



Article

Improved Allocation and Reallocation Approaches for Software Trustworthiness Based on Mathematical Programming

Hongwei Tao ¹, Lianyou Fu ¹, Yixiang Chen ^{2,*}, Lin Han ³ and Xiao Wang ¹

¹ College of Computer and Communication Engineering, Zhengzhou University of Light Industry, Zhengzhou 450002, China; hongweitao@zzuli.edu.cn (H.T.); fly13233761631@163.com (L.F.); pandaxiaoxi@gmail.com (X.W.)

² MoE Engineering Center for Software/Hardware Co-Design Technology and Application, East China Normal University, Shanghai 200062, China

³ Henan Province Supercomputing Centre, Zhengzhou University, Zhengzhou 450002, China; hanlin@zzu.edu.cn

* Correspondence: yxchen@sei.ecnu.edu.cn; Tel.: +86-021-6223-525

Abstract: Software trustworthiness allocation and reallocation are the symmetry of software trustworthiness measure. They can provide the optimization scheme for trustworthiness development and improvement, according to the requirements. The existing allocation and reallocation models do not consider the absolute majority of software trustworthiness classification; therefore, they cannot be very accurate in the allocation and reallocation of software trustworthiness. In this paper, improved allocation and reallocation models are constructed, which can resolve the above problem, and their polynomial solving algorithms are designed. At the same time, a demonstration application of the improved models and algorithms is given, and the trustworthiness enhancement specification of spacecraft software, based on factory reports, is established, including trustworthiness development specification and trustworthiness improvement specification. This enhancement specification provides a scientific and reasonable theory and method for the delivery acceptance of spacecraft software, from qualitative to quantitative grading acceptance, and furnishes a standard guarantee for the trustworthy development and improvement of such software.

Keywords: software trustworthiness; allocation approach; reallocation approach; trustworthiness enhancement specification; trustworthiness development specification; trustworthiness improvement specification



Citation: Tao, H.; Fu, L.; Chen, Y.; Han, L.; Wang, X. Improved Allocation and Reallocation Approaches for Software Trustworthiness Based on Mathematical Programming. *Symmetry* **2022**, *14*, 628. <https://doi.org/10.3390/sym14030628>

Academic Editors: Clemente Cesarano and Ioannis Dassios

Received: 24 January 2022

Accepted: 18 March 2022

Published: 21 March 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Software is an indispensable technical product in human life. With the increasing scale of software, it is inevitable that software products contain many known and unknown defects when they leave the factory. After the defects are triggered, they often cause serious losses and negative social impacts [1]. Reference [2] is titled “Fatal Bugs: Disasters and Inspirations of Software Defects”. It shows readers the Patriot missile, Mars probe, Ariane 5 rocket, Toyota’s pedal door, Therac-25 medical accident, 2003 blackouts in the northeastern United States, South Korea’s digital budget accounting system and other safety-related areas, which have led to major safety incidents and caused significant material losses due to software defects. All these have made the issue of software trustworthiness increasingly prominent, so that people are now paying more and more attention to software trustworthiness. Governments, large scientific research institutions and software companies have successively put forward targeted research plans. For example, in China, the State Council has listed the high-performance trustworthy computer, supporting the development of the trustworthy software industry, in the key areas of the Outline of National Medium and Long Term Science and Technology Development Program (2006–2020); the 863 plan has launched the key project of “High-Credible Software Production Tools and Integrated Environment”; the National Natural Science Foundation of China has organized the major

research program of “Basic Research on Trustworthy Software”. In America, the National Software Development Strategy has put the development of trustworthy software at the forefront. In Europe, a research program, called “Open Trusted Computing” was launched to develop open-source trustworthy computing software; both the Fifth Framework Plan and the Sixth Framework Plan of the European Union regard highly trustworthy software as the focus of software technology development. Constructing trustworthy software has become an important trend and inevitable choice for the development and application of modern software technology. Software trustworthiness describes the ability of software behavior and results, to accommodate user expectations. The measurement and evaluation of software trustworthiness is able to offer the basis for the improvement of the software trustworthiness development, which is listed as the first core scientific issue in the major research plan of “Basic Research on Trustworthy Software” [1]. Software trustworthiness enhancement is also one of the core research issues of software trustworthiness [3], which realizes the transformation of software trustworthiness development and improvement, from qualitative to quantitative.

As mentioned above, the software trustworthiness measure is used to calculate software trustworthiness through the given attribute values and their weight values [4]. A number of measures oriented to trustworthy attributes are built. Deng et al. established a structural equation model for software trustworthiness evaluation, to obtain the weights of trustworthy attributes and the relationship among them [5]. Gene et al. used cognitive task analysis methods and qualitative analysis techniques to propose a descriptive model of computer code trustworthiness, which includes three trustworthy factors: performance, transparency, and reputation [6]. To help users select the most trustworthy available alternatives, Tania et al. investigated whether a quality model is able to provide values useful for expressing trustworthy attributes [7]. A theoretical model for evaluating code trustworthiness and code reuse is proposed by Gene et al. [8]. This model characterizes a five-step process of how programmers perceive computer code, consisting of acquisition, initial viewing, in-depth viewing, merging, and reevaluating. Zhihan et al. built an evaluation framework for a cyber physical system, on the basis of the knowledge of machine learning, designing an online rank algorithm, and proposing a trustworthy intelligent prediction model [9]. Cristiano et al. used coding practices to describe the trustworthiness of web applications from a security perspective. Their method is instantiated for the concrete case of input validation practices and contains a quality model to calculate trustworthiness values that can be used to compare different applications, or different code elements in the same application [10]. Medeiros et al. conducted a comprehensive study of detecting/predicting vulnerable code, using software metrics and machine learning algorithms [11]. Their main result is that using machine learning algorithms over software metrics helps to identify vulnerable code units, with relatively high confidence in security-critical software systems, but they do not help low critical or non-critical systems, mainly due to the unbalanced nature of their data set. Nádia et al. proposed an approach for assessing and benchmarking software trustworthiness. Their approach is based on software metrics that can be used as indicators of software security [12]. In order to more strictly measure software trustworthiness and carry out the theoretical validation of models, Tao et al. used axiomatic approaches to evaluate software trustworthiness, give the expected properties of software trustworthiness measures and present a series of models satisfying these properties [13–15]. Khan et al. analyzed the dependability and trustworthiness for the component-based software development. Their results show that the component with the most reusability will prove to be more dependable and trustworthy [16]. Gul et al. proposed the loss speed index, to integrate the most important variables of software security, and a software trustworthy security growth model was built, based on this index. This model can be applied to predict the vulnerability severity/type [17]. Muhammad et al. used the values assigned to test case results to classify software trustworthiness [18]. The rating strategy contains test strategies imposed, completeness of system test execution, test iterations, test case priority and test case results, for each iteration. Wang et al. evaluated software

trustworthiness from three aspects: process entity, behavior and products. They propose a software process trustworthiness model, including 37 trustworthiness principles, 182 process entities and behaviors, and 108 artifacts as evidence [19]. Ogunniye et al. developed a prototype e-recruitment system, to explore users' perceptions of algorithmic systems. They suggest that the data and reasoning behind the algorithmic results must be interpreted in order to improve users' perception of the fairness, reliability and transparency of e-recruitment systems, and to build trust in the systems [20]. Both references [21,22] use the fuzzy comprehensive evaluation method to assess software trustworthiness, but there is a difference in the methods for selecting trustworthy attribute sets and calculating trustworthy attribute weights. Reference [21] built the trustworthy attribute set through inviting experts and computing the trustworthy attribute weights, according to information entropy. Reference [22] determined the trustworthy attribute set by the authors in advance, and obtained the trustworthy attribute weights by the rough set theory and expert opinion. Zhang et al. presented a real-time trust measurement theory based on a non-interfering model [23]. In their theory, system calls are handled as atomic actions, and system call sequences are structured as the real behavior of the process. The theoretically expected behavior is calculated according to the mutual non-interference relationship between the security domains, to which the system call belongs in the actual behavior. The trustworthiness of a process is assessed through determining whether actual and theoretical expected behavior deviates. A trustworthiness assessment, on the basis of factors affecting software behavior, is presented by Tian et al. [24]. The behavioral trajectories are represented by a behavioral trajectory matrix, converted to a grayscale image. These images are then used to train a deep residual network to classify software behavior. Ji et al. introduced a Noisy-OR Gate Bayesian Network model to obtain the joint probability distribution (JPD) of target risk systems, in the probabilistic evaluation of construction risk [25]. This model requires only connection probabilities with high availability and reliability as the prior knowledge, thereby greatly reducing the dimensionality of risk factors, while preserving the reasoning ability of JPD. Ogundoyin et al. used a fuzzy analytic hierarchy process technique to identify and prioritize trust parameters in fog computing [26]. Their results show that quality of service is the best prioritized parameter that the service requester can use to assess the trust level of a service provider, followed by quality of security, with recommendations being the worst. It is also revealed that social relationship is the highest-ranking trust parameter that a service provider can use to determine the level of truthfulness of a service requester; past reputation is the least considered. Sahu et al. applied fuzzy method, neural network, fuzzy-neural network and neural-fuzzy, to predict the reliability of the dataset retrieved from John Musa of bell laboratories [27]. Their results indicate the fuzzy-neural method is the best among all the proposed methods. In the fuzzy-neural method, the Levenberg–Marquardt algorithm is used to train the neurons. Sahu et al. also presented a review-based study, to evaluate the previously established methodologies for reliability prediction [28]. Al-Mejibli et al. first identified security risk factors, which affect usability, as well as the security of a healthcare web application, and then calculated the weight of each security factor using Fuzzy AHP [29]. Their study also concludes that user satisfaction is the most crucial factor among all the six security risk factors. Marshall et al. presented a range of models for tool verification and evaluated them based on estimated risk and cost for end-user and provider [30]. Maza et al. proposed a governance framework, based on blockchain, for building trustworthy software [31]. This framework enables transparency and auditability, by allowing activities to be recorded, monitored, and analyzed throughout the application development life cycle. Buraga et al. established an OWL 2 ontology that can provide a high-level machine-processable description of the Database Management Systems (DBMS) domain [32]. This work is intended to facilitate the proper execution of various software engineering processes and database-centric administrative tasks. In addition, it can be applied to improve the decision process, to identify/select the appropriate DBMS, on the basis of specific requirements. Riehle et al. discussed the importance of ensuring that software developers adopt measurable approaches to making data-related

design decisions [33]. Their results show that the more accurate the data type design of a program is, the less complex the algorithm using instances of these data types is. Xu built a data-driven trust evaluation model on the basis of perceptual sources [34]. In this model, the monitoring module is taken as the evaluation unit, and the relay node completes the trust evaluation of the sensing node in its monitoring module. A direct trust calculation is achieved by sensing the relationship between the nodes' own data, and recommendation trust is calculated by monitoring the relationship between neighboring nodes in the module. The comprehensive trust of the perception node is the output, by combining with the historical trust. Novikova et al. presented a novel algorithm to compute an integral trustworthiness risk score, based on privacy, reliability, resilience, safety, and CIA risk scores, and the resulting trustworthiness risk score can be further transformed to trustworthiness ratings [35]. Alzahari et al. gave a new automatic elicitation method, namely, a library of the trust requirements of blockchain applications [36]. A novel trust taxonomy is identified to classify and analyze the associations between trust factors and their attributes, and a proof-of-concept prototype tool is developed to implement the method. Choudhary et al. presented a software readiness rating, on the basis of the combination of weighting methods, and the combination weights are computed, depending on experiments, expert judgments, and statistical calculations [37].

Software trustworthiness allocation is the symmetry of the software trustworthiness measurement. It is to determine the attribute values, according to the user's expectation of the software, which is used to guide the design of the optimal value of each trustworthiness attribute [38]. Software trustworthiness reallocation calculates the optimal sum of the increase in all the trustworthy attribute values, based on the improved users' expectation, which is applied to guide the improvement of software trustworthiness [39]. The determination of software trustworthy level requires software to satisfy not only the requirements of the software trustworthiness metric value, but also the requirements of the trustworthy attribute value. For two software with the same trustworthiness metric value, if a certain attribute of one software does not meet the trustworthy attribute value requirements, then its trustworthy level will be reduced. At present, the existing allocation and reallocation methods do not involve the above situation [38,39]; that is, they do not consider the absolute majority of software trustworthiness classification [3,40], so they do not very accurately conduct the software trustworthiness allocation and reallocation. On the other hand, the current research only demonstrates the effectiveness of their methods with simple examples, and does not go deep into the practical application of the methods.

Research question: Is it possible to build new allocation and reallocation models based on mathematical programming methods by adding the constraint of the absolute majority of software trustworthiness classification? If these new models can be constructed, can polynomial time algorithms be designed for them? If polynomial time algorithms can be given, do these algorithms have practical application scenarios?

The research goals are to construct software trustworthiness allocation and reallocation models, involving an absolute majority of software trustworthiness classification, based on a mathematical programming method, design polynomial time solution algorithms for these improved models, and give their demonstration application.

The main contributions of the paper consist of putting forward improved methods for software trustworthiness allocation and reallocation, constructing improved allocation and reallocation models, involving the absolute majority of trustworthiness classification, and designing polynomial time algorithms for these improved models. In these algorithms, the Newton iteration algorithms are used to initially allocate or reallocate software trustworthiness, and then, according to the absolute majority constraint, the final allocation or reallocation is made based on attribute weights. At the same time, these improved methods are used to build trustworthiness enhancement specification for spacecraft software, based on factory reports, consisting of software trustworthiness development specification and software trustworthiness improvement specification. The contributions in this paper can fill the previously mentioned research gaps.

The rest of the paper is organized as follows. In Section 2, we describe a software trustworthiness measurement model used in [40] and a software trustworthiness classification model proposed in [3]. An improved allocation approach for software trustworthiness is introduced in Section 3, including the improved allocation model, based on mathematical programming and its polynomial-time algorithm. An improved reallocation approach for software trustworthiness is given in Section 4, consisting of the improved reallocation model and its polynomial time algorithm. In Section 5, we establish the trustworthiness enhancement specification for spacecraft software, based on factory reports, according to the improved allocation and reallocation approaches. We present the discussion in Section 6. The conclusions and future work come in the last section.

2. Software Trustworthiness Measurement and Classification

2.1. Software Trustworthiness Measurement Model

Axiomatic approaches formally describe the empirical understanding of software attributes by defining the desired properties of numerical relationship systems in software measures [41,42]. They can provide accuracy and a formal basis for the quantification of software attributes, which are the desired measurement activities. In order to measure software trustworthiness more strictly and validate the model theoretically, we once used axiomatic approaches to assess software trustworthiness, on the basis of decomposition of attribute, proposed seven expected properties, and established a software trustworthiness model, satisfying these seven properties [14]. In view of the particularity of spacecraft software, the simplified version, shown in Definition 1, was applied to measure the trustworthiness of spacecraft software [3,40].

Definition 1. (Simplified Software Trustworthiness Measurement Model based on Attribute Decomposition [3,40])

$$\begin{cases} T = \prod_{i=1}^n y_i^{\alpha_i} \\ y_i = \prod_{j=1}^{n_i} x_{ij}^{\beta_{ij}} \end{cases}$$

where

1. T : trustworthy degree of software;
2. n : number of trustworthy attributes;
3. $y_i (1 \leq i \leq n)$: trustworthy value of the i -th trustworthy attribute;
4. $\alpha_i (1 \leq i \leq n)$: weight value of the i -th trustworthy attribute, satisfying $0 \leq \alpha_i \leq 1$, $\sum_{i=1}^n \alpha_i = 1$.
5. n_i : number of trustworthy sub-attributes that comprise the i -th trustworthy attribute;
6. x_{ij} : trustworthy value of the j -th sub-attribute of the i -th trustworthy attribute;
7. β_{ij} : weight value of the j -th sub-attribute of the i -th trustworthy attribute, such that $0 \leq \beta_{ij} \leq 1$, $\sum_{j=1}^{n_i} \beta_{ij} = 1$.

This simplified measurement model not only satisfies the seven properties, but also complies with Cannikin's law [3]. At the same time, its effectiveness is verified through the trustworthiness evaluation of 23 spacecraft software. Validation results show that it can effectively assess the trustworthiness of spacecraft software and find deficiencies in the development process [3,14,40].

2.2. Software Trustworthiness Classification Model

The software trustworthiness classification model is used to divide the software trustworthiness into several grades, according to the software trustworthiness measurement results, and its desirable properties are as follows.

Uniqueness of trustworthy level [3,40]. The levels evaluated by the software trustworthiness classification model should completely cover all possible distribution intervals of

the software trustworthiness measurement values. Each value of software trustworthiness can only be assessed as one level, which ensures the uniqueness of its trustworthy level. In addition, the evaluation of the trustworthy level in the classification model should require that the software not only meets the requirements of the software trustworthiness measurement, but also meets the requirements of the trustworthy attribute. For two software with the same measurement result, if one of the software does not meet the trustworthy attribute requirements, then its trustworthy level will be reduced.

Threshold [3,40]. Threshold means that when the software trustworthiness is to reach a certain level, the trustworthy attributes must reach the lowest value required at this level. This property indicates that the value of the trustworthy attribute should not be too low, and it is possible to increase the trustworthy level of software by improving the trustworthy attribute with the lowest value.

Absolute majority [3,40]. Absolute majority refers to the software trustworthiness reaching a certain level; at least two-thirds of the trustworthy attribute values must reach this level. Assume that there are n trustworthy attributes, then the number of trustworthy attributes with this level should be not less than $\lceil n \times 2/3 \rceil$. Therefore, the number of trustworthy attributes below this level should not exceed $n - \lceil n \times 2/3 \rceil$. This property is used to describe that the trustworthy level is likely to reach a certain level, only when the $2/3$ majority of the trustworthy attribute values achieve this level.

A software trustworthiness classification model is established in Table 1 [3]. This classification model is constructed based on software trustworthiness measurement results, ranging between 1 and 10. It is easy to prove that this model satisfies the above three properties. Considering that the bigger the software trustworthiness is, the more difficult it is to improve it, and the higher the trustworthy level is, the higher the requirement of the trustworthy attribute value is, in this model, the classification interval is not equidistant. The value interval of increasing the trustworthy level from the lowest level to the top is approximately decreasing, according to the golden ratio.

Table 1. Software trustworthiness classification model [3].

Software Trustworthiness Measurement Requirements	Trustworthy Attribute Requirements	Trustworthy Level
$9.50 \leq T$	<ol style="list-style-type: none"> 1. The number of trustworthy attributes with a value lower than 9.50 shall not exceed $n - \lceil n \times 2/3 \rceil$ 2. No trustworthy attribute with a value of less than 8.50 	V
$8.50 \leq T < 9.50$ or $9.50 \leq T$ and cannot be rated as level V	<ol style="list-style-type: none"> 1. The number of trustworthy attributes with a value lower than 8.50 shall not exceed $n - \lceil n \times 2/3 \rceil$ 2. No trustworthy attribute with a value of less than 7.00 	IV
$7.00 \leq T < 8.50$ or $8.50 \leq T$ and cannot be rated as level IV or above	<ol style="list-style-type: none"> 1. The number of trustworthy attributes with a value lower than 7.00 shall not exceed $n - \lceil n \times 2/3 \rceil$ 2. No trustworthy attribute with a value less than 4.50 	III
$4.50 \leq T < 7.00$ or $7.00 \leq T$ and cannot be rated as level III or above	<ol style="list-style-type: none"> 1. The number of trustworthy attributes with a value less than 4.50 is not more than $n - \lceil n \times 2/3 \rceil$ 2. No trustworthy attribute with a value less than 1 	II

Table 1. Cont.

Software Trustworthiness Measurement Requirements	Trustworthy Attribute Requirements	Trustworthy Level
$T < 4.50$ or $4.50 \leq T$ and cannot be rated as level II or above	No trustworthy attribute with a value less than 1	I

3. Improved Allocation Approach for Software Trustworthiness

Assuming that the trustworthy attribute value is positively correlated with the software development cost, in order to minimize the development cost, the improved allocation model, based on the model given in Definition 1, is defined as a mathematical programming model, as shown in Definition 2.

Definition 2. (Improved allocation model for software trustworthiness)

$$\begin{aligned} & \min \sum_{i=1}^n y_i \\ \text{subject to } & \begin{cases} T = \prod_{i=1}^n y_i^{\alpha_i} \geq T_0 \\ 1 \leq y_0 \leq y_i \leq 10 \\ |\{i | y_i < y'_0\}| \leq (n - \lceil n \times 2/3 \rceil) \end{cases} \end{aligned}$$

where

1. T : trustworthy degree of software;
2. T_0 : minimum degree that the software needs to reach;
3. n : number of trustworthy attributes;
4. $y_i (1 \leq i \leq n)$: value of the i -th trustworthy attribute;
5. y_0 : minimal value that all the trustworthy attributes must achieve, in the software trustworthiness classification model given in Table 1, $y_0 = 1$ for level I software, $y_0 = 1$ for level II software, $y_0 = 4.5$ for level III software, $y_0 = 7$ for level IV software, and $y_0 = 8.5$ for level V software;
6. y'_0 : minimal value of next-level trustworthy attributes, in the software trustworthiness classification model given in Table 1, $y'_0 = 1$ for level I software, $y'_0 = 4.5$ for level II software, $y'_0 = 7$ for level III software, $y'_0 = 8.5$ for level IV software, and specify $y'_0 = 9.5$ for level V software;
7. $\alpha_i (1 \leq i \leq n)$: weight value of the i -th trustworthy attribute, such that $0 \leq \alpha_i \leq 1$, $\sum_{i=1}^n \alpha_i = 1$.

Sensitivity analysis of T , on the basis of the method given in reference [3], shows that the attribute sensitivity is equal to the attribute weight, and the larger the weight is, the more important it is to improve the software trustworthiness, so it is necessary to increase the value of the most important attribute as much as possible. At the same time, in order to simplify the allocation process, the following growth function of the trustworthy attribute value is defined.

Definition 3. (Trustworthy Attribute Value Growth Function [38]).

$$y_i = y_0 + k \times (\alpha_i - \alpha_{\min})$$

where

1. n : number of trustworthy attributes;
2. $y_i (1 \leq i \leq n)$: trustworthy attribute values;
3. y_0 : minimum value that all the trustworthy attributes must reach;

4. $\alpha_i (0 \leq i \leq 1)$: weight value of the i -th trustworthy attribute, satisfying $0 \leq \alpha_i \leq 1$, $\sum_{i=1}^n \alpha_i = 1$, $\alpha_{\min} = \min_{1 \leq i \leq n} \{\alpha_i\}$;
5. k : growth rate of the trustworthy attribute value with $0 \leq k$.

Substitute the trustworthy attribute value growth function into the improved allocation model for software trustworthiness, and the improved allocation model for software trustworthiness, based on the trustworthy attribute value growth function, can be obtained.

Definition 4. (Improved Allocation Model for Software Trustworthiness based on Trustworthy Attribute Value Growth Function)

$$\text{minksatisfy} \begin{cases} T = \prod_{i=1}^n (y_0 + k \times (\alpha_i - \alpha_{\min}))^{\alpha_i} \geq T_0 \\ 1 \leq y_0 \leq y_0 + k \times (\alpha_i - \alpha_{\min}) \leq 10 \\ |\{i | y_0 + k \times (\alpha_i - \alpha_{\min}) < y'_0\}| \leq (n - \lceil n \times 2/3 \rceil) \end{cases}$$

where

1. k : growth rate of the trustworthy attribute value, such that $0 \leq k$.
2. y_0 : minimum value that all the trustworthy attributes must reach, the meaning is the same as in Definition 1;
3. y'_0 : minimum value of next-level trustworthy attributes must achieve. The meaning is the same as in Definition 1;
4. $\alpha_i (1 \leq i \leq n)$: trustworthy attribute weight values, such that $0 \leq \alpha_i \leq 1$, $\sum_{i=1}^n \alpha_i = 1$, $\alpha_{\min} = \min_{1 \leq i \leq n} \{\alpha_i\}$;
5. T_0 : minimum degree that the software needs to achieve.

Software trustworthiness allocation is the process of assigning trustworthiness to software trustworthy attributes through their weight values. Therefore, the software trustworthiness allocation, on the basis of the growth function of the trustworthiness attribute value, is actually looking for the minimum k within the interval $[0, \frac{10-y_0}{\alpha_{\max}-\alpha_{\min}}]$, such that

$$\begin{cases} \prod_{i=1}^n (y_0 + k \times (\alpha_i - \alpha_{\min}))^{\alpha_i} \geq T_0 \\ |\{i | y_0 + k \times (\alpha_i - \alpha_{\min}) < y'_0\}| \leq (n - \lceil n \times 2/3 \rceil), \end{cases}$$

where $0 \leq \alpha_i \leq 1$, $\sum_{i=1}^n \alpha_i = 1$, $\alpha_{\max} = \max_{1 \leq i \leq n} \{\alpha_i\}$.

Let $N = \{1, 2, \dots, n\}$, $P = \{i | i \in N \wedge y_i \geq y'_0\}$. The solution process can be divided into the following two cases:

- (1) When $y_0 \geq T_0$, take the $\lceil n \times 2/3 \rceil$ trustworthy attributes with the largest weight and make their values y'_0 , and the values of the remaining attributes y_0 , so that the allocation conditions can be satisfied.
- (2) When $y_0 < T_0$, let

$$f(k) = \prod_{i \in N - M^{>10}} [y_0 + k(\alpha_i - \alpha_{\min})]^{\alpha_i} \prod_{i \in M^{>10}} y_i^{\alpha_i} - T_0$$

where $M^{>10}$ is a set, whose initial value is \emptyset . In the following, the idea of the reallocation method given in [39] is borrowed for initial allocation. With $k = 0$ as the initial iteration value, the approximate root k , satisfying $f(k) = 0$, is obtained by the Newton iteration method. k is substituted into the trustworthy attribute value growth function. If there are attribute values greater than 10, the subscript set of the attribute values over 10 is included in $M^{>10}$, and the new set is still denoted as $M^{>10}$. Sort the attributes subscripted in $M^{>10}$ in descending order of weight. The attribute

with the largest weight is set to 9.90, the attribute with the second largest weight is set to 9.85, and the attribute with the third largest weight is set to 9.80, and so on. However, when setting the attribute values, it is necessary to ensure that they meet $y_0 \leq y_i$ ($i \in M^{>10}$). The initial value of the attribute with the largest weight value and the step size can be changed as needed. Going to the next round of the calculation, the Newton iteration method is applied to find a new approximate root k , satisfying $f(k) = 0$, and a similar method is used to update the value of $M^{>10}$ and the values of the attributes subscripted in $M^{>10}$. When the set $M^{>10}$ is no longer updated, the attribute value growth function is used to compute the value of the attributes with subscripts in $N - M^{>10}$, and the values of the attributes with subscripts in $M^{>10}$ are obtained by the above setting rules. Denote the trustworthy attribute values obtained as y_1, y_2, \dots, y_n in turn, if the following condition

$$R = |\{i | i \in N \wedge y_i < y'_0\}| \leq (n - \lceil n \times 2/3 \rceil) \quad (1)$$

is satisfied, then y_1, y_2, \dots, y_n are the allocation results. Otherwise, select the attribute whose subscript is in R and has the largest weight value in turn, set its value to y'_0 , and delete its subscript from R until condition (1) is satisfied.

The above specific solution process is shown in Algorithm 1.

Algorithm 1 Algorithm for allocating software trustworthiness: for given T_0, y_0 and weight values of trustworthy attributes $\alpha_1, \alpha_2, \dots, \alpha_n$, output the assigned trustworthy attribute values y_1, y_2, \dots, y_n .

Input: $T_0, y_0, \alpha_1, \alpha_2, \dots, \alpha_n$

Output: $y_j (j = 1, 2, \dots, n)$

1. Initialize $N = \{1, 2, \dots, n\}, M^{>10} = \emptyset, Q = \emptyset, \Delta = 0$;
 2. **if** $y_0 \geq T_0$ **then**
 3. **while** $|Q| \leq \lceil n \times 2/3 \rceil$ **do**
 4. $\alpha_0 = 0, j = 0$
 5. **for** $i \in N$ **do**
 6. **if** $\alpha_i > \alpha_0$ **then**
 7. $\alpha_0 = \alpha_i, j = i$;
 8. **end if**
 9. **end for**
 10. $Q = Q \cup \{j\}, N = N - \{j\}$;
 11. **end while**
 12. **return** $y_j = y_0 (j \in \{1, 2, \dots, n\} - Q), y_j = y'_0 (j \in Q)$;
 13. **else**
 14. **while** $N - M^{>10} \neq \emptyset$ **do**
 15. $f(k) = \prod_{i \in N - M^{>10}} [y_0 + k(\alpha_i - \alpha_{\min})]^{\alpha_i} \prod_{i \in M^{>10}} y_i^{\alpha_i} - T_0$;
 16. Taking $k = 0$ as the initial value of the iteration, $m = 1000$ as the maximum number of iterations, 10^{-10} as the error, the Newton iteration method is used to find the approximate root k that satisfies $f(k) = 0$;
 17. $M = \emptyset$;
 18. **for** $i \in N - M^{>10}$ **do**
 19. $y_i = y_0 + k(\alpha_i - \alpha_{\min})$;
 20. **if** $y_i > 10$ **then**
 21. $M = M \cup \{i\}$
 22. **end if**
 23. **end for**
 24. $M^{>10} = M^{>10} \cup M$
 25. **if** $M == \emptyset$ **then**
 26. **break**;
 27. **else**
 28. **while** $M \neq \emptyset$ **do**
-

Algorithm 1 *Cont.*

```

29.  $\alpha_0 = 0, j = 0;$ 
30. for  $i \in M$  do
31.   if  $\alpha_i > \alpha_0$  then
32.      $\alpha_0 = \alpha_i, j = i;$ 
33.   end if
34. end for
35.  $y_j = 9.90 - \Delta \times 0.05, \Delta = \Delta + 1, M = M - \{j\};$ 
36. if  $y_j < y_0$  then
37.    $y_j = y_0;$ 
38. end if
39. end while
40. end if
41. end while
42. if  $|\{i | i \in N \wedge y_i < y'_0\}| \leq (n - \lceil n \times 2/3 \rceil)$  then
43.   return  $y_j (j = 1, 2, \dots, n);$ 
44. else
45.    $R = \{i | i \in N \wedge y_i < y'_0\}$ 
46.   while  $|R| > \lceil n \times 1/3 \rceil$  do
47.      $\alpha_0 = 0, j = 0$ 
48.     for  $i \in P$  do
49.       if  $\alpha_i > \alpha_0$  then
50.          $\alpha_0 = \alpha_i, j = i;$ 
51.       end if
52.     end for
53.      $R = R - \{j\}, y_j = y'_0;$ 
54.   end while
55.   return  $y_j (j = 1, 2, \dots, n);$ 
56. end if
57. end if

```

Theorem 1. *The time complexity of Algorithm 1 is $O(n^3)$.*

Proof. Steps 3–12 are a double nested loop, used to calculate the allocation results when $y_0 \geq T_0$. The number of loops in the first loop is $|Q|$, and that in the second loop is $|N|$. Since $|Q| \leq n$ and $|N| \leq n$, the time complexity of Steps 3–12 is $O(n^2)$.

Steps 14–41 are a while loop, applied to initially allocate software trustworthiness to trustworthy attributes when $y_0 < T_0$. The maximum number of loops in this while loop is $|N - M^{>10}|$, which satisfies $|N - M^{>10}| \leq n$. Steps 16–17 are applied to look for the approximate root k , satisfying $f(k) = 0$, through the Newton iteration method. It takes $O(m)$, where m expresses the maximum number of iterations the designer wants to perform. Steps 18–23 are a for loop, updating the values of attributes with subscripts in $N - M^{>10}$ through the attribute value growth function. The number of loops in this for loop is $|N - M^{>10}|$. Because of $|N - M^{>10}| \leq n$, then the time complexity of Steps 18–23 is $O(n)$. Steps 24–40 are an if-else conditional statement, updating the values of attributes with subscripts in $M^{>10}$ by the setting rules. The else statement part is a double nested loop, the maximum number of loops for both the loops are $|M|$. Due to $|M| \leq n$, then the time complexity of Steps 24–40 is $O(n^2)$. Consequently, we infer that the time complexity of Steps 14–41 is $O(nm + n^2 + n^3)$.

Steps 42–56 are an if-else statement, which are applied to finally assign software trustworthiness to trustworthy attributes when $y_0 < T_0$. The else statement part is a double nested loop, and the maximum number of loops for both the loops are $|R|$. Notice that $|R| \leq n$, and it follows the time complexity of Steps 42–56, is $O(n^2)$.

In summary, we can obtain the time complexity of Algorithm 1 is $O(nm + n^3)$. Because the expected maximum number of iterations of the Newton iteration method m is usually given before the algorithm is executed, the time complexity of Algorithm 1 is $O(n^3)$. \square

4. Improved Reallocation Approach for Software Trustworthiness

The basic idea of software trustworthiness reallocation is that the software trustworthy level needs to be upgraded due to actual needs, for example, upgrading level III software to level IV software, so it is necessary to reallocate the software trustworthiness to meet the requirements of the improved level. Similarly, in view of the fact that the development cost of trustworthy attributes is positively correlated with the value of trustworthy attributes, it is expected that under the condition of meeting the requirements of increasing the trustworthy level, the improved software trustworthiness reallocation minimizes the sum of the increases in all the trustworthy attribute values and takes into account the absolute majority of the classification model.

Definition 5. (Improved Software Trustworthiness Reallocation Model)

$$\begin{aligned} & \min \sum_{i=1}^n \Delta y_i \\ \text{such that } & \begin{cases} T = \prod_{i=1}^n y_i^{\alpha_i} \geq T_0 \\ 1 \leq y_0 \leq y_i \leq 10, 1 \leq i \leq n \\ 1 \leq y'_i \leq y_i, 1 \leq i \leq n \\ |\{i | y_i < y'_i\}| \leq (n - \lceil n \times 2/3 \rceil) \end{cases} \end{aligned}$$

where

1. T : trustworthy degree of software;
2. T_0 : minimum degree that the software needs to achieve after reallocation;
3. n : number of trustworthy attributes;
4. $y_i (1 \leq i \leq n)$: attribute values after reallocation;
5. $y'_i (1 \leq i \leq n)$: attribute values before reallocation;
6. y_0 : minimum value that all the trustworthy attributes must reach after reallocation, in the software trustworthiness classification model given in Table 1, $y_0 = 1$ for level I software, $y_0 = 1$ for level II software, $y_0 = 4.5$ for level III software, $y_0 = 7$ for level IV software, and $y_0 = 8.5$ for level V software;
7. y'_0 : minimum value of next-level trustworthy attributes after reallocation, in the software trustworthiness classification model given in Table 1, $y'_0 = 1$ for level I software, $y'_0 = 4.5$ for level II software, $y'_0 = 7$ for level III software, $y'_0 = 8.5$ for level IV software, and specify $y'_0 = 9.5$ for level V software;
8. $\Delta y_i = y_i - y'_i$: increased value of the i -th attribute after reallocation;
9. $\alpha_i (1 \leq i \leq n)$: weight values of trustworthy attributes, satisfying $0 \leq \alpha_i \leq 1, \sum_{i=1}^n \alpha_i = 1$.

From the results in Section 3, it can be seen that attribute sensitivity of T is equal to the attribute weight, and the larger the weight is, the more important it is to improve the software trustworthiness, so it is also necessary to increase the attribute value as much as possible. In order to simplify the reallocation process, and in view of the fact that the trustworthy attribute value after reallocation should be no less than that before reallocation, the trustworthy attribute value growth function is modified as follows.

Definition 6. (Modified Trustworthy Attribute Value Growth Function [39])

$$y_i = \max\{y_0, y'_i\} + k \times (\alpha_i - \alpha_{\min}), 1 \leq i \leq n$$

Among

1. n : number of attributes;

2. $y_i (1 \leq i \leq n)$: attribute values after reallocation;
3. $y'_i (1 \leq i \leq n)$: attribute values before reallocation;
4. y_0 : minimum value that all the trustworthy attributes must reach after reallocation;
5. $\alpha_i (1 \leq i \leq n)$: weight values of trustworthy attributes, satisfying $0 \leq \alpha_i \leq 1$, $\sum_{i=1}^n \alpha_i = 1$, $\alpha_{\min} = \min_{1 \leq i \leq n} \{\alpha_i\}$;
6. k : growth rate of the attribute value with $0 \leq k$.

Substituting the modified trustworthy attribute value growth function into the improved software trustworthiness reallocation model, the improved software trustworthiness reallocation model, on the basis of the modified trustworthy attribute value growth function, is obtained.

Definition 7. (Improved Software Trustworthiness Reallocation Model based on Modified Trustworthy Attribute Value Growth Function)

$$\text{minksatisfy} \begin{cases} \prod_{i=1}^n [\max\{y_0, y'_i\} + k \times (\alpha_i - \alpha_{\min})]^{\alpha_i} \geq T_0 \\ 1 \leq \max\{y_0, y'_i\} + k \times (\alpha_i - \alpha_{\min}) \leq 10 \\ |\{i | \max\{y_0, y'_i\} + k \times (\alpha_i - \alpha_{\min}) < y'_0\}| \leq (n - \lceil n \times 2/3 \rceil) \end{cases}$$

Among

1. k : growth rate of the attribute value, such that $0 \leq k$.
2. y_0 : minimum value that all the trustworthy attributes must reach after reallocation. The meaning is the same as in Definition 5;
3. $y'_i (1 \leq i \leq n)$: attribute values before reallocation;
4. y'_0 : minimum value of next-level trustworthy attributes after reallocation, see Definition 5 for specific meanings.
5. $\alpha_i (1 \leq i \leq n)$: weight values of trustworthy attributes, satisfying $0 \leq \alpha_i \leq 1$, $\sum_{i=1}^n \alpha_i = 1$, $\alpha_{\min} = \min_{1 \leq i \leq n} \{\alpha_i\}$;
6. T_0 : minimum degree that the software needs to achieve after reallocation.

Software trustworthiness allocation is to assign trustworthiness to software trustworthy attributes, according to their weight, and software trustworthiness reallocation is also to reassign trustworthiness to software trustworthy attributes by their weight values. Therefore, the software trustworthiness reallocation, on the basis of the modified trustworthiness attribute value growth function, is actually to look for the minimum k , within the interval

$$\left[0, \min_{i \in \{1, \dots, n\} - O} \left\{ \frac{10 - \max\{y_0, y'_i\}}{\alpha_i - \alpha_{\min}} \right\} \right], \text{ such that}$$

$$\begin{cases} 1 \leq \max\{y_0, y'_i\} + k \times (\alpha_i - \alpha_{\min}) \leq 10 \\ |\{i | \max\{y_0, y'_i\} + k \times (\alpha_i - \alpha_{\min}) < y'_0\}| \leq (n - \lceil n \times 2/3 \rceil) \end{cases}$$

where $0 \leq \alpha_i \leq 1$, $\sum_{i=1}^n \alpha_i = 1$, $\alpha_{\min} = \min_{1 \leq i \leq n} \{\alpha_i\}$, $O = \left\{ j \mid \alpha_j = \min_{1 \leq i \leq n} \{\alpha_i\} \right\}$.

Because the purpose of software trustworthiness reallocation is to improve the software trustworthiness, it is necessary to reduce the software modification as much as possible, in the case of improving the software trustworthiness to meet the level requirements. Therefore, if the value of the trustworthy attribute before reallocation is already greater than y'_0 , no modification is required. Let $N = \{1, 2, \dots, n\}$, $P = \{i | y_i \geq y'_0\}$.

- (1) When $|N - P| \leq (n - \lceil n \times 2/3 \rceil)$, let

$$g(k) = \prod_{i \in N - P - M > 10} [\max\{y_0, y'_i\} + k \times (\alpha_i - \alpha_{\min})]^{\alpha_i} \prod_{i \in P} y_i^{\alpha_i} \prod_{i \in M > 10} y_i^{\alpha_i},$$

where, $M^{>10}$ is a set with initial value \emptyset . At this time, the software trustworthiness is allocated by a method similar to that in Algorithm 1. It should be noted that the values of attributes with subscripts in $M^{>10}$ must meet $\max_{1 \leq i \leq n} \{y_0, y'_i\} \leq y_i$ in the updating process.

- (2) When $|N - P| > (n - \lceil n \times 2/3 \rceil)$. Since the greater the attribute weight is, the more important it is to improve the trustworthiness of the software, set the value of the trustworthy attribute with the subscript in P and the largest weight to y'_0 , and delete this subscript from the set P . Repeat the above process until $|N - P| \leq (n - \lceil n \times 2/3 \rceil)$ is satisfied. If the software trustworthiness meets the reallocation requirements at this time, return. Otherwise, let $P = \{i | y_i \geq y'_0\}$, and jump to (1) to execute.

The above specific solution process is shown in Algorithm 2.

Algorithm 2 Algorithm for reallocating software trustworthiness: for given trustworthy attribute values y'_1, \dots, y'_n before reallocation, trustworthy attribute weight values $\alpha_1, \alpha_2, \dots, \alpha_n$, T_0 that the software must achieve after reallocation and y_0 that all of the trustworthy attributes must reach after reallocation, output the reallocated trustworthy attribute values y_1, y_2, \dots, y_n .

Input: $y'_1, \dots, y'_n, T_0, y_0, \alpha_1, \alpha_2, \dots, \alpha_n$

Output: $y_j (j = 1, 2, \dots, n)$

1. Initialize $N = \{1, 2, \dots, n\}, P = \{i | i \in N \wedge y_i \geq y'_0\}, y_i = y'_i (i \in P), M^{>10} = \emptyset, \Delta = 0$;
 2. **if** $|N - P| \leq (n - \lceil n \times 2/3 \rceil)$ **then**
 3. $y_i = y'_i (i \in N - P)$
 4. **if** $\prod_{i \in N} y_i^{\alpha_i} \geq T_0$ **then**
 5. $Q = \{i | i \in (N - P) \wedge y_i < y_0\}$
 6. **for** $i \in Q$ **do**
 7. $y_i = y_0$
 8. **end for**
 9. **return** $y_j (j = 1, 2, \dots, n)$
 10. **else**
 11. **while** $N - P - M^{>10} \neq \emptyset$ **do**
 12. $g(k) = \prod_{i \in N - P - M^{>10}} [\max\{y_0, y'_i\} + k \times (\alpha_i - \alpha_{\min})]^{\alpha_i} \prod_{i \in P} y_i^{\alpha_i} \prod_{i \in M^{>10}} y_i^{\alpha_i} - T_0$;
 13. Taking $k = 0$ as the initial value of the iteration, $m = 1000$ as the maximum number of iterations, 10^{-10} as the error, the Newton iteration method is used to find the approximate root k that satisfies $g(k) = 0$;
 14. $M = \emptyset$;
 15. **for** $i \in N - P - M^{>10}$ **do**
 16. $y_i = \max\{y_0, y'_i\} + k \times (\alpha_i - \alpha_{\min})$;
 17. **if** $y_i > 10$ **then**
 18. $M = M \cup \{i\}$
 19. **end if**
 20. **end for**
 21. $M^{>10} = M^{>10} \cup M$
 22. **if** $M == \emptyset$ **then**
 23. **break**;
 24. **else**
 25. **while** $M \neq \emptyset$ **do**
 26. $\alpha_0 = 0, j = 0$;
 27. **for** $i \in M$ **do**
 28. **if** $\alpha_i > \alpha_0$ **then**
 29. $\alpha_0 = \alpha_i, j = i$;
 30. **end if**
 31. **end for**
 32. $y_j = 9.90 - \Delta \times 0.05, \Delta = \Delta + 1, M = M - \{j\}$;
-

Algorithm 2 *Cont.*

```

33.  if  $y_j < \max\{y_0, y'_j\}$  then
34.     $y_j = \max\{y_0, y'_j\}$ ;
35.  end if
36.  end while
37.  end if
38.  end while
39.  end if
40.  else
41.    while  $|N - P| > (n - \lceil n \times 2/3 \rceil)$  do
42.       $S = \left\{ i \mid \max_{i \in N-P} \{\alpha_i\} \right\}$ ;
43.      for  $i \in S$  do
44.         $y_i = y'_0$ ;
45.         $P = P \cup S$ ;
46.      end for
47.    end while
48.     $y_i = y'_i (i \in N - P)$ 
49.    if  $\prod_{i \in N} y_i^{\alpha_i} \geq T_0$  then
50.      return  $y_j (j = 1, 2, \dots, n)$ ;
51.    else
52.       $M^{>10} = \emptyset, \Delta = 0$ ;
53.      execute Steps 2–39;
54.    end if
55.  end if

```

Theorem 2. *The time complexity of Algorithm 2 is $O(n^3)$.*

Proof. Algorithm 2 is an if-else statement.

The if statement part is applied to calculate the reallocation result when $|N - P| \leq (n - \lceil n \times 2/3 \rceil)$. It is easy to obtain the time complexity of Steps 3–9 as $O(n)$. Steps 11–38 are a while loop, and its time complexity analysis is similar to that of Steps 14–41 in Algorithm 1. Because $|N - P - M^{>10}| \leq n$ and $|M| \leq n$, it is easy to get that the time complexity of Steps 11–38 is $O(nm + n^2 + n^3)$, where m is the maximum number of iterations that the designer wishes to execute.

The else statement part is used to compute the reallocation result when $|N - P| > (n - \lceil n \times 2/3 \rceil)$. Steps 41–47 are a double nested loop, the maximum number of loops of the first loop is $|N - P|$, and the number of loops of the second loop is $|S|$. Due to $|N - P| \leq n$ and $|S| \leq n$, it follows that the time complexity of Steps 41–47 is $O(n^2)$. Steps 49–54 are also an if-else statement. The if statement part is a return statement, so its time complexity is $O(1)$. Step 53 calls Steps 2–9, so the time complexity of Step 53 is $O(nm + n^2 + n^3)$. Thus, we arrive at the conclusion that the time complexity of the else statement part is $O(nm + n^2 + n^3)$.

Thus, the time complexity of Algorithm 2 is $O(nm + n^3)$. Because the expected maximum number of iterations of the Newton iteration method m is usually given before the algorithm is preformed, the time complexity of Algorithm 2 is $O(n^3)$. \square

5. Trustworthiness Enhancement Specification for Spacecraft Software Based on Factory Reports

In this section, we demonstrate the effectiveness of the improved allocation and reallocation approaches, by establishing the trustworthiness enhancement specification for

spacecraft software, based on factory reports, which includes a software trustworthiness development specification and software trustworthiness improvement specification.

5.1. Software Trustworthiness Measurement and Classification Based on Factory Reports

When the software is delivered, experts are usually organized to evaluate the software trustworthiness, based on the software factory reports. In order to enable the experts to give an objective evaluation, according to the review materials, a software trustworthiness hierarchical model, based on factory reports, is established in [3]. This hierarchical model consists of 9 attributes, which are subdivided into 28 sub-attributes. Each sub-attribute involves four measurement elements, and each of the four measurement elements only corresponds to one of the four levels of A, B, C, and D. Therefore, once the measurement element is selected, the corresponding level of the measurement element is determined, and then the sub-attribute measurement value, determined by the level of the measurement element, can be obtained. The trustworthy attributes, trustworthy sub-attributes in the software trustworthiness hierarchical model, based on factory reports, and their weight values are given in Table 2 [3,14].

Table 2. Trustworthy attributes, sub-attributes and their weight values in the software trustworthiness hierarchical model based on factory reports.

Attribute	Attribute Weight	Sub-Attribute	Sub-Attribute Weight
Overall planning and implementation	0.05	Development planning and execution	0.31
		Functionality and performance compliance with specification	0.36
		Integrity of software development documentation	0.33
Analysis and design	0.17	Input file controlled condition	0.33
		Requirement analysis	0.33
		Software design	0.34
Test verification	0.20	Test plan and implementation	0.16
		Code walk through and static analysis	0.17
		Test content comprehensiveness	0.17
		Special testing situation	0.17
		Test coverage	0.17
Reliability and safety	0.15	Test environment, methods and tool usage	0.16
		Reliability and safety analysis	0.33
		Reliability and safety design	0.34
Software technology status change	0.09	Reliability and safety verification	0.33
		Basis, demonstration and approval of technical status change	0.34
		Test and verification after changes	0.33
Quality problem close loop	0.09	Implementations after changes	0.33
		Quality problem zero completion	0.50
Configuration management	0.11	Implementation of one example against three tasks	0.50
		Configuration management organization, requirements, and tools	0.33
		Change control situation	0.34
		Configuration audit, documentary	0.33

Table 2. Cont.

Attribute	Attribute Weight	Sub-Attribute	Sub-Attribute Weight
Software development environment	0.05	Development supporting software	0.50
		Development supporting hardware	0.50
Third party evaluation situation	0.09	Evaluation input and evaluation plan situation	0.33
		Evaluation of implementation situation	0.33
		Problem solving situation	0.34

In order to compute, allocate and reallocate software trustworthiness, the trustworthy levels of measurement elements are first converted to specific values. That is, if the A-level measurement element is selected, the corresponding sub-attribute value is 10; if the B-level measurement element is chosen, the corresponding sub-attribute value is 9; if the C-level measurement element is selected, the corresponding sub-attribute value is 7; if the D-level measurement element is chosen, the corresponding sub-attribute value is 2. The software trustworthiness classification model, established in Table 1, is required to meet the absolute majority; that is, to reach a certain level of software trustworthiness, at least 2/3 of the trustworthy attribute values must reach this level. There are nine attributes in the software trustworthiness hierarchical model, based on factory reports. Then, the number of trustworthy attributes lower than the corresponding level is not more than three, and the software trustworthiness classification model, based on factory reports, can be built, as shown in Table 3.

Table 3. Software trustworthiness classification model based on factory reports.

Software Trustworthiness Measurement Requirements	Trustworthy Attribute Requirements	Trustworthy Level
$9.50 \leq T$	1. The number of trustworthy attributes with a value lower than 9.50 shall not exceed 3 2. No trustworthy attribute with a value of less than 8.50	V
$8.50 \leq T < 9.50$ or $9.50 \leq T$ and cannot be rated as level V	1. The number of trustworthy attributes with a value lower than 8.50 shall not exceed 3 2. No trustworthy attribute with a value of less than 7.00	IV
$7.00 \leq T < 8.50$ or $8.50 \leq T$ and cannot be rated as level IV or above	1. The number of trustworthy attributes with a value lower than 7.00 shall not exceed 3 2. No trustworthy attribute with a value less than 4.50	III
$4.50 \leq T < 7.00$ or $7.00 \leq T$ and cannot be rated as level III or above	1. The number of trustworthy attributes with a value less than 4.50 is not more than 3 2. No trustworthy attribute with a value less than 1	II
$T < 4.50$ or $4.50 \leq T$ and cannot be rated as level II or above	No trustworthy attribute with a value less than 1	I

A panel of 10 experts was invited to rate the 28 sub-attributes of the 23 spacecraft software, and the model presented in Definition 1 was used to measure these 23 spacecraft software. The classification model, shown in Table 3, was applied to classify these software trustworthiness measurement results. As such, 11 representative software are selected as study subjects, and the numbers of these 11 representative software are 2, 4, 6, 7, 9, 18, 19,

20, 21, 22, 23. The trustworthy attribute values, trustworthy degrees and trustworthy levels of 11 representative spacecraft software are given in Table 4 [3].

Table 4. Trustworthy attribute values, trustworthy degrees and trustworthy levels of 11 representative spacecraft software.

Attribute	No. 2	No. 4	No. 6	No. 7	No. 9	No. 18	No. 19	No. 20	No. 21	No. 22	No. 23
Overall planning and implementation	8.28	7.66	7.66	7.66	7.66	8.33	8.33	7.00	8.33	8.33	9.00
Analysis and design	7.61	7.61	7.87	7.87	7.87	7.61	7.61	7.61	7.87	7.87	7.61
Test verification	6.13	4.76	5.90	6.15	7.94	8.41	9.16	8.28	8.20	5.99	8.40
Reliability and safety	8.26	7.61	7.00	4.63	7.00	9.00	8.28	7.61	7.00	7.00	9.00
Software technology status change	8.28	8.26	8.26	8.26	8.26	8.28	8.26	7.61	9.00	9.00	8.58
Quality problem close loop	8.37	8.37	8.37	8.37	7.93	9.49	8.37	8.37	8.37	8.37	8.37
Configuration management	9.00	9.00	7.00	7.62	8.28	10.00	9.66	8.59	8.59	8.59	10.00
Software development environment	9.00	4.24	7.00	7.00	7.94	9.00	9.00	9.49	9.00	9.00	9.00
Third party evaluation situation	9.65	9.37	9.65	9.65	9.64	8.26	9.00	9.00	8.56	10.00	8.26
Trustworthy degree	7.90	7.09	7.36	7.04	7.97	8.61	8.58	8.08	8.17	7.76	8.57
Trustworthy level	III	II	III	III	III	IV	IV	III	III	III	IV

5.2. Trustworthiness Development Specification for Spacecraft Software Based on Factory Reports

In the following, we apply the improved software trustworthiness allocation algorithm, given in Section 3, to establish the software trustworthiness development specification of spacecraft software, based on factory reports. This specification guides the development of trustworthiness of this type of software. According to this specification, in order to develop software that achieves the corresponding trustworthy level, the developer must at least follow the level specified by the complete trustworthy sub-attribute during the development process. Considering that the trustworthy level of this type of software needs to be at least level III, only three specifications with trustworthy levels of level V, level IV and level III are given below.

It can be seen from the software trustworthiness classification model, based on factory reports given in Table 3, that to achieve level V, the degree of software trustworthiness is at least 9.50, the trustworthy attribute value is at least 8.50, and the number of attributes, whose values are greater than or equal to 8.50 and less than 9.50, cannot exceed 3. Before the trustworthiness allocation of the V-level software, the three attributes with the lowest weight value are pre-processed first. From the weight value of each attribute given in Table 2, it can be seen that there are two trustworthy attributes with the lowest weight value; namely, the overall planning and execution and the software development environment, both of which have weight values of 0.05, and their values are set as 8.50. There are three trustworthy attributes with the second lowest weight value, which are software technology status change, quality problem close loop and third-party evaluation situation, and the weight values are 0.09. If the values of these three are all set to 8.50, the condition that the number of attributes, whose values are greater than or equal to 8.50 and less than 9.50, cannot exceed 3 is not met. Because the trustworthiness allocation is based on the weight, the attributes with the same weight value need to be assigned the same value. Therefore, the attribute values of these three trustworthy attributes are all set to 9.50. For the remaining attributes, the trustworthy attribute values can be specified as 9.50. At this

time, it can be calculated that the software trustworthiness is 9.40, which does not meet the condition that the software trustworthiness must be greater than or equal to 9.5 in level V.

Denote the values of the nine trustworthy attributes as y_1, \dots, y_9 , respectively. According to Algorithm 1, the process of software allocation is as follows.

Due to $y_1 = 8.50, y_5 = 9.50, y_6 = 9.50, y_8 = 8.50, y_9 = 9.50$ are determined in the pre-processing stage. Let $N = \{2, 3, 4, 7\}, M^{>10} = \emptyset, Q = \emptyset, \Delta = 0$. For level V software, $y_0 = 8.50, T_0 = 9.50$. Since $y_0 < T_0$, let

$$\begin{aligned} f(k) &= \prod_{i \in \{2,3,4,7\} - M^{>10}} [y_0 + k(\alpha_i - \alpha_{\min})]^{\alpha_i} \prod y_1^{\alpha_1} y_5^{\alpha_5} y_6^{\alpha_6} y_8^{\alpha_8} y_9^{\alpha_9} \prod_{i \in M^{>10}} y_i^{\alpha_i} - T_0 \\ &= \prod_{i \in \{2,3,4,7\}} [y_0 + k(\alpha_i - \alpha_{\min})]^{\alpha_i} \prod 8.50^{0.05} 9.50^{0.09} 9.50^{0.09} 8.50^{0.05} 9.50^{0.09} - 9.50 \end{aligned}$$

According to the Newton iterative method, the approximate solution $k = 1.4816$ of $f(k) = 0$ is obtained with $k = 0$ as the initial iteration value, $m = 1000$ as the maximum number of iterations, and 10^{-10} as the error. Substituting $k = 1.4816$ into the attribute value growth function $y_i = y_0 + k(\alpha_i - \alpha_{\min}), i \in \{2, 3, 4, 7\}$, we can obtain

$$y_2 = 9.68, y_3 = 9.72, y_4 = 9.65, y_7 = 9.59$$

Because all of these are less than 10.00, Algorithm 1 is stopped.

Since spacecraft software belongs to the safety-critical field, the trustworthy sub-attributes, corresponding to the V-level software, are limited to at least level C. By using the software attribute trustworthiness allocation algorithm, given in reference [43], we can assign software attribute trustworthiness to sub-attribute. The trustworthiness development specification of V-level spacecraft software, based on factory reports, is shown in Table 5. The highest trustworthy value of software attributes is 9.72, and the corresponding attribute is test verification. In addition, the attributes with a trustworthy value greater than 9.50 include analysis and design (9.68), reliability and safety (9.65), and configuration management (9.59). The attributes with a trustworthy value equal to 9.50 contain software technical status change, quality problem zeroing, and third-party evaluation. The attributes with a trustworthy value equal to 8.50 are overall planning and execution, and software development environment. The degree of software trustworthiness is equal to 9.50. As can be seen from the specification, the more important the attribute is, the greater its trustworthy value is. The sub-attribute level in the specification only specifies the minimum requirements of the sub-attribute, and in the actual development process, it can be carried out according to the high requirements.

Similarly, from Table 3, it can be seen that in order to reach level IV, the degree of software trustworthiness should be at least 8.50, the trustworthy attribute value should be at least 7.00, and the number of attributes, whose value is greater than or equal to 7.00 and less than 8.50, should not exceed 3. The values of attributes with the lowest weight value are all set as 7.00. The values of attributes with the second lowest weight value are all set to 8.50. For the same reason, the trustworthy sub-attribute corresponding to the IV-level software are restricted to at least level C.

According to Algorithm 1 and the software attribute trustworthiness allocation algorithm, presented in reference [43], the trustworthiness specification of IV-level spacecraft software, based on factory reports, is shown in Table 6. From this specification, we can see that most of the sub-attribute levels of IV-level trustworthy software are level B, which is quite different from that of V-level trustworthy software.

In the same way, we can obtain the level III specification of this type, based on factory reports, as shown in Table 7. Most of the sub-attribute levels of III-level spacecraft software are level C and there is no level A.

Table 5. Trustworthiness development specification of V-level spacecraft software based on the factory reports.

Attribute	Attribute Value	Sub-Attribute	Sub-Attribute Level
Overall planning and implementation	8.50	Development planning and execution	C
		Functionality and performance compliance with specification	A
		Integrity of software development documentation	B
Analysis and design	9.68	Input file controlled condition	A
		Requirement analysis	A
		Software design	A
Test verification	9.72	Test plan and implementation	A
		Code walk through and static analysis	A
		Test content comprehensiveness	A
		Special testing situation	A
		Test coverage	A
		Test environment, methods and tool usage	B
Reliability and safety	9.65	Reliability and safety analysis	B
		Reliability and safety design	A
		Reliability and safety verification	A
Software technology status change	9.50	Basis, demonstration and approval of technical status change	A
		Test and verification after changes	A
		Implementations after changes	B
Quality problem close loop	9.50	Quality problem zero completion	A
		Implementation of one example against three tasks	A
Configuration management	9.59	Configuration management organization, requirements, and tools	A
		Change control situation	A
		Configuration audit, documentary	B
Software development environment	8.50	Development supporting software	B
		Development supporting hardware	B
Third party evaluation situation	9.50	Evaluation input and evaluation plan situation	A
		Evaluation of implementation situation	B
		Problem solving situation	A

Table 6. Trustworthiness development specification of IV-level spacecraft software based on factory reports.

Attribute	Attribute Value	Sub-Attribute	Sub-Attribute Level
Overall planning and implementation	7.00	Development planning and execution	C
		Functionality and performance compliance with specification	C
		Integrity of software development documentation	C
Analysis and design	8.78	Input file controlled condition	B
		Requirement analysis	B
		Software design	B
Test verification	8.85	Test plan and implementation	B
		Code walk through and static analysis	B
		Test content comprehensiveness	B
		Special testing situation	B
		Test coverage	B
		Test environment, methods and tool usage	B
Reliability and safety	8.73	Reliability and safety analysis	B
		Reliability and safety design	B
		Reliability and safety verification	B

Table 6. *Cont.*

Attribute	Attribute Value	Sub-Attribute	Sub-Attribute Level
Software technology status change	8.50	Basis, demonstration and approval of technical status change	A
		Test and verification after changes	B
		Implementations after changes	C
Quality problem close loop	8.50	Quality problem zero completion	B
		Implementation of one example against three tasks	B
Configuration management	8.64	Configuration management organization, requirements, and tools	B
		Change control situation	B
		Configuration audit, documentary	B
Software development environment	7.00	Development supporting software	C
		Development supporting hardware	C
Third party evaluation situation	8.50	Evaluation input and evaluation plan situation	C
		Evaluation of implementation situation	B
		Problem solving situation	A

Table 7. Trustworthiness development specification of III-level spacecraft software based on the factory reports.

Attribute	Attribute Value	Sub-Attribute	Sub-Attribute Level
Overall planning and implementation	4.50	Development planning and execution	C
		Functionality and performance compliance with specification	C
		Integrity of software development documentation	C
Analysis and design	7.54	Input file controlled condition	C
		Requirement analysis	C
		Software design	B
Test verification	7.67	Test plan and implementation	C
		Code walk through and static analysis	B
		Test content comprehensiveness	B
		Special testing situation	C
		Test coverage	C
Reliability and safety	7.45	Test environment, methods and tool usage	C
		Reliability and safety analysis	C
		Reliability and safety design	B
Software technology status change	7.00	Reliability and safety verification	C
		Basis, demonstration and approval of technical status change	C
		Test and verification after changes	C
Quality problem close loop	7.00	Implementations after changes	C
		Quality problem zero completion	C
Configuration management	7.28	Implementation of one example against three tasks	C
		Configuration management organization, requirements, and tools	C
		Change control situation	B
Software development environment	4.50	Configuration audit, documentary	C
		Development supporting software	C
Third party evaluation situation	7.00	Development supporting hardware	C
		Evaluation input and evaluation plan situation	C
		Evaluation of implementation situation	C
		Problem solving situation	C

From these specifications, it can be seen that there is only one C-level sub-attribute in V-level spacecraft software, while there is no A-level sub-attribute in III-level spacecraft software. There are no D-level sub-attributes in level V, IV and III software. In fact, level D specifies that the trustworthy value of this sub-attribute is 2, which means that this sub-attribute is not trusted.

5.3. Trustworthiness Improvement Specification for Spacecraft Software Based on Factory Reports

Let y'_1, \dots, y'_9 represent the values of the nine trustworthy attributes in Table 4, before software trustworthiness reallocation, and y_1, \dots, y_9 express that after reallocation. Suppose the degree of the software trustworthiness, before improvement, is T' , and the degree of improved software trustworthiness is T .

It is assumed that the level II software in Table 4 needs to be improved to level III software; that is, the software numbered 4 is upgraded from level II to level III. Then, based on the improved software trustworthiness reallocation method, a specification can be given to guide the software trustworthiness improvement. The process of trustworthiness reallocation of the software numbered 4 is as follows. First, initialize $N = \{1, 2, \dots, 9\}$, then $n = 9$. Since $y_0 = 4.50$, $y'_0 = 7.00$, $T_0 = 7.00$ for level III. We can obtain $P = \{i | i \in N \wedge y_i \geq y'_0\} = \{1, 2, 4, 5, 6, 7, 9\}$, and let $\{y_i = y'_i | i \in P\}$. Because $|N - P| = 2 < (n - \lceil n \times 2/3 \rceil) = 3$, let $y_i = y'_i (i \in N - P)$. Since $\prod_{i \in N} y_i^{a_i} = 7.09 \geq 7.00$, it can be derived that $Q = \{i | i \in (N - P) \wedge y_i < y_0\} = \{8\}$, set $y_8 = 4.50$, then Algorithm 2 is stopped.

The results calculated using Algorithm 2 are shown in Table 8. It can be seen from Table 8 that in order to improve the software numbered 4, from level II to level III, it is only necessary to upgrade the software development environment attribute value from 4.24 to 4.50. Table 7 shows that only two sub-attributes of the software development environment, development supporting software and development supporting hardware, need to be raised to level C respectively.

Table 8. Trustworthiness improvement specification of software numbered 4 from level II to level III.

Changes of Trustworthy at Tribute Values	No. 4	Changes of Trustworthy Attribute and Software Trustworthiness Values	No. 4
$y'_1 \rightarrow y_1$	7.66 \rightarrow 7.66	$y'_6 \rightarrow y_6$	8.37 \rightarrow 8.37
$y'_2 \rightarrow y_2$	7.61 \rightarrow 7.61	$y'_7 \rightarrow y_7$	9.00 \rightarrow 9.00
$y'_3 \rightarrow y_3$	4.76 \rightarrow 4.76	$y'_8 \rightarrow y_8$	4.24 \rightarrow 4.50
$y'_4 \rightarrow y_4$	7.61 \rightarrow 7.61	$y'_9 \rightarrow y_9$	9.37 \rightarrow 9.37
$y'_5 \rightarrow y_5$	8.26 \rightarrow 8.26	$T' \rightarrow T$	7.09 \rightarrow 7.12

It is assumed that the software numbered 2 in Table 4 needs to be upgraded from level III to level IV. The following is the process of trustworthiness reallocation of software numbered 2. First, initialize $N = \{1, 2, \dots, 9\}$, then $n = 9$. Since $y_0 = 7.00$, $y'_0 = 8.50$, $T_0 = 8.50$ for level IV, it follows that $P = \{i | i \in N \wedge y_i \geq y'_0\} = \{7, 8, 9\}$, and let $\{y_i = y'_i | i \in P\}$. Because $|N - P| = 6 > (n - \lceil n \times 2/3 \rceil) = 3$, take the three attributes with the largest weight values from set $\{y'_i | i \in (N - P)\}$ and set their attribute values to 8.5 after software trustworthiness reallocation. That is, $y_2 = 8.5$, $y_3 = 8.5$, $y_4 = 8.5$. Update $P = \{2, 3, 4, 7, 8, 9\}$ and let $\{y_i = y'_i | i \in (N - P)\}$. Due to $\prod_{i \in N} y_i^{a_i} = 8.63 > T_0 = 8.50$, Algorithm 2 is stopped. The reallocation results are $y_1 = 8.28$, $y_2 = 8.50$, $y_3 = 8.50$, $y_4 = 8.50$, $y_5 = 8.50$, $y_6 = 8.37$, $y_7 = 9.00$, $y_8 = 9.00$, $y_9 = 9.65$, as shown in Table 9.

At this time, it is necessary to adjust the attribute values of test verification, analysis and design, and reliability and safety, to 8.50. According to the calculation of the software attribute trustworthiness allocation algorithm designed in [30], the level of measurement element, corresponding to each sub-attribute of analysis and design, reliability and security, should be adjusted to B. Test validation can be adjusted in a variety of ways, such as

adjusting test planning and execution to level C and the rest to level B, or adjusting test environment, method and tool usage to level C and the rest to level B.

Table 9. Trustworthiness improvement specification of software numbered 2 from level III to level IV.

Changes of Trustworthy at Tribute Values	No. 2	Changes of Trustworthy Attribute and Software Trustworthiness Values	No. 2
$y'_1 \rightarrow y_1$	8.28 \rightarrow 8.28	$y'_6 \rightarrow y_6$	8.37 \rightarrow 8.37
$y'_2 \rightarrow y_2$	7.61 \rightarrow 8.50	$y'_7 \rightarrow y_7$	9.00 \rightarrow 9.00
$y'_3 \rightarrow y_3$	6.13 \rightarrow 8.50	$y'_8 \rightarrow y_8$	9.00 \rightarrow 9.00
$y'_4 \rightarrow y_4$	8.26 \rightarrow 8.50	$y'_9 \rightarrow y_9$	9.65 \rightarrow 9.65
$y'_5 \rightarrow y_5$	8.28 \rightarrow 8.28	$T' \rightarrow T$	7.90 \rightarrow 8.63

6. Discussion

Reference [38] defines a software trustworthiness allocation model as a mathematical programming model, and reference [43] presents a similar allocation model for software attribute trustworthiness. The improved software trustworthiness allocation model given in this paper has the same objective function as the model proposed in [38], but adds constraints related to the absolute majority of software trustworthiness classification. On the other hand, their attribute variables have different value ranges. The value range of the attribute variable of the model established in [38] is (0,1). In this case, when the attribute value y_i is kept unchanged and the attribute weight value α_i is increased, $y_i^{\alpha_i}$ will decrease, which is inconsistent with expectations. The value range of attribute variables in the improved model is (1,10), which can solve this problem well. At the same time, the improved model has different objective functions and constraints from the software attribute trustworthiness allocation model described in [43]. Therefore, the allocation algorithms proposed in [38,43] cannot be used to solve the improved allocation model. In order to solve the improved model, a polynomial time algorithm is designed in this paper. In the improved allocation algorithm, the trustworthiness initial allocation is carried out by using the Newton iteration algorithm, and the constraint of absolute majority is taken into account in the final allocation.

Reference [39] also builds a software trustworthiness reallocation model, based on mathematical programming, and the improved software trustworthiness reallocation model presented here has the same objective function, but more constraints. Meanwhile, reference [39] does not comprehensively consider reducing software modifications as much as possible when designing the reallocation algorithm. In order to solve this problem, in the improved reallocation algorithm, if the value of the trustworthy attribute, before reallocation, is already greater than y'_0 , no modification is required. Similar to the improved allocation algorithm, the Newton iteration algorithm is used for the initial reallocation, and then the final reallocation is carried out according to the absolute majority constraint.

The effectiveness of allocation and reallocation methods, given in [38,39,43], are validated by examples. In this paper, the improved allocation and reallocation methods are applied to formulate a trustworthiness enhancement specification for spacecraft software, based on factory reports, to demonstrate their effectiveness. The improved allocation algorithm is used to establish the software trustworthiness development specification of spacecraft software, and the improved reallocation algorithm is utilized to build a software trustworthiness improvement specification.

7. Conclusions and Future Work

In this paper, the software trustworthiness measurement model used in [3,40], the software trustworthiness classification model established in [3] and the software trustworthiness hierarchical model, based on factory reports, given in [3], are introduced. We present a software trustworthiness classification model, based on factory reports, and construct improved models to allocate and reallocate the trustworthy degree of software to its

attribute appropriately. Both the improved allocation and reallocation model are defined as mathematical programming models. Compared with the allocation model defined in [38], the improved allocation model adds the absolute majority of software trustworthiness classification constraint, and has more reasonable attribute variable value range. The improved reallocation model also adds the absolute majority constraint, compared to the model given in [39]. Polynomial time algorithms are given to calculate the optimal solutions of these improved models. They can more accurately assign or reassign software trustworthiness. Moreover, the designed reallocation algorithm can also reduce the software modifications as much as possible, in the process of improving the software. It should be noted that the allocation and reallocation algorithms given in this paper are only suitable for the case where the growth rates of all attributes are the same. The trustworthiness enhancement specification of spacecraft software, based on our improved allocation and reallocation approaches, is formulated, including trustworthiness development specification and trustworthiness improvement specification. It demonstrates the rationality and effectiveness of these improved methods. Meanwhile, these improved methods can be used to develop trustworthiness enhancement specifications for other types of software.

The following questions deserve further study. Firstly, both the improved allocation model and the improved reallocation model assume that all the attributes have the same growth rate. If their growth rates are not the same, how to design polynomial time algorithms for related models is very important to study. Secondly, both the trustworthy attribute value growth function and the modified trustworthy attribute value growth function are linear functions. If they are other types of functions, how can the corresponding allocation and reallocation model be solved? Thirdly, the improved software trustworthiness allocation model and reallocation model are both constructed based on the simplified software trustworthiness measurement model. In the future, we will study how to build the software trustworthiness allocation model and reallocation model, according to other software trustworthiness measurement models, and design corresponding solving algorithms. Finally, we assume that the software attribute values are positively correlated with the software development cost, but do not give the quantitative relationship model between them. In the future, we will study the quantitative relationship model between the software attribute values and the software development cost, and introduce the quantitative relationship model into the software trustworthiness allocation and reallocation models, to guide the control of the software development cost more accurately.

Author Contributions: Conceptualization, H.T. and Y.C.; methodology, H.T. and Y.C.; formal analysis, H.T., Y.C. and L.H.; software, L.F.; writing—original draft preparation, H.T. and L.F.; writing—review and editing, L.H. and X.W.; funding acquisition, H.T. All authors have read and agreed to the published version of the manuscript.

Funding: This work was financially supported by Doctoral Research Fund of Zhengzhou University of Light Industry (2016BSJJ037) and Science and Technology Project of Henan Province (212102210076, 202102210351).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: We would like to thank all the scholars who give many constructive suggestions and critical comments, which help us improve the general quality of the paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. He, J.F.; Shan, Z.G.; Wang, J.; Pu, G.G.; Fang, Y.F.; Liu, K.; Zhao, R.Z.; Zhang, Z.T. Review of the Achievements of Major Research Plan of Trustworthy Software. *Bull. Natl. Nat. Sci. Found. China* **2018**, *32*, 291–296.
2. Jin, Z.H. *Fatal Bug: The Disaster and Enlightenment of Software Defects*; People's Posts and Telecommunications Press: Beijing, China, 2016.

3. Chen, Y.X.; Tao, H.W. *Software Trustworthiness Measurement Evaluation and Enhancement Specification*; Science Press: Beijing, China, 2019.
4. Tao, H.W.; Chen, Y.X.; Wu, H.Y.; Deng, R.M. A Survey of Software Trustworthiness Measurements. *Int. J. Perform. Eng.* **2019**, *15*, 2364–2372.
5. Deng, R.M.; Chen, Y.X.; Wu, H.Y.; Tao, H.W. Software Trustworthiness Evaluation using Structural Equation Modeling. *Int. J. Perform. Eng.* **2019**, *15*, 2628–2635.
6. Alarcon, G.M.; Militello, L.G.; Ryan, P.; Jessup, S.A.; Calhoun, C.S.; Lyons, J.B. A Descriptive Model of Computer Code Trustworthiness. *J. Cogn. Eng. Dec. Mak.* **2017**, *11*, 107–121. [\[CrossRef\]](#)
7. Basso, T.; Silva, H.; Moraes, R. On the Use of Quality Models to Characterize Trustworthiness Properties. In Proceedings of the International Workshop on Software Engineering for Resilient Systems, Naples, Italy, 17 September 2019; pp. 147–155.
8. Alarcon, G.M.; Ryan, T.J. Trustworthiness Perceptions of Computer Code: A Heuristic-Systematic Processing Model. In Proceedings of the 51st Hawaii International Conference on System Sciences, Waikoloa Village, HI, USA, 2–6 January 2018; pp. 5384–5393.
9. Lv, Z.H.; Han, Y.; Singh, K.A.; Manogaran, G.; Lv, H.B. Trustworthiness in Industrial IoT Systems Based on Artificial Intelligence. *IEEE Trans. Industr. Inform.* **2021**, *17*, 1496–1504.
10. Lemes, C.I.; Naessens, V.; Vieira, M. Trustworthiness Assessment of Web Applications: Approach and Experimental Study Using Input Validation Coding Practices. In Proceedings of the 30th International Symposium on Software Reliability Engineering (ISSRE), Berlin, Germany, 28 October–1 November 2019; pp. 435–445.
11. Medeiros, N.; Ivaki, N.; Costa, P.; Vieira, M. Vulnerable Code Detection Using Software Metrics and Machine Learning. *IEEE Access* **2020**, *8*, 219174–219198. [\[CrossRef\]](#)
12. Medeiros, N.; Ivaki, N.; Costa, P.; Vieira, M. An Approach for Trustworthiness Benchmarking Using Software Metrics. In Proceedings of the 23rd IEEE Pacific Rim International Symposium on Dependable Computing (PRDC 2018), Taipei, Taiwan, 4–7 December; pp. 84–93.
13. Tao, H.W.; Zhao, J. An Improved Attributes-Based Software Trustworthiness Metric Model. *J. Wuhan Univ.* **2017**, *63*, 151–157.
14. Tao, H.W.; Chen, Y.X.; Wu, H.Y. Decomposition of Attributes Oriented Software Trustworthiness Measure Based on Axiomatic Approaches. In Proceedings of the 20th International Conference on Software Quality, Reliability and Security Companion (QRS-C), Macau, China, 11–14 December 2020; pp. 308–315.
15. Liu, H.; Tao, H.W.; Chen, Y.X. An Approach for Trustworthy Evidence of Source Code Oriented Aerospace Software Trustworthiness Measurement. *AER Control Appl.* **2021**, *47*, 32–41.
16. Khan, S.; Jha, S.K.; Khatri, S.K. Dependability and Trustworthiness Analysis for Component Based Software Development. *Int. J. Rec. Techn. Eng.* **2019**, *8*, 2277–3878.
17. Jabeen, G.; Ping, L. A Unified Measurable Software Trustworthy Model Based on Vulnerability Loss Speed Index. In Proceedings of the 18th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/13th IEEE International Conference on Big Data Science and Engineering, Rotorua, New Zealand, 5–8 August 2019; pp. 18–25.
18. Muhammad, D.M.S.; Fairul, R.F.; Loo, F.A.; Nur, F.A.; Norzamarini, B. Rating of Software Trustworthiness Via Scoring of System Testing Results. *Int. J. Dig. Enterp. Technol.* **2018**, *1*, 121–134.
19. Wang, D.X.; Wang, Q.; He, J. Evidence-based Software Process Trustworthiness Model and Evaluation Method. *J. Softw.* **2017**, *28*, 1713–1731.
20. Ogunniye, G.; Legastelois, B.; Rovatsos, M.; Dowthwaite, L.; Portillo, V.; Vallejos, E.P.; Zhao, J.; Jirotko, M. Understanding User Perceptions of Trustworthiness in E-recruitment Systems. *IEEE Internet Comput.* **2021**, *25*, 23–32. [\[CrossRef\]](#)
21. Shi, H.L.; Ma, J.; Zou, F.Y. A Fuzzy Comprehensive Evaluation Model for Software Dependability based on Entropy Weight. In Proceedings of the 2008 International Conference on Computer Science and Software Engineering, Wuhan, China, 12–14 December 2008; pp. 683–685.
22. Li, B.; Cao, Y. An Improved Comprehensive Evaluation Model of Software Dependability based on Rough Set Theory. *J. Soft.* **2009**, *4*, 1152–1159. [\[CrossRef\]](#)
23. Zhang, F.; Xu, M.D.; Chao, H.C.; Zhang, C.; Liu, X.L.; Hu, F.N. Real-time Trust Measurement of Software: Behavior Trust Analysis Approach based on Noninterference. *J. Softw.* **2019**, *30*, 2268–2286.
24. Tian, F.; Guo, Y.H. Software Trustworthiness Evaluation Model based on Behavior Trajectory Matrix. *Inform. Softw. Technol.* **2020**, *119*, 106–233.
25. Ji, C.Y.; Su, X.; Qin, Z.F.; Nawaz, A. Probability Analysis of Construction Risk based on Noisy-or Gate Bayesian Networks. *Rel. Eng. Syst. Saf.* **2022**, *217*, 107974. [\[CrossRef\]](#)
26. Ogundoyin, S.O.; Kamil, I.A. A Fuzzy-AHP based Prioritization of Trust Criteria in Fog Computing Services. *Appl. Soft Comput.* **2020**, *97*, 106789. [\[CrossRef\]](#)
27. Sahu, K.; Srivastava, R.K. Soft Computing Approach for Prediction of Software Reliability. *ICIC Express Lett.* **2018**, *12*, 1213–1222.
28. Sahu, K.; Srivastava, R.K. Revisiting Software Reliability. In *Book Data Management, Analytics and Innovation, Advances in Intelligent Systems and Computing*; Balas, V., Sharma, N., Chakrabarti, A., Eds.; Springer: Singapore, 2019; Volume 808, pp. 221–235.
29. Al-Mejibli, I.S.; Alharbe, N.R. A Fuzzy Analytic Hierarchy Process for Security Risk Assessment of Web Based Hospital Management System. *Int. J. Adv. Trends Comput. Sci. Eng.* **2019**, *8*, 2470–2474. [\[CrossRef\]](#)

30. Marshall, A.M. Digital Forensic Tool Verification: An Evaluation of Options for Establishing Trustworthiness. *Forensic Sci. Int.: Digit. Investig.* **2021**, *38*, 301181. [[CrossRef](#)]
31. Maza, S.; Megouas, O. Framework for Trustworthiness in Software Development. *Int. J. Perf. Eng.* **2021**, *17*, 241–252. [[CrossRef](#)]
32. Buraga, S.C.; Amariei, D.; Dospinescu, O. An OWL-Based Specification of Database Management Systems. *Comput. Mater. Cont.* **2022**, *70*, 5537–5550. [[CrossRef](#)]
33. Riehle, R.D.; Michael, J.B. Improving the Trustworthiness of Software Through Rigorous Data Type Design. *Computer* **2021**, *54*, 89–95. [[CrossRef](#)]
34. Xu, Z.S. Research on Software Trustworthiness Measurement Evaluation Model based on Data Driven. In Proceedings of the 2nd International Conference on Computer Science Communication and Network Security (CSCNS2020), Sanya, China, 22–23 December 2020; pp. 1–4.
35. Novikova, E.; Doynikova, E.; Gaifulina, D.; Kotenko, I. Construction and Analysis of Integral User-Oriented Trustworthiness Metrics. *Electronics* **2022**, *11*, 234. [[CrossRef](#)]
36. Alzahari, S.; Kamalrudin, M. An Approach to Elicit Trustworthiness Requirements in Blockchain Technology. *J. Phys. Conf. Ser.* **2021**, *1807*, 012031. [[CrossRef](#)]
37. Choudhary, C.; Kapur, P.K.; Khatri, S.K.; Majumdar, R. Software Quality and Reliability Improvement in Open Environment. In *Book Advances in Interdisciplinary Research in Engineering and Business Management: Asset Analytics (Performance and Safety Management)*; Kapur, P.K., Singh, G., Panwar, S., Eds.; Springer: Singapore, 2021; pp. 263–276.
38. Ma, Y.J.; Chen, Y.X.; Gu, B. An Attributes-Based Allocation Approach of Software Trustworthy Degrees. In Proceedings of the 2015 IEEE International Conference on Software Quality, Reliability and Security Companion, Vancouver, BC, Canada, 3–5 August 2015; pp. 89–94.
39. Tao, H.W.; Chen, Y.X.; Wu, H.Y. A Reallocation Approach for Software Trustworthiness Based on Trustworthy Attributes. *Mathematics* **2020**, *8*, 14. [[CrossRef](#)]
40. Wang, J.; Chen, Y.X.; Gu, B.; Guo, X.Y.; Wang, B.H.; Jin, S.Y.; Xu, J.; Zhang, J.Y. An Approach to Measuring and Grading Software Trust for Spacecraft Software. *Sci. Sin. Technol.* **2015**, *45*, 221–228.
41. Kitchenham, B.; Pfleeger, S.L.; Fenton, N. Towards a Framework for Software Measurement Validation. *IEEE Trans. Softw. Eng.* **1995**, *21*, 929–943. [[CrossRef](#)]
42. Briand, L.C.; Morasca, S.; Basili, R.V. Property-based Software Engineering Measurement. *IEEE Trans. Softw. Eng.* **1996**, *22*, 68–86. [[CrossRef](#)]
43. Tao, H.W.; Wu, H.Y.; Chen, Y.X. An Approach of Trustworthy Measurement Allocation Based on Sub-Attributes of Software. *Mathematics* **2019**, *7*, 237. [[CrossRef](#)]