," in *Proc. Int. Conf. Appl. Syst. Innov. (ICASI)*, May 2015, pp. 135–142.

[9] B. Gu, J. Xu, J. Zhang, Y. Chen, X. Guo, S. Jin, J. Wang, and B. Wang, "An approach to measureing and grading software trust for spacecraft software," *Scientia Sinica Technologica*, vol. 45, no. 2, pp. 221–228, Jan. 2015.

[10] F. Brosig, P. Meier, S. Becker, A. Koziolek, H. Koziolek, and S. Kounev, "Quantitative evaluation of model-driven performance analysis and simulation of component-based architectures," *IEEE Trans. Softw. Eng.*, vol. 41, no. 2, pp. 157–175, Feb. 2015.

[11] T. Hongwei, C. Yixiang, W. Hengyang, and D. Rumei, "A survey of software trustworthiness measurements," *Int. J. Performability Eng.*, vol. 15, no. 9, p. 2364, 2019.

[12] J. Li, M. Li, D. Wu, and H. Song, "An integrated risk measurement and optimization model for trustworthy software process management," *Inf. Sci.*, vol. 191, pp. 47–60, May 2012.

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

[14] Y. Ma, Y. Chen, and B. Gu, "An attributes-based allocation approach of software trustworthy degrees," in *Proc. IEEE Int. Conf. Softw. Quality, Rel. Secur. Companion*, Aug. 2015, pp. 89–94.

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

[16] T. Vale, I. Crnkovic, E. S. de Almeida, P. A. D. M. S. Neto, Y. C. Cavalcanti, and S. R. D. L. Meira, "Twenty-eight years of component-based software engineering," *J. Syst. Softw.*, vol. 111, pp. 128–148, Jan. 2016.

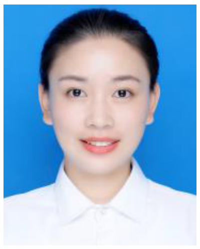
[17] B. Wang, Y. Chen, S. Zhang, and H. Wu, "Updating model of software component trustworthiness based on users feedback," *IEEE Access*, vol. 7, pp. 60199–60205, 2019.

[18] D. Huang, "Component-based software trustworthiness measurement and allocation model," (in Chinese), M.S. thesis, School Comput. Sci. Technol., Huaibei Normal Univ., Huaibei, China, 2019.

[19] Y. Ma, M. Wang, and W. Zhou, "Trustworthiness-based development costs allocation algorithm of modular software," (in Chinese), *Comput. Eng. Sci.*, vol. 42, no. 6, pp. 50–57, 2020.

[20] C. J. Dale and A. Winterbottom, "Optimal allocation of effort to improve system reliability," *IEEE Trans. Rel.*, vol. R-35, no. 2, pp. 188–191, Jun. 1986.

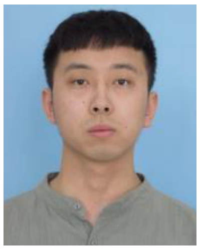
[21] W. F. Lu Xu and L. Jia, "An approach of software reliability evolution in open environment," (in Chinese), *Chin. J. Computers.*, vol. 33, no. 3, pp. 452–461, 2010.



MENGYUE WANG was born in Bengbu, Anhui, China, in 1996. She received the B.E. degree from the School of Computer Science and Technology, Huaibei Normal University, Anhui, in 2017, where she is currently pursuing the master's degree in software engineering. She has participated in the National Natural Science Foundation Project. Her research interests include trade-off analysis of software trustworthy attributes, software trustworthiness measurement, and software development costs allocation.

YANFANG MA was born in Daqing, Heilongjiang, China, in 1978. She received the Ph.D. degree in computer science and technology from East China Normal University, China, in 2010. She has completed the National Natural Science Foundation of China, in 2017, and the Natural Science Foundation of Anhui Province, in 2015 and 2019. She is currently supported by the Natural Science Foundation for Colleges and Universities of Anhui Province. She is also a Professor with the School of Computer Science and Technology, Huaibei Normal University, Anhui, China. Her research interests include formal semantic model, formal method, and software trustworthiness measurement and allocation.

GUANRU LI was born in Hefei, Anhui, China, in 1997. He received the B.E. degree from the School of Computer Science and Technology, Huaibei Normal University, Anhui, in 2018, where he is currently pursuing the master's degree in software engineering. His research interests include software quality, software testing, and trust evaluation.



WEI ZHOU was born in Tianchang, Anhui, China, in 1996. He received the B.E. degree from the School of Computer Science and Technology, Huaibei Normal University, Anhui, in 2018, where he is currently pursuing the master's degree in software engineering. His research interests include trustworthy service, software trustworthiness measurement, cloud computing, and information security.

LIANG CHEN was born in Yancheng, Jiangsu, China, in 1977. He received the Ph.D. degree in computational mathematics from Shanghai University, China, in 2013. He has completed several funds from Anhui province. He is currently supported by the Foundation for Excellent Young Talents in Universities of Anhui Province. He is currently a Professor with the School of Mathematical Sciences, Huaibei Normal University, Anhui, China. His research interests include matrix computation, numerical optimization, and their applications.