Yang X, Jabeen G, Luo P *et al.* A unified measurement solution of software trustworthiness based on social-to-software framework. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 33(3): 603–620 May 2018. DOI 10.1007/s11390-018-1843-2

A Unified Measurement Solution of Software Trustworthiness Based on Social-to-Software Framework

Xi Yang[1], *Member*, *CCF*, *ACM*, Gul Jabeen[1], Ping Luo[1], Xiao-Ling Zhu[2], and Mei-Hua Liu[3]

[1] *Key Laboratory for Information System Security, School of Software, Tsinghua University, Beijing 100084, China*

[2] *School of Computer and Information, Hefei University of Technology, Hefei 230009, China*

[3] *Department of Foreign Languages and Literatures, Tsinghua University, Beijing 100084, China*

E-mail: {x-yang14, jgl14}@mails.tsinghua.edu.cn; luop@mail.tsinghua.edu.cn; zhuxl@hfut.edu.cn
liumeihua@mail.tsinghua.edu.cn

Received July 28, 2016; revised June 8, 2017.

Abstract As trust becomes increasingly important in software domain, software trustworthiness — as a complex high-composite concept, has developed into a big challenge people have to face, especially in the current open, dynamic and ever-changing Internet environment. Furthermore, how to recognize and define trust problem from its nature and how to measure software trustworthiness correctly and effectively play a key role in improving users' trust in choosing software. Based on trust theory in the field of humanities and sociology, this paper proposes a measurable S2S (Social-to-Software) software trustworthiness framework, introduces a generalized indicator loss to unify three parts of trustworthiness result, and presents a whole metric solution for software trustworthiness, including the advanced J-M model based on power function and time-loss rate for ability trustworthiness measurement, the fuzzy comprehensive evaluation advanced-model considering effect of multiple short boards for basic standard trustworthiness, and the identity trustworthiness measurement method based on the code homology detecting tools. Finally, it provides a case study to verify that the solution is applicable and effective.

Keywords software trustworthiness, measurement solution, loss, social-to-software (S2S) framework, generalized indicator

1 Introduction

Trust in software is an important concept because people today rely on software more than ever before[1]. Software trustworthiness, though a new subject in research, is a challenge people have to face, especially in the current open, dynamic and ever-changing Internet environment[2-5]. The software's incompatibility between flexibility and other properties like dependability, security and so on, has never been so urgent as the trust crisis in software applications is increasingly deepening and broadening. Traditional software quality and reliability metrics are no longer sufficient in the current cyber environment.

As a complex high-composite concept, whether it can be measured correctly and effectively plays a key role in improving users' trust in choosing software. Though increasingly more research has been done on it, few breakthroughs have been made[2]. This lies in the following. 1) Different measure solutions have been developed, while there is a lack of widely endorsed definition of trustworthiness. In contrast, measurement has advanced more rapidly than conceptual clarification, which just feels like putting the cart before the horse. 2) It is hard to compare different metrics due to the lack of a unified indicator. 3) The existing measurements of trustworthiness focus on different aspects of the issue such as software process[3-4], software

Regular Paper

This work was supported by the National Natural Science Foundation of China under Grant No. 90818021, the HeGaoJi Program of China under Grant No. 2012zx01039-004-46, and the Information Security Program of National Development and Reform Commission of China under Grant No. 2012-1424.

products (after release)[5-6], or software components[7]. Then what is the relationship between these measurements and how to unify or combine their indicators together in order to facilitate to compare each other? 4) Current research often simply divides trustworthiness into different quality attributes[2,5-6,8], as done in traditional software quality research[9] or software engineering research, which often does not address the trust application problems of software under the complex environment nowadays. And software trustworthiness is not the simple addition or combination of the attributes. Moreover, the trust of one attribute does not guarantee the trust of the whole software for which academia has not defined clearly, let alone other important researches like overall measurement, multi-dimensional and multi-metric quantified indicators system, dynamic acquiring and indicator variation of trustworthiness monitoring[10-11]. Consequently, it is extremely difficult to define software trustworthiness, which is exactly the reason why no agreement has been reached on how to define it and how to measure it, and contradictory findings are often observed in studies on software trustworthiness. Traditional software quality and reliability metrics are no longer sufficient in the current cyber environment. On the other hand, software trustworthiness evaluation plays a key role in improving users' trust in choosing software, which is so complex that it requires the establishment of an absolutely new theory system for trustworthiness measurements in the Internet and cloud computing environment.

To solve these problems, it is necessary to have a clear understanding of the nature of "trust" and definition of trustworthiness, which is the exact focus of this paper. We proceed under the belief that conceptual work is a scientific pursuit as valid as empirical testing of theory. From the perspective of sociology trust theory, the paper further reveals the essential definition of software trustworthiness based on the existing research results. This paper first presents a measurable S2S (social-to-software) software trust framework (STF) based on the trust theory of humanities and sociology, which makes a model for the essential inherence of trustworthiness consisting of three parts of software history information or evidence: identity evidence, basic standard or norms evidence, and ability evidence. In order to measure and combine these three parts of trustworthiness, the paper introduces a generalized indicator, which improves the completeness of trustworthiness metrics. Finally, the paper proposes a whole metric solution for the S2S software trust framework and unifies all different parts of metric results, which is significant and important to compare different software not only from the vertical perspective (comparison in the different attributes within software) but also from the horizontal perspective (comparison between different software).

In the software trustworthiness field, the paper first proposes the conceptual framework (STF) of software trustworthiness from the aspect of trust theory based on humanities and sociology. We cooperate with China Information Technology Security Evaluation Center (CNITSEC), trying to develop the industrial measurement standardization and formulate the industrial rule set for basic standards trustworthiness. We attempt to speed up the standardization process of the trustworthiness evaluation work. At present, STF and some measurement solutions have been accepted by CNITSEC and have been put into practical applications.

2 Research Background and Motivation

During recent decades, software trust in information systems enabled situations has caught the attention of increasingly more authorities and researchers. As for research on software trustworthiness, a number of definitions have been proposed by some international authority organizations, including ISO/IEC①, CNSS②, TCG (Trusted Computing Group)③, NRC (National Research Council)[12], TruSoft[13], and NIST[14]. For example, ISO/IEC① proposed "a trusted component, operation, or process is one whose behavior is predictable under almost any operating condition and which is highly resistant to subversion by application software, viruses, and a given level of physical interference." TCG③ claimed "an entity is trustworthy if it always behaves according to its defined goal as expected." And NRC[12] presented "trustworthiness is assurance that a system deserves to be trusted — that

①ISO/IEC 15408-1. Information Technology — Security Techniques — Evaluation Criteria for IT Security (4th edition). Part 1: Introduction and General Model. 2012.

②Committee on National Security Systems (CNSS). National Information Assurance (IA) Glossary, Instruction 4009, Revised June 2006. http://www.cnss.gov/Assets/pdf/cnssi_4009.pdf, Dec. 2017.

③Trusted Computing Group. TCG Architecture Overview, v1.4, https://trustedcomputinggroup.org/wp-content/uploads/TCG_1_4_Architecture_Overview.pdf, Dec. 2017.

it will perform as expected despite environmental disruptions, human and operator errors, hostile attacks, and design and implementation errors. Trustworthy systems reinforce the belief that they will continue to produce expected behavior and will not be susceptible to subversion." Although each definition fits its specific research context, the scopes they defined are incompatible and even contradictory when examined together according to the definitions of quality attributes from ISO standard[15], as shown in Fig.1.

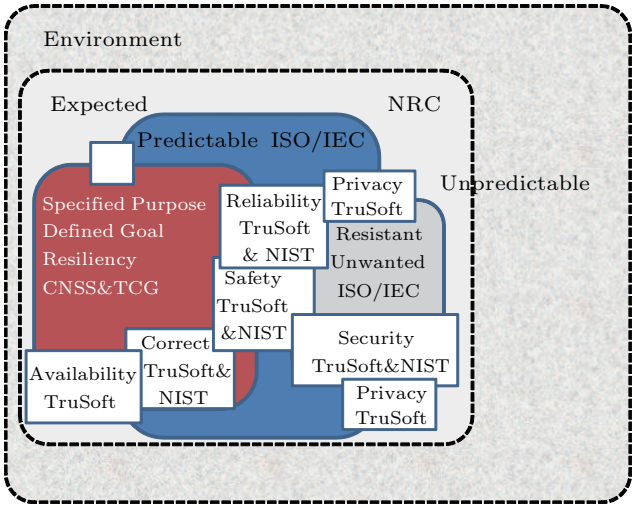

Fig.1. Graphical form of different definitions.

A possible reason for this incompatibility is that the definitions fail to account for the "nature" of trustworthiness. Without a clear understanding of the nature of software trustworthiness and its boundary, it would be impossible to measure trustworthiness scientifically and effectively.

As is known to all, the word "trust", originating from sociology④, is a moral value, regarded as a virtue. It is a reaction in the human brain of the objective reality[8] and an evaluation of others from social contract relationship. Likewise, as a product of human thinking, software has much in common with human society. In fact, it is the informatization of human requirements, reflecting the characteristics of human thinking and behavior, while the users of software are also human. The subtle relationship between software and human gives us certain reasons to believe that we must study the nature of "trust" from the perspective of humanities and sociology.

In another side, in traditional research, software indicators are usually quantified based on reliability models[16-17]. Nevertheless, a single attribute measurement of reliability does not meet the evaluation requirements of complex software. Therefore, in recent years, scholars worldwide have carried out research for the metrics method of overall credibility in software and have developed software trustworthy metrics models[18-20] from different aspects. However, all the proposed models have certain limitations: 1) some models rely on artificial and subjective evaluation and lack objectivity; 2) some models take trustworthiness as a single attribute, without considering the severity of the problem of affecting the software's overall credibility; 3) different trustworthy attributes have different metric standards and indicators, which makes it difficult to compare one another.

All these can be largely attributed to the immaturity but rapid advances in this field. Therefore, to develop a standard, measurable and integral framework of trustworthy based on a strict and complete "trust" system definition has been the focus of extensive and ongoing research on software trustworthiness. The first purpose of this paper is to design an integrated software trustworthy framework, namely the social-to-software (S2S) framework, based on humanities and sociology trust theory and the achievements of Yang *et al.*[10-11,20-22]. The other purpose is to present a new unified measure indicator, defined as loss ω, which covers all different trustworthiness parts and makes it possible to establish a whole solution with mathematical methods and models to evaluate software trustworthiness.

3 S2S Software Trust Framework

3.1 Humanities and Sociology Trust System Model

A consistent conceptual model helps researchers by linking science to the real-world upfront, so that the results of the scientific research may later be of great use for practice. In terms of trust, researchers of humanities and social science have broad and deep discussions of the issue and thus have universal and rather good understanding of it, which can help us to reveal the "natural" of software trustworthiness. Their research has resulted in a well-established system of trust[23-26], the Humanities and Sociology Trust System Model (HSTM) (see Fig.2).

In HSTM, trust is initiated by the trustor based on the perception of the "trustworthiness" of the trustee.

④Trustworthiness. http://en.wikipedia.org/wiki/Trustworthiness, Dec. 2017.

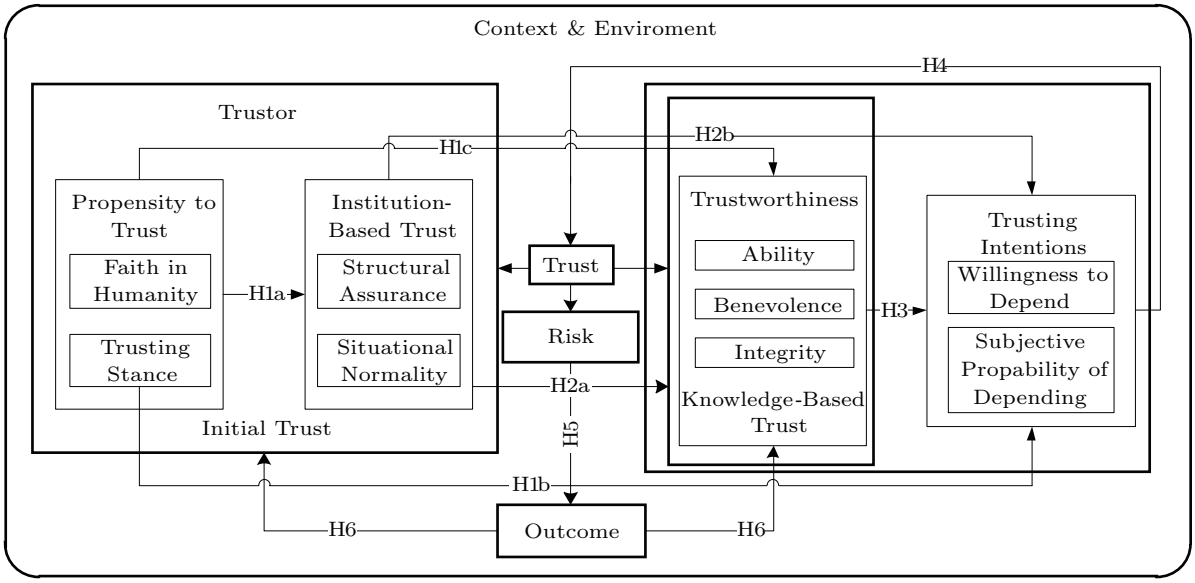

Fig.2. Humanities and sociology trust system model.

One factor that will affect the trust one party has for another involves traits of the trustor, which are referred to as the propensity to trust and the institution-based trust[24,26] (shown as Fig.2). Propensity to trust means the extent to which one displays a consistent tendency to be willing to depend on others in general across a broad spectrum of situations and persons, which is proposed to be a within-party factor that will affect the likelihood the party will trust. The higher the trustor's propensity to trust, the higher the trust for a trustee prior to the availability of information about the trustee. Institution-based trust means one believes that favorable conditions are in place that is conducive to situational success in an endeavor or aspect of one's life. For example, in the Internet context, "favorable conditions" refer to the legal, regulatory, business, and technical environment perceived to support success. Institution-based trust in technology will positively affect trust in a specific technology.

One approach to understanding why a given party will have a greater or smaller amount of trust for another party is to consider characteristics of the trustee. Obviously, the characteristics of trustee are the core of our research, that is, the trustworthiness of software (trustee). Clearly, then, the concept of trustworthiness is central to understanding and predicting trust levels[23-26].

[23-26] take a statistical analysis on a large number (more than 100 literatures) of social science, psychology and other humanities literatures, and draw a conclusion: ability, benevolence and integrity (as shown in Fig.2) cover 91.8% of the characteristics-based trust definitions found. As a set, these three appear to explain a major portion of trustworthiness[24,26]. Each contributes a unique perceptual perspective from which to consider the trustee, while the set provides a solid and parsimonious foundation for the empirical study of trust for another party. Ability has become one of the more commonly discussed components of trustworthiness[23-26], which captures the knowledge and skills needed to do a specific job along with the interpersonal skills and general wisdom needed to succeed in a risky situation. Benevolence is the extent to which a trustee is believed to want to do good to the trustor, aside from an egocentric profit motive[24], with synonyms including loyalty, openness, caring, or supportiveness. Integrity is defined as the extent to which a trustee is believed to adhere to sound moral and ethical principles that the trustor finds acceptable, with synonyms including fairness, justice, consistency, and promise fulfillment[23-26]. Therefore, trust for a trustee will be a function of the trustee's perceived ability, benevolence and integrity. The ability, benevolence and integrity components of trustworthiness all have significant, unique relationships with trust.

Based on the knowledge of trustee and trustor themselves, the trustor will have a further judgment — trusting intentions. Trusting intentions mean that one is willing to depend on, or intends to depend on the other party even though one cannot control that party. Definitions of trusting intentions embody three elements synthesized from the trust literature.

As shown in Fig.2, the trustor has his/her propensity to trust, which will positively affect institution-based trust (arrow H1a). And his/her propensity to trust will also positively affect trusting intentions (post adoption of trustee, arrow H1b) and exert a mediated, positive effect on specific trustee's trustworthiness (arrow H1c). Similarly, the trustor's institution-based trust will positively affect specific trustee's trustworthiness (arrow H2a) and will exert a mediated, positive effect on trusting intentions (arrow H2b). However, whether the trustor will trust the trustee or not also depends on the trustee's characteristics. The trustor acquires more knowledge of the characteristics (trustworthiness) of the trustee through further communication and exchange with the trustee. Obviously, the trustworthiness of the trustee will positively affect the final trusting intentions (arrow H3). Trusting intentions will decide whether the trustor will choose to trust the trustee or not (arrow H4). And once choosing to trust, the trustors need to assume the risk arising from the trust, that is, behind the trust behavior, there must be a risk, more or less. And the outcome of the risk will affect the characteristics of the trustor and the trustworthiness of the trustee (arrow H5 and arrow H6), so as to form a dynamic evolution of trust system.

The trust theory system of humanities and sociology is a very complicated subject. We read hundreds of articles and papers to summarize and extract HSTM. Therefore, we discuss this complicated and difficult subject in another paper[10] separately because of limited article space. For more detailed analysis and demonstration have been detailed, please refer to [10].

3.2 Mapping to the S2S Framework

By integrating this HSTM into the research on software trustworthiness, our research group has accumulated a wealth of experiences and achievements. Through strict inference, deduction, and demonstration, HSTM is further mapped to S2S software trust framework (STF)[10] as shown in Fig.3. As for how HSTM maps to STF, it is another complex topic. More detailed content please refer to [10].

In this framework, AEA is the third-party authority, accredited by the government with powerful knowledge and testing tools, which evaluates the trustworthiness of software system $T(a)$ on behalf of regular software users J_i, providing the evaluation results to J_i to judge $T(a)$. Historical running records or evidence of trustworthiness d_i is the essential property of software system, which is influenced by STR (software trustworthiness requirement) dynamically. Like the trusted problem of human, we focus only on its essence. It is the basis on which AEA evaluates $T(a)$. Obviously, $T(a)$ is related to and impacted by the context and environment $ET(t,r)$ (here, t means time and r means the relationships among different elements helping to run the software system, including software users, software itself and AEA). $ET(t,r)$ will finally influence the evidence

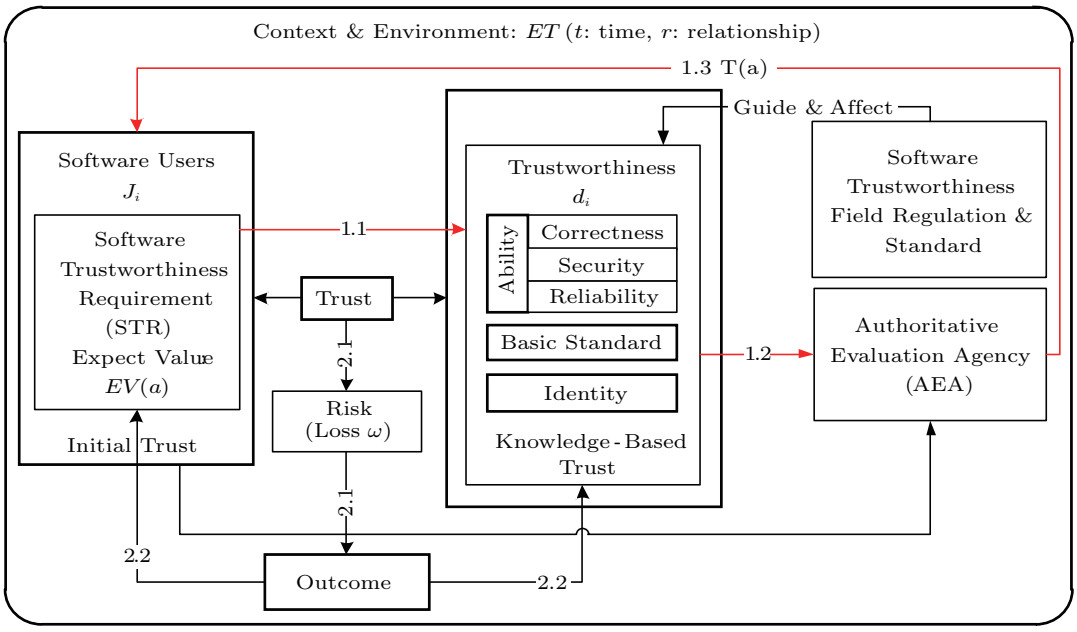

Fig.3. S2S software trust framework.

provided by $T(a)$. Besides, the model is related to time t, whose attributes only emerge in runtime. Therefore, $T(a)$ is a function of time t, indicated as $T_a(t)$.

Step 1.1. Like human interaction software users J_i have their expected value $EV(a)$ of STR (like the trustor's propensity or institution-based trust), which will influence $T(a)$.

Step 1.2. All of d_i (including ability trustworthiness, indicated as $A(T_a(t))$, basic standard Trustworthiness, indicated as $S(T_a(t))$, and identity trustworthiness, indicated as $I(T_a(t))$) will be collected and combined to provide to AEA.

Step 1.3. Through professional assessment tools and knowledge-base, AEA will propose the final assessment results $T_a(t)$ to J_i.

Once $T_a(t)$ is greater than $EV(a)$.

Step 2.1. J_i will choose to "trust" the software and accept it, but be on risk-taking at the same time. Otherwise, J_i need to return to the original circle (from step 1.1).

Step 2.2. The outcome of it will not only affect the subsequent STR but also affect $T_a(t)$, if the risk occurs.

From the above, it proves further that discussing the nature of trustworthiness from the perspective of humanities and sociology is not only sufficient but also necessary.

Obviously, the software trust framework is a whole complicated system, even a dynamic evolution system, rather than a single trustworthiness research. Thus we cannot split the trustworthiness from other factors. However, all of the impact of external factors will ultimately reflect into the trustworthiness as forms of running records, data, results and evaluation reports. Therefore we need to focus on the metrics of the software trustworthiness.

Before giving the definition of software trustworthiness, we need to introduce some concepts and definition that will be used later. According to the IEEE Std 610-12-1990, we have the following definitions.

Definition 1 (Fault). *An incorrect step, process, or data definition in a computer program.*

Definition 2 (Failure). *The inability of a system or component to perform its required functions within specified performance requirements.*

Definition 3 (Disfunction). *The inability of a system to perform its required functions when encountering a fault.*

Note that there are differences between failure and disfunction. If there is error tolerant design in development, the software can have failure but no disfunction; otherwise failure becomes disfunction. Definition 1 has its limitations, thereby we give an expanded definition to serve for the software trustworthiness better.

Definition 4 (Security Fault). *An incorrect step, process, data definition or design, encode and data definition that triggers system security issue by disobeying the principle and tactic of system security in a computer program.*

The fault of software in this paper refers to Definition 4. Based on that, we give a definition of vulnerability.

Definition 5 (Vulnerability). *If the fault in software does not cause disfunction in a normal situation (it has no influence on the function of software), but may be used by attackers to execute extra baleful code, or leak out information, then this fault is called vulnerability. And this phenomenon is called invasion.*

Definition 6 (Trustworthiness). *Formalized as $GT(T_a(t))$, the trustworthiness of software $T(a)$ is the property of being able to gain the trust of J_i, with three parts of evidence as follows: identity evidence, basic standard or norms (which is the basic rule of the software industry) evidence and ability evidence in demanded environment $ET(t, r)$ and during demanded time period t even with other factors (like inappropriate operation) interfering or man-made attack.*

Note. Identity evidence includes the software $T(a)$ itself and code source, shared resource of code, document to use, configuration of hardware and software, illustration of running environment, help document and so on. Basic standard evidence refers to that the design and requirement match with basic standard of the trade or the principle, and the tactic of system safety, so that it will not bring system safety issue or make other damage or problem (like leaking private information). Ability evidence refers to the reliability on broad sense which means the extent that the software does not cause the failures of the system in the specified environment and time (defined as generalized reliability, i.e., G-reliability, indicated as $AR(T_a(t))$), anti-attack ability of software which means the extent that the software does not cause the system invasion (defined as generalized security, i.e., G-security, indicated as $AS(T_a(t))$), and ability of restoring the customer's requirement correctly (defined as generalized correctness, i.e., G-correctness, indicated as $AC(T_a(t))$).

Definition 7 (Loss ω). *Because of the existence of non-trust factors in software, the users of software must pay an extra cost (including money, maintenance workload, effort, lines of code, and so on), which is com-*

posed of three parts: 1) *in order to achieve the specified requirement, the extra cost is due to the software without satisfying the basic standards;* 2) *in order to restore the software running, the extra cost is due to the wrong running of software caused by trusted problems;* 3) *the extra cost of the user's business loss is caused by failure operation.*

Now, the quantifications of the three parts of trustworthiness are all integrated into loss ω. Therefore $GT(T_a(t))$ can be deduced as:

$$GT(T_a(t)) = \alpha I(T_a(t)) + \beta S(T_a(t)) + \gamma A(T_a(t)),$$

where $\alpha+\beta+\gamma = 1$ and α, β and γ represent the corresponding weights respectively. Different software users have different requirements for the parts of trustworthiness, thus, it is very necessary to assign an appropriate weight for different parts of trustworthiness. According to Definition 7, the three parts of trustworthiness could be finally calculated into the same unit of loss ω. Therefore, the higher $GT(T_a(t))$ is, the lower the trustworthiness will be.

Based on the theoretical derivation and applications, we propose the following solutions to measure the three trusted properties: 1) putting forward the ability trustworthiness measurement method based on time-loss rate and power function to get the value of $A(T_a(t))$; 2) establishing the rule sets in order to get the value of $S(T_a(t))$ based on multiple short-board measure method; 3) getting the value of $I(T_a(t))$ depending on the code homology detection tool and its measure model.

4 Proposed Metrics Solution

4.1 Related Work in Trustworthiness Metrics

As for the measurement and evaluation of trustworthiness, National Institute of Standard and Technology (NIST)[14] believes that the software trustworthiness can be measured and there may be different quantifiable representations of trustworthiness. It proposed a preliminary evaluation framework, which described the characteristics of software systems and will help to identify and improve metrics and measurement methods. [27] brings out a method on quantitative software trustworthy evaluation across software development life cycle based on knowledge discovery in database techniques and evidence theory. [28] proposes a trustworthiness evaluation mechanism by incorporating non-negative constraints and similarity with existing mechanisms. [29] addresses the service selection issue via a service agent framework coupled with a computational model based on trust ontology. From the software architecture, [30] proposes an evaluation model of dependability for Internetware based on Bayesian network.

It is known that the software trustworthiness is gradually developed from the traditional single attribute (mainly reliability and security) to the complex high-composite concept, and the related measurement methods always have growth together. In terms of software reliability, researchers have put forward many different measurement models since 1960s. J-M model is the earliest software reliability model[9]. G-O model[31] is the first NHPP (non-homogenous Poisson process) model, which believes that the cumulative number of failures obeys the Poisson distribution and the total number of failures is a random variable. Musa model[32] uses CPU executing-time to measure the reliability based on the relationship between the execution time and the calendar time. S-W model[33] believes that the probability of the found fault will increase with the development of software testing while the failure rate is proportional to the residual fault and the test time, and the failure interval follows Weibull distribution. Nuria *et al.* estimated the number of software failure through the Bayesian model with applying the Gauss process[34]. Software vulnerability serves as the main influence on software security and always represents the significant risk in the software[35]. Therefore, about the metrics of software vulnerabilities, there are some representative models of thermal dynamic model[36], logarithmic Poisson model[37], exponential model[38], and so on. These models take time as an important parameter and try to predict the number of software vulnerability.

As a complex high-composite concept, the measurement of trustworthiness also has been researched deeply by academic. Our team[39] proposed a comprehensive measurement model (including five trustworthy attributes), which is based on the fuzzy comprehensive evaluation method. [40] proposes a layered software trustworthiness classification model, which defines the trustworthiness attributes model and the classification levels, establishes a reference model for software trustworthiness evidence, and describes the intrinsic relations between them. [41] provides an evidence model for software trustworthiness evaluation in a cognitive and cost-progressive way and proposes three profiles, namely reputation, experience and mechanism. [42] proposes a new software trustworthiness evaluation approach for open source system.

However, the above research results have the follow-

ing problems: 1) they fail to consider the effect caused by the severity[3-6] of attributes which can bring about the loss of cost; 2) they lack a unified indicator and process to integrate different aspects of trustworthiness; 3) many evaluation methods are based on the specific software framework or software types, such as resource sharing, embedded software, and so on, which are not universal; 4) some evaluation models usually depend on human subjective assessment, while few models from mathematics aspect.

Subsection 4.2 elaborates the combined trustworthiness value $GT(T_a(t))$ based on loss ω, which tries to solve the problems mentioned above.

4.2 Ability Trustworthiness Metrics Solution

According to STF, the ability trustworthiness includes correctness, reliability and security. ISO/IEC[43] defines "correctness" as: "the capability of the software product to provide the right or agreed results or effects with the needed degree of precision", which belongs to software functions and requirements review job. Therefore we do not discuss the correctness measurement and assessment in this paper.

It must be noted that traditional measurement methods are usually defined as a function taking the occurrence time of failures or vulnerabilities as a variable to predict the total number of failures or vulnerabilities and their future occurrence time. Nevertheless, it is not sufficient to decide software reliability or security by simply depending on future occurrence time and the amount of failures or vulnerabilities. Therefore, we should examine not only the emergence rate of failures or vulnerabilities, but also the loss caused by the severity of failures and vulnerabilities. Therefore, we have the definitions below.

Definition 8 (G-Reliability). *G-reliability is, in the specified environment and prescribed time, the probability of the severity of system failure which is not caused by software* (*confer to Definition* 2).

Definition 9 (G-Security). *G-security is, in the specified environment and prescribed time, the probability of the severity of system invading which is not caused by software* (*confer to Definition* 5).

4.2.1 Mathematical Measurement Model

We propose a mathematical model to measure $A(T_a(t))$ based on power function theory and time-loss rate, which is a new indicator.

Definition 10 (Time-Loss Rate ξ). *The time-loss rate of a failure or vulnerability* $\xi = \frac{T}{L}$, *where* T *is the time interval, and* L *is the loss-amount.*

In fact, time-loss rate is a specific time, which accords with the J-M Model's[9] basic assumption. However, the linear function of the J-M model causes some problems in the actual application process. For example, (1) due to that the inherent error number N_0 is a constant in the J-M model's basic assumptions, there will occur an impossible condition that the predicted time is a negative while the predicted errors are more than N_0; 2) the J-M model based on linear decline function cannot accurately predict the failure (vulnerability) rate of growth. However, the power function is better to deal with these two issues. Therefore, choosing the power function to replace the linear function will be better to fit our J-M improvement model.

Here, the J-M improved model's basic assumptions are as follows. 1) Each error is independent of one another, and the time interval of the error occurrence is also independent of one another; 2) the detected errors are immediately removed, no error is introduced again, and the debugging time needs to be ignored; 3) the G-failure is a constant during every error interval time, and its size and number satisfy the power function.

We can define G-failure as below:

$$\lambda(x_i) = a \times i^b.$$

And then the G-reliability of ξ is:

$$R(x_i) = \exp(-a \times i^b \times x_i).$$

The expectation of time-loss rate $E(\xi)$ as mean loss rate between failures (MLBF) is:

$$E(\xi_i) = (MLBF)_i = \int_0^\infty R(x_i)\mathrm{d}x = \frac{1}{\lambda(x_i)} = \frac{1}{a \times i^b}. \quad (1)$$

And we know mean time between failures (MTBF) is:

$$(MTBF)_i = \frac{1}{c \times i^d}. \quad (2)$$

Then, the joint-probability density function is:

$$f(x_1, x_2, \ldots, x_n) = a^n n!^b e^{-a(\sum_{i=1}^{n} i^b x_i)}.$$

Combining (1) and (2) can predict the loss-amount of the n+1 failure:

$$L_{n+1} = \frac{(MTBF)_{n+1}}{(MLBF)_{n+1}} = \frac{a(n+1)^{b-d}}{c}.$$

Therefore, we can get:

$$A(T_a(t))_{n+1} = A(T_a(t))_n + L_{n+1}. \quad (3)$$

4.2.2 Parameters Estimation

According to (3), we define the failure rate of experience as:

$$\lambda_{Ei} = \frac{1}{(MLBF)_i} = \frac{1}{x_i} = \frac{L_i}{t_i}.$$

After λ_{Ei} is obtained through the history datasets, it can further estimate parameters a and b by using the least squares method which means that the overall solution minimizes the sum of the squares of the errors $S(a,b)$ made in the results of every single equation.

$$S(a,b) = \sum_{i=1}^{n} (\lambda_{Ei} - \lambda(x_i))^2 = \sum_{i=1}^{n} \left(\frac{L_i}{t_i} - a \times i^b \right)^2.$$

Now, it can get the estimation of parameters a and b, indicated as $\hat{a}$ and $\hat{b}$ respectively. Then:

$$(MLBF)_{n+1} = \frac{1}{\hat{a}(n+1)^{\hat{b}}}.$$

Similarly, it can also obtain:

$$(MTBF)_{n+1} = \frac{1}{\hat{c}(n+1)^{\hat{d}}}.$$

Then, the predicted loss is:

$$L_{n+1} = \frac{\hat{a}(n+1)^{\hat{b}-\hat{d}}}{\hat{c}}. \tag{4}$$

Accordingly, the accumulated loss is:

$$\hat{A}(T_a(t))_{n+1} = \hat{A}(T_a(t))_n + \frac{\hat{a}(n+1)^{\hat{b}-\hat{d}}}{\hat{c}}. \tag{5}$$

And the predictive value of the cumulative loss when the $n+1$ fault happened is:

$$\hat{W}_{n+1} = \hat{W}_n + \hat{A}(T_a(t))_{n+1}.$$

Summary. We put forward a G-reliability (G-security) measurement solution based on a new concept time-loss rate and power function. We introduce a measurement indicator loss. Our metric solutions not only can predict the count and the time of software failures (vulnerabilities), but also can predict the loss (severity) caused by software failures (vulnerabilities). Their significance is mainly reflected in the following aspects. 1) The new indicators improve the completeness of trustworthiness metric. The traditional reliability (security) model based on failure (vulnerability) time is a little unilateral. But the cumulative loss $A(T_a(t))$ and time-loss rate are proposed to enrich the content of the reliability (security) metric. 2) The introduction of loss ω not only considers the severity caused by failures (vulnerabilities), but also unifies the scale of the trustworthiness measurement. The loss is a real practical metric indicator which could be applied into software contrast, project development and financial planning, etc.

4.3 Basic Standard Trustworthiness Metrics Solution

According to Definition 6 and Definition 7, basic standard trustworthiness has its industrial domain characteristics because different industrial fields have their different basic standards. This subsection discusses an $S(T_a(t))$ metric method of multi-short boards effect based on different industry rules devised by different industrial professional experts.

4.3.1 Trustworthiness Rules Measurement

First, we create the basic standard rule sets for different industries. When evaluating trustworthiness, several different experts are invited to score each rule on a scale of 1∼10 to get a judgment matrix. We measure the trustworthiness rules using fuzzy comprehensive evaluation (FCE) that is a comprehensive method on a system's measurement based on fuzzy set theory.

1) For a trustworthiness rule, we define an evaluation factors set according to the sub-rules set and score them:

$$U = \{U_1, U_2, \ldots, U_n\}.$$

2) According to the importance of each sub-rule, the weight vector is:

$$\boldsymbol{A} = (a_1, a_2, \ldots, a_n), con. \sum_{i=1}^{n} a_i = 1.$$

3) In order to generate the value more reasonably, we propose experts grading method. Given N experts, we define the expert number who puts the value j for sub-rule i as r_{ij} (i is in $[1,n]$, j is in $[1,10]$). The measurement result of the sub-rule i is defined as follows:

$$\boldsymbol{R}_i = (r_{i,1}, r_{i,2}, \ldots r_{i,10}), con. r_{ij} = \frac{N_{ij}}{N}.$$

4) Set up a judgment matrix:

$$\boldsymbol{R} = \begin{pmatrix} \boldsymbol{R}_1 \\ \boldsymbol{R}_2 \\ \vdots \\ \boldsymbol{R}_n \end{pmatrix} = \begin{pmatrix} r_{1,1} & r_{1,2} & \cdots & r_{1,10} \\ r_{2,1} & r_{2,2} & \cdots & r_{2,10} \\ \vdots & \vdots & \ddots & \vdots \\ r_{n,1} & r_{n,2} & r_{n,3} & r_{n,10} \end{pmatrix}.$$

5) We define the fuzzy comprehensive evaluation sets $\boldsymbol{B}$ as:

$$\boldsymbol{B} = \boldsymbol{A} \cdot \boldsymbol{R} = (b_1, b_2, \ldots, b_{10}),$$

where b_j is the probability of this sub-rule over value j.

6) Let b_j as the weight of value j, and we can compute the final score of this trustworthiness rule as:

$$\boldsymbol{v} = \sum_{j=1}^{10} b_j \times j.$$

Iterating the steps above, the final value of every trustworthiness rule is derived. Besides, the experts' grading method makes the result reliable.

4.3.2 Basic Metric Model and Improvement

In order to obtain the final value of $S(T_a(t))$, we introduce a basic and matured attributes model[27] of software trustworthiness. Coupled with the consideration of realities and multi-short board effects, we present a more complete and advanced $S(T_a(t))$ metric model.

1) We propose a basic metric model on the basis of metric model in [44] and its developed model in [45]. We define a function T to represent the final trustworthiness value as $T = T(y_1, y_2, \ldots, y_n)$[46]. We also define a weight value to individual trustworthiness rule. Assuming that each rule has m sub-rules, their corresponding weights of the sub-rule are defined as $\alpha_1, \alpha_2, \ldots, \alpha_m$, where

$$\sum_{i=1}^{m} \alpha_i = 1.$$

If the i-th trustworthiness rule value is minimal, it is denoted as $i = min$; otherwise $i \neq min$. Then we define the basic metric model for trustworthiness as:

$$T = 10(\frac{y_{min}}{10})^{\varepsilon} \times y_1^{\alpha_1} \times y_2^{\alpha_2} \times \ldots \times y_m^{\alpha_m},$$

where T denotes the final value of the trustworthiness, y_i is the measurement value of the rule i, α_i is the corresponding weight, and y_{min} is the minimal measurement value. Then we interpret y_{min} as the shortest board of the bucket in terms of buckets effect, which decreases the value of T, e.g., the effect of short board. The control parameter $\varepsilon (0 \leqslant \varepsilon \leqslant 1$ and $\alpha_{min} + \varepsilon \leqslant 1)$ is used to constrain the effectiveness of the minimal rule on the final trustworthiness score T: The larger ε is, the more effective y_{min}. Especially, this metric model is not completely equal to the bucket effect although it is based on the bucket effect. Our method gives that: rules whose metric values are above the minimum contribute to the final trustworthiness value.

2) As mentioned above, a fixed parameter is used to constrain the effectiveness of the minimal rule on the final trustworthiness score T. The larger ε is, the greater the effectiveness of y_{min} is.

Obviously, each minimal value influences on T separately as we do not know the rule with the minimal trustworthiness value in advance. We consider the control parameter as a function, which is related to the weight of the rule with the minimal value, other than a fixed value. Intuitively, those rules that have larger weight would result in larger effectiveness. Therefore, the new model generated from improvement 1 is:

$$T = 10(\frac{y_{min}}{10})^{k\alpha_{min}} \times y_1^{\alpha_1} \times y_2^{\alpha_2} \times \ldots \times y_m^{\alpha_m},$$

where k is a fixed parameter to constrain effectiveness of the weight value of which has a minimal final trustworthiness score and $k\alpha_{min}$ constrains the effectiveness of the rule with minimal value on T. $k\alpha_{min}$ is defined in the range [0, 1] and with $\alpha_{min}(1+k) \leqslant 1$.

3) Actually, there exists more than one board with the minimal value. For instance, multiple rules with minimal trustworthiness score produce effects on the overall trustworthiness which we can explain using a bucket. It is hard to tell which board determines the volume for a bucket when there exists more than one shortest board. If all the shortest boards are joined together, they should be regarded as a unique one. Thus, it is asserted that if there is more than one rule with the minimal trustworthiness value in our model, we describe them into a new rule jointly. The weight value of a new rule is the summation of the original weights. Formally, we call it an "accumulation board" which explains the new rule to become a new board of the trustworthiness.

The new model generated from improvement 2 is:

$$T = 10(\frac{y_{min}}{10})^{k \sum\limits_{j=min} \alpha_j} \times y_1^{\alpha_1} \times y_2^{\alpha_2} \times \ldots \times y_m^{\alpha_m}.$$

In the new model, there may exist more than one rule whose trustworthiness value is minimal, and $0 \leqslant k \sum\limits_{j=min} \alpha_j \leqslant 1$, for any rule that $j = min$, $(k+1)\alpha_j \leqslant 1$.

4.3.3 Metric Model Considering Effect of Multiple Short Boards

The short board is the minimal trustworthiness value of a rule, and is derived from objective measurement. Generally, we presume upon a value for each rule, which could result in trustworthiness' subjectivity.

For example, there are two rules, one's trustworthiness value is 5 and the other's is 7, which seems good enough. In fact, it is still under expectation and has big effect on the whole trustworthiness metric, even if 7 is above the minimal value. As a result, we consider this rule a short board for its subjectivity.

Under this point, the concept of short board has expended from the rule with the minimal value to the rule that cannot reach the people's expectation. All short boards that cannot reach the lowest expectation contribute to decreasing the volume of the bucket. The real volume is only affected by the shortest board, whereas the expected volume is determined by all the boards whose heights are under the lowest expectation. In general, we define a threshold for a trustworthiness rule as the lowest expectation of this rule in advance. Accordingly, such rules whose value cannot reach the threshold, minimal or not, decrease the whole trustworthiness correspondingly. We define these rules as "short board rules", which help to reduce the trustworthiness as a whole and the reduced trustworthiness become larger when there exist more short board rules or the values of these rules are lower.

The proposed model should be refreshed with a threshold. Metric model for trustworthiness formulates the effects of multiple short boards:

$$\begin{cases} S\left(T_a\left(t\right)\right) = \theta\left(T_h\right) \\ = \theta(10 \times \prod_{i=1}^{n} f\left(y_i\right) \times y_1^{\alpha_1} \times y_2^{\alpha_2} \times \ldots \times y_n^{\alpha_n}), \\ f\left(y_i\right) = \begin{cases} \left(\frac{y_i}{10}\right)^{k\alpha_i}, & \text{if } \ y_i < h, \\ 1, & \text{otherwise}, \end{cases} \end{cases}$$

$$con0 \leqslant k \sum_{y_i} < h\alpha_i \leqslant 1 \text{ and } (k+1)\alpha_i \leqslant 1. \qquad (6)$$

Let h be the predefined threshold, and T_h be the trustworthiness score with h. We define $f(y_i)$ as the short board function. If the value of the trustworthiness rule is greater than and equal to h, and the result of the short board function is equal to 1, it means no short board effect. Otherwise the result of the short board function should be a value less than 1 which reduces the score of T_h so that we can consider these rules coming into short board effect.

Summary. On the basis of the research of attributes partition method in software trustworthiness, we apply FCE method in the measurement of basic standard rules. After that, we propose a basic metric model and make improvements. Ultimately, by combining the bucket effect in practical use, we develop the basic model to a comprehensive model considering the effect of multiple short boards. But the FCE method mentioned in this paper is based on experts' grading result, which belongs to the indirect measurement method. It will be the focus of future work to find a direct measurement method to get the value of trustworthiness rule. Meanwhile, our metric model of the effect of multiple short boards needs a predefined threshold. In practical use, people have different expectations on different rules separately. Therefore, as a future development, we can consider setting up an independent threshold for each rule. Finally, we need to note that the value of $S(T_a(t))$ is also based on loss ω which unifies all parts of trustworthiness.

Although further optimization work is needed, we believe that our metric model will help to speed up the standardization process of the evaluation work on basic standard trustworthiness. It will contribute to make the modern product design process safer and more effective.

4.4 Identity Trustworthiness Metrics Solution

With the continuous development of software reuse technology and the increasing of the number and quality of open source software, the homology codes' similarity in different kinds of software is also getting higher and higher. The reuse or plagiarism of software codes leads to the problem of software identity trustworthiness.

As mentioned above, identity evidence includes the software itself and code source, shared resource of code, document to use, configuration of hardware and software, illustration of running environment, help document and so on. Among these influence factors, the source code of the software is the most important representation of the software identity. And most of other evidences could be evaluated through the review meeting technology. Meanwhile, various studies show that much duplicated code exists in large code bases[47-48]. Much duplication can be attributed to poor programming practice since programmers often copy and paste code to quickly duplicate functionality. This tendency not only produces code that is difficult to maintain, but also may introduce subtle errors[48].

In terms of software identity trustworthiness, the main problem our team needs to face is how to detect the clone code more quickly and effectively, in which we have made preliminary progress[49-50]. And now we have our own first version of homologous code detecting tool. Therefore, once the homologous code is detected, the Loss caused by it will be easily measured.

We use professional code homology detecting tools to identify similar code and their percentages from the history software for the testing code. We assume that the loss ω caused by historical software during its lifecycle is proportional to the loss ω of the similar code part. Then we predict the testing software's loss ω caused by the identity of code based on historical data.

First, the following provisions are given: 1) T_a is on behalf of the current evaluation software; 2) T_i is a similar software with T_a through the detection of the code homology detecting tool, here $1 \leqslant i \leqslant N$. N is the total number of similar software with T_a. Correspondingly, ω_i represents a historical loss, whose measurement method is to summate all losses during the software's life cycle.

The measure model is:

$$I(T_a(t)) = \sum_{i=1}^{N} \omega_i \times \alpha_i \times \beta_i.$$

Here, α_i is the similarity rate, and β_i is the ratio of the similarity code to the total code.

5 Experiments and Evaluations

In the software trustworthiness field, it is the first time for us to propose the software trustworthiness measurement framework based on the trust theory of humanities and sociology and also the first time to propose the loss indicator to integrate all parts of the software trustworthiness. In this section, all experimental data is real and effective though it is not from the same one software, which is enough to verify the solution feasible. Some data is provided by China Information Technology Security Evaluation Center (CNITSEC).

5.1 Experiment and Evaluation of Ability Trustworthiness

Here, we need to use root-mean-square deviation (RMSD) to predict the results' precision. RMSD or root-mean-square error (RMSE) is a frequently used measure of the differences between values (sample values and population values) predicted by a model or an estimator and the values actually observed. Here, we define RMSD as:

$$RMSD = \sqrt{\frac{1}{n-m} \sum_{i=n+1}^{m} \left(A\left(T_a\left(t\right)\right)_i - \hat{A}\left(T_a\left(t\right)\right)_i\right)^2}.$$

5.1.1 Experiment of Failure Data

The failure data is derived from the distributed processing system in [38]. The loss data covers from 1 to 4, which represents different severity levels. Through the previous 117 pieces of history data and the least square method, we can estimate the parameters $\hat{a} = 3.508\,4$, $\hat{b} = -0.430\,7$, $\hat{c} = 1.131\,2$, and $\hat{d} = -0.389\,5$. Putting $\hat{a}$, $\hat{b}$, $\hat{c}$ and $\hat{d}$ into (4) and (5), it can obtain the predicted loss L_{n+1} and the accumulated loss $\hat{A}\left(T_a\left(t\right)\right)_{n+1}$, and the RMSD is 0.393 7 (small enough to ensure the validity of the forecast result). In the same way, we can get other predictive values, here $117 \leqslant n \leqslant 123$, and the predicted step is 2. The experimental data and predicted results are shown in Table 1. Fig.4 shows the comparison between the actual failure loss and the predicted loss. It can find that the 124th data is a singular point, and we will study it further in future work.

Table 1. Prediction of Accumulated Loss of Failure Data

No.	Days	Loss	Accumulated Loss (AL)	AL Prediction	RMSD
118	2 323	2	294	294.548 4	0.393 7
119	2 930	3	297	297.095 9	
120	3 110	2	299	299.549 5	0.746 8
121	3 321	4	303	302.098 0	
122	4 116	3	306	305.549 6	0.712 6
123	5 485	3	309	308.098 4	
124	5 509	3	312	311.550 6	1.380 4
125	6 150	4	316	314.100 3	

Note: Days: number of days between failures.

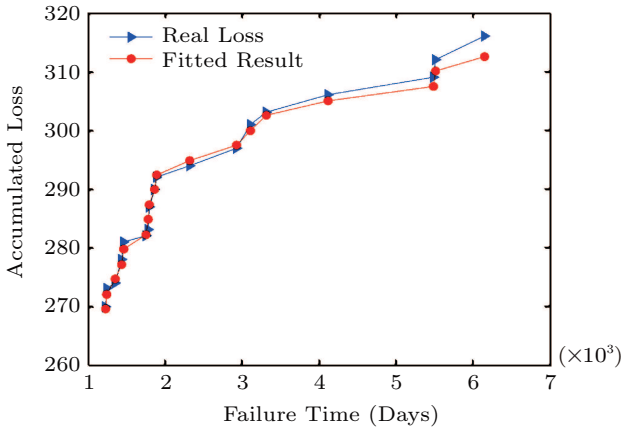

Fig.4. Comparison between AL (accumulated loss) and prediction of failure data. Fitted result means the result fitted by our model.

5.1.2 Experiment of Vulnerability Data

The vulnerability data comes from China National Vulnerability Database of Information Security (CNNVD)[5]. These vulnerabilities are divided into

[5]China Information Technology Security Evaluation Center. China national vulnerability database of information security. Internet draft. http://www.cnnvd.org.cn/, Dec. 2017.

four categories according to the severity and the most dangerous grade is 4. Similarly, through the previous 50 pieces of history vulnerability data, we can get the parameters $\hat{a} = 0.102\,3$, $\hat{b} = 0.286$, $\hat{c} = 0.042\,19$ and $\hat{d} = 0.230\,6$. Furthermore, we can obtain the accumulated Loss. The experimental data and predicted results are shown in Table 2. Fig.5 shows the comparison between the actual vulnerability loss and the predicted loss.

Table 2. Prediction of AL of Vulnerability Data

No.	Days	Loss	Accumulated Loss (AL)	AL Prediction	RMSD
55	2 454	4	157	156.023 3	0.691 5
56	2467	2	159	159.049 3	
57	2 488	3	162	162.071 5	0.605 8
58	2 505	4	166	165.146 3	

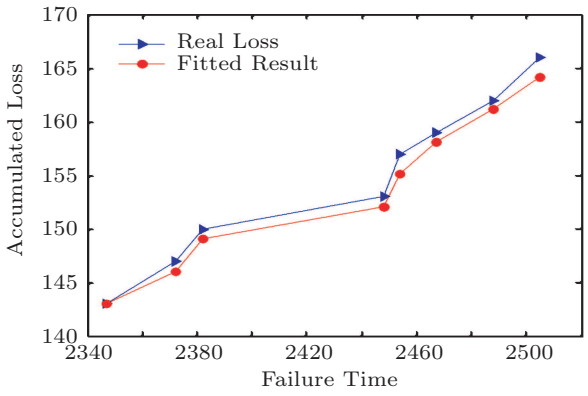

Fig.5. Comparison between AL and prediction of vulnerability data.

The experimental results show that the proposed mathematical model in Subsection 4.1 can obtain better results not only for failure data but also for vulnerability data. The power function makes up the limitation of the linear function of the J-M model when adapting itself to different datasets. Furthermore, short-term forecasting can improve the accuracy of the whole prediction.

5.1.3 Evaluation of Ability Trustworthiness

According to Definition 6, we define the ability trustworthiness evaluation framework as:

$$A(T_a(t)) = \rho A_f(T_a(t)) + (1-\rho)A_v(T_a(t)).$$

Here, $A_f(T_a(t))$ means G-reliability mainly based on the failure data, $A_v(T_a(t))$ means G-Security mainly based on the vulnerability data, and ρ is the weighted coefficient.

Now, we design the evaluation steps as bellow.

Step 1. Let $n = \min\{|S_f|, |S_v|\}$, where $|S_f|$ and $|S_v|$ represent the number of elements of the failure dataset and the vulnerability dataset respectively. We use the least square method to obtain the parameters of S_f and S_v.

Step 2. To obtain S_f and S_v, we need to calculate $\bar{\lambda_f} = \sum_{i=1}^{n} \lambda_{f,i}/n$ and $\bar{\lambda_v} = \sum_{i=1}^{n} \lambda_{v,i}/n$, where $\lambda_{f,i} = \hat{a}_f i^{\hat{b}_f}$ and $\lambda_{v,i} = \hat{a}_v i^{\hat{b}_v}$.

Step 3. While the number of failures and vulnerabilities is $1 \leqslant i \leqslant n$, we need to compute the mean time-loss rate of failure data and vulnerability data respectively: $\bar{x_f} = \sum_{i=1}^{n} x_{f,i}/n$ and $\bar{x_v} = \sum_{i=1}^{n} x_{v,i}/n$, where $x_{f,i} = \frac{t_{f,i}}{l_{f,i}}$ and $x_{v,i} = \frac{t_{v,i}}{l_{v,i}}$, where t is the time interval and l is the loss.

Step 4. Let $\bar{x} = \min\{\bar{x}_f, |\bar{x}_v|\}$. We can obtain the reliability-degree of S_f based on time-loss rate: $R_f(\bar{x}) = \exp(-\bar{\lambda}_f \bar{x})$, and the security-degree of S_v based on time-loss rate: $S_v(\bar{x}) = \exp(-\bar{\lambda}_v \bar{x})$.

Step 5. Let $A_f(T_a(t)) = \varphi(R_f(\bar{x}))$ and $A_v(T_a(t)) = \varphi(S_v(\bar{x}))$, and we can get the final $A(T_a(t))$ according to the parameter ρ based on the practical requirements. Here φ represents the mapping relationship from $R_f(\bar{x})$ (or $S_v(\bar{x})$) to $A_f(T_a(t))$ (or $A_v(T_a(t))$).

$A_f(T_a(t))$ and $A_v(T_a(t))$ take the three factors into account (loss, time and numbers, based on the new indicator time-loss rate), which has a wider significance and practical value compared with the traditional reliability just considering a factor of time.

5.2 Verification of Basic Standard Trustworthiness Metrics

This subsection verifies the correctness of our basic standard trustworthiness metrics model described in Section 4 by data simulation with several application examples. Firstly, it is assumed that there exist three rules, i.e., $m = 3$.

5.2.1 Verification of the Basic Model

Data from different groups are compared as shown in Table 3.

1) Comparing group 1 and group 2, when y_2 increases from 6 to 9, T increases. Therefoe T is monotonic increasing.

2) Comparing group 1 and group 3, when ε increases from 0.5 to 0.6, T degreases. It shows that the larger ε is, the more obvious the buckets effect is.

3) Comparing group 4 and group 5, when y_1 increases from 3 to 4, T increases by 0.129. The growth

is considerable. While in the comparison between group 1 and group 6, T just increases by 0.068 when y_1 increases from 8 to 9. With the same amount of increase of y_1, T increases less in the second data comparison. It can clearly be seen that the growth ratio of T becomes slow along with the increasing of y_i.

Table 3. Data Simulation for Basic Model

	y_1/α_1	y_2/α_2	y_3/α_3	ε	T
Group 1	8/0.3	6/0.4	2/0.3	0.5	2.822
Group 2	8/0.3	9/0.4	2/0.3	0.5	3.158
Group 3	8/0.3	6/0.4	2/0.3	0.6	2.537
Group 4	3/0.3	6/0.4	2/0.3	0.5	2.334
Group 5	4/0.3	6/0.4	2/0.3	0.5	2.463
Group 6	9/0.3	6/0.4	2/0.3	0.5	2.890

5.2.2 *Verification of the Model Improvement 1*

Data from different groups are compared as shown in Table 4.

Table 4. Data Simulation for the Model Improvement 1

	y_1/α_1	y_2/α_2	y_3/α_3	k_1	T
Group 1	8/0.3	8/0.4	5/0.3	1	6.040
Group 2	8/0.3	9/0.4	5/0.3	1	6.287
Group 3	8/0.3	8/0.4	5/0.3	2	5.076

1) Comparing group 1 and group 2, when y_2 increases from 6 to 9, T increases. Therefore T is monotonic increasing.

2) Comparing group 1 and group 3, when k increases from 1 to 2, T degreases. It shows that the larger k is, the more obvious the buckets effect is.

5.2.3 *Verification of the Model Improvement 2*

Data from different groups are compared as shown in Table 5.

Table 5. Data Simulation for the Model Improvement 2

	y_1/α_1	y_2/α_2	y_3/α_3	k_1	T
Group 1	8/0.3	6/0.3	5/0.4	1	5.099
Group 2	8/0.3	5/0.3	5/0.4	1	4.131

Comparing group 1 and group 2, when y_2 and y_3 are all equal to the minimal value in group 2, T decreases obviously. This is might relevant to the decrease of y_2. Excluding the effect of accumulation boards, T should be 4.876. Therefore it is easy to conclude that the effect of short-boards accumulation plays an important role in trustworthiness measurement. Specifically, when there exist multiple rules with a minimal value, all these rules affect T as a whole and decrease the trustworthiness score, that is, the buckets effect becomes more obvious when there exist more short boards.

Furthermore, the basic model and the model after improvement 1 are compared as shown in Table 6.

Table 6. Data Simulation for the Model Improvement 3

	y_1/α_1	y_2/α_2	y_3/α_3	ε/T_1	k/T_2
Group 1	8/0.3	8/0.5	4/0.2	0.4/5.298	2/5.298
Group 2	8/0.3	8/0.4	4/0.3	0.4/4.602	2/4.318

Comparing group 1 and group 2, we can see that in group 2, T_1 and T_2 both decrease, because the weight of y_2 decreases. However T_2 increases more than T_1, because the weight of minimal value increases in group 2. It is easy to conclude that in the model after improvement 1, the larger the weight of the minimal value is, the more obvious the buckets effect is.

5.2.4 *Verification of the Model Considering Effect of Multiple Short Boards*

Data from different groups are compared as shown in Table 7.

Table 7. Data Simulation for the Model Improvement 4

	y_1/α_1	y_2/α_2	y_3/α_3	k	b/T
Group 1	8/0.5	4/0.3	2/0.2	1	3/4.154
Group 2	8/0.5	6/0.3	2/0.2	1	3/4.574
Group 3	8/0.5	4/0.3	2/0.2	2	3/3.261
Group 4	8/0.5	4/0.3	2/0.2	1	4/4.154
Group 5	8/0.5	4/0.3	2/0.2	1	5/3.374

1) Comparing group 1 and group 2, when y_2 increases from 4 to 6, T increases. Thus T is monotonic increasing.

2) Comparing group 1 and group 3, when k increases, T degreases. It shows that the larger k is, the more obvious the buckets effect is.

3) Comparing group 1, group 4 and group 5, from 1, 4 to 5, threshold b increases successively. From group 1 to group 4, although threshold b increases, the amount of short-boards does not increase and T does not decrease. While in group 5, the increasing of threshold b brings the increasing of the short boards, so that T deceases clearly. For different groups with same data, along with the changing of threshold, the trustworthiness score might differ from each other, that is, when the threshold is predefined as a larger number, the expectation to the trustworthiness rules becomes higher,

and then the trustworthiness score we compute in our metric model trends to be lower.

5.2.5 Evaluation of Basic Standard Trustworthiness

Based on the above metric solution, we design the evaluation steps bellow.

Step 1. Based on the trustworthy attribute set in software trustworthiness problem, we partition the trustworthy rule set. The metric result depends on the satisfaction of these rules.

Step 2. Several different experts are invited to score each rule and measure the trustworthiness rules using fuzzy comprehensive evaluation (FCE).

Step 3. According to the metric model considering the effect of multiple short boards (4), it can obtain the value of basic standard trustworthiness $S\left(T_a\left(t\right)\right)$.

The FCE method mentioned above is based on experts' grading result, which is belonging to the indirect measurement method. It remains future work to find a direct measurement method to get the value of trustworthiness rule. Meanwhile, our metric model considering the effect of multiple short boards needs a predefined threshold. In practice, people usually have different expectations on different rules separately. Therefore, as a future development, we can consider setting up an independent threshold for each rule.

5.3 Case Study

Considering a financial software application, we can obtain $I\left(T_a\left(t\right)\right)$, $S\left(T_a\left(t\right)\right)$ and $A(T_a(t))$ respectively.

1) Ability Trustworthiness $A(T_a\left(t\right))$

Firstly, we need to determine the value of the weighted coefficient ρ according to the actual requirements of the software. Here, $\rho = 0.4$ for $A_f\left(T_a\left(t\right)\right)$ and $1-\rho = 0.6$ for $A_v\left(T_a\left(t\right)\right)$. And $S_f = 58$ and $S_v = 125$, thus $n = 58$. Using the least square method to obtain the parameters S_f and S_v and calculate $\bar{\lambda_f} = 1.119\,4$ and $\bar{\lambda_v} = 0.242\,4$. The mean time-loss rates of the failure and vulnerability are $\bar{x}_f = 2.701\,149$ and $\bar{x}_v = 16.689\,655$ respectively, and thus $\bar{x} = 2.701\,149$. According to the evaluation step 4, $R_f\left(\bar{x}\right) = 0.048\,622$ and $S_v\left(\bar{x}\right) = 0.519\,635$.

After a lot of experiments and practical research (large financial software), we get Table 8.

Therefore,

$$\begin{aligned} A(T_a(t)) &= \rho A_f(T_a(t)) + (1-\rho)A_v(T_a(t)) \\ &= 0.4 \times 10 + 0.6 \times 60 = 40 \text{ k\$}. \end{aligned}$$

2) Basic Standard Trustworthiness $S(T_a(t))$

Because the basic standard trustworthiness has its great industrial domain characteristics, it needs the experts to propose a comparison table (as shown in Table 9) between the basic standard trustworthiness level and its loss.

Table 8. Ability Trustworthiness Mapping Table Between Different Thresholds and Corresponding Loss

$R_f\left(\bar{x}\right)$	$A_f\left(T_a\left(t\right)\right)$ (k$)	$S_v\left(\bar{x}\right)$	$A_v\left(T_a\left(t\right)\right)$ (k$)
0.0∼0.1	10	0.0∼0.1	20
0.1∼0.3	10∼20	0.1∼0.3	20∼50
0.3∼0.5	20∼100	0.3∼0.5	50∼200
0.5∼0.8	100∼200	0.5∼0.8	200∼500
0.8∼1.0	>200	0.8∼1.0	>500

Table 9. Financial Software Comparison Table Between Basic Standard Trust Worthiness Level and Loss

Rule Value	Trustworthiness Level	Loss (k$)
0∼2	1	1 000
2∼4	2	500
4∼6	3	200
6∼8	4	50
8∼10	5	10

Through the evaluation steps, we obtain the rule value of 7.8. Therefore $I(T_a(t)) = 50$ k$.

3) Identity Trustworthiness $I(T_a(t))$

Table 10 shows the detecting results.

Table 10. Code Homology Detecting Results

T_i	β_i (%)	α_i (%)	ω_i (k$)
Software 1	20	60	100
Software 2	50	35	20
Software 3	30	10	30

According to the identify trustworthiness measure model, we can obtain that:

$$\begin{aligned} I(T_a(t)) &= \sum_{i=1}^{N} \omega_i \times \alpha_i \times \beta_i \\ &= 100 \times 20\% \times 60\% + 20 \times 50\% \times 35\% + \\ &\quad 30 \times 30\% \times 10\% = 16.4 \text{ k\$}. \end{aligned}$$

Therefore, the loss caused by the software identify trustworthiness is 16.4 thousand dollars.

4) Final Software Trustworthiness $GT(T_a(t))$

α, β and γ represent the corresponding weights, which can be obtained from the users and experts. Here, $\alpha = 0.1$, $\beta = 0.3$ and $\gamma = 0.6$.

$$\begin{aligned} GT(T_a(t)) &= \alpha I(T_a(t)) + \beta S(T_a(t)) + \gamma A(T_a(t)) \\ &= 0.1 \times 16.4 + 0.3 \times 50 + 0.6 \times 40 \\ &= 40.64 \text{ k\$}. \end{aligned}$$

If the value of $GT(T_a(t))$ is smaller, the trustworthiness of software will be higher. If it is bigger, the trustworthiness will be lower.

6 Conclusions

This paper described a measurable software trust framework based on humanities and sociology trust theory in order to facilitate trustworthiness evaluation with the same assessment indicator. Then it presents different metric models and methods for the three parts of software trustworthiness respectively. To summarize, it makes the following contributions.

• After strict reasoning, induction and certification, the S2S framework and its related concept "trustworthiness" prove to be a complete, compatible and independent trust system. And this framework is measurable and unified.

• It introduces a generalized indicator "loss" which unifies all parts of trustworthiness metric value, so that we can compare different software using the same indicator and standard.

• It proposes a complete metric solution for the three parts of software trustworthiness.

• It provides a case study to verify the solution applicable and effective.

• It is an innovation and the first time to study the "nature" and measurement of software trustworthiness based on the HSTM, which can open a new window for future research on the issue. Even so, future research needs to further prove the validity and completeness of the S2S framework, work out more and better measurement methods to form a complete and unified measurement system under the guidance of the S2S framework, and develop a widely acknowledged valid software trustworthiness measurement and application system.

References

[1] Nami M, Suryn W. Software trustworthiness: Past, present and future In *Proc. ISCTCS 2012*, May 28-June 2, 2012, pp.1-12.

[2] Tang Y X, Liu Z L. Progress in software trustworthiness metrics models. *Computer Engineering and Applications*, 2010, 46(27): 12-16. (in Chinese)

[3] Amoroso E, Taylor C, Watson J *et al.* A process-oriented methodology for assessing and improving software trustworthiness. In *Proc. the 2nd ACM Conference on Computer and Communications Security*, Nov. 1994, pp.39-50.

[4] Zhang H P, Shu F D, Yang Y *et al.* A fuzzy-based method for evaluating the trustworthiness of software processes. In *New Modeling Concepts for Today's Software Processes*, Münch J, Yang Y, Schafer W (eds.), Springer Berlin Heidelberg, 2010, pp.297-308.

[5] Nazila Gol Mohammadi, Sachar Paulus, Mohamed Bishr, *et al.* Trustworthiness attributes and metrics for engineering trusted Internet-based software systems. In *Cloud Computing and Services Science*, Helfert M, Desprez F, Ferguson D, Leymann F (eds.), Springer International Publishing, 2014, pp.19-35.

[6] Zhang X, Li W, Zheng Z M, *et al.* Optimized statistical analysis of software trustworthiness attributes. *Science China Information Sciences*, 2012, 55(11): 2508-2520. (in Chinese)

[7] Meyer B. The grand challenge of trusted components. In *Proc. the 25th International Conference on Software Engineering*, May 2003, pp.660-667.

[8] Yang F Q. Thinking on the development of software engineering technology. *Journal of Software*, 2005, 16(1): 1-7.

[9] Jelinski Z, Moranda P B. Software reliability research. In *Statistical Computer Performance Evaluation*, Greiberger W (ed), Academic Press, 1972, pp.465-484.

[10] Yang X, Gul Jabeen, Luo P. The concept model and definition system of software trustworthiness based on trust theory of humanities and sociology, 2016. http://c.eemet.cn/trustworthiness.pdf, Jan. 2018.

[11] Huang Y F, Liu Y Z, Luo P. SSRGM: Software strong reliability growth model based on failure loss. In *Proc. the 5th International Symposium on PAAP*, Dec. 2012, pp.255-261.

[12] Fred B. Schneider, Editor. Trust in Cyberspace. Washington, DC: National Academy Press, 1998.

[13] Becker S, Hasselbring W, Paul M *et al.* Trustworthy software systems: A discussion of basic concepts and terminology. *SIGSOFT Softw. Eng. Notes*, 2006, 31(6): 1-18.

[14] National Institute of Standards and Technology, U.S. Department of Commerce. Toward a preliminary framework for assessing the trustworthiness of software. IST Interagency Report 7755, Sept. 2010. http://ws680.nist.gov/publication/get_pdf.cfm?pub_id=906717, Jan. 2018.

[15] ISO/IEC 14598-1. Information technology-software product evaluation-Part 1: General overview. Published standard. ISO/IEC, 1999.

[16] Shooman M L. Probabilistic models for software reliabilty prediction In *Statistical Computer Performance Evaluation*, Freiberger W (ed.), Academic Press, New York, June 1972, pp.485-502.

[17] Wang H M, Tang Y B, Yin G *et al.* Trustworthiness of Internet-based software. *Science in China Series F: Information Sciences*, 2006, 49(10): 1156-1169. (in Chinese)

[18] Ding X L, Wang H M, Wang Y Y *et al.* Verification oriented trustworthiness evidence and trustworthiness evaluation of software. *Journal of Frontiers of Computer Science and Technology*, 2010, 4(1): 46-53.

[19] Shen G H, Huang Z Q, Qian J *et al.* Research on software trustworthiness evaluation model and its implementation. *Journal of Frontiers of Computer Science and Technology*, 2011, 5(6): 553-561.

[20] Liu Y Z, Zhang L, Luo P, Yao Y. Research of trustworthy software system in the network. In *Proc. the International Symposium on Parallel Architectures, Algorithms and Programming*, Dec. 2012, pp.17-20.

[21] Geng J K, Ye D R, Luo P. Forecasting severity of software vulnerability using grey model GM. In *Proc. the IEEE Advanced Information Technology, Electronic and Automation Control Conference* (*IAEAC*), Dec. 2015, pp.344-348.

[22] Yang X, Luo P, Jabeen G. A measurable SocialToTech software trust framework based on cost-loss. In *Proc. the 10th AEARU Workshop on Computer Science and Web Technology*, Mar. 2015, pp.63-65.

[23] Colquitt J A, Scott B A, LePine J A. Trust, trustworthiness, and trust propensity: A meta-analytic test of their unique relationships with risk taking and job performance. *Journal of Applied Psychology*, 2007, 92(4): 909-927.

[24] Mayer R C, Davis J H, Schoorman F D. An integrative model of organizational trust. *Acad. Manag. Rev.*, 1995, 20(3): 709734.

[25] Mcknight D H, Carter M, Thatcher J *et al.* Trust in a specific technology: An investigation of its components and measures. *ACM Trans. Manage. Inf. Syst.*, 2011 2(2): 1-25.

[26] Mcknight D H, Chervany N L. What trust means in e-commerce customer relationships: An interdisciplinary conceptual typology. *Int. J. Electron. Commerce*, 2001, 6(2): 3559.

[27] Zhang W, Liu W, Wu X. Quantitative evaluation across software development life cycle based on evidence theory. In *Proc. the 9th International Conference on Intelligent Computing Theories and Technology*, Jul. 2013, pp.353-362.

[28] Yan G, Xu F, Yao Y *et al.* Enhancing trustworthiness evaluation in internetware with similarity and non-negative constraints. In *Proc. the 5th Asia-Pacific Symposium on Internetware*, Oct. 2013.

[29] Zhu M L, Jin Z. Approach for evaluating the trustworthiness of service agent. *Journal of Software*, 2011, 22(11): 2593-2609. (in Chinese)

[30] Si G N, Ren Y H, Xu J *et al.* A dependability evaluation model for Internetware based on Bayesian network. *Journal of Computer Research and Development*, 2012, 49(5): 1028-1038. (in Chinese)

[31] Goel A L, Okumoto K. Time-dependent error detection rate model for software reliability and other performance measures. *IEEE Transactions on Reliability*, 1979, 28(3): 206-211.

[32] Musa J D Okumoto K. A logarithmic Poisson execution time model for software reliability measurement. In *Proc. the 7th Int. Conf. Software Engineering*, Mar. 1984, pp.230-238.

[33] Shick G J, Wolverton R W. An analysis of competing software reliability models. *IEEE Transactions on Software Engineering*, 1978, 4(2): 104-120.

[34] Torrado N, Wiper M P, Lillo R E. Software reliability modeling with software metrics data via Gaussian processes. *IEEE Transactions on Software Engineering*, 2013 39(8): 1179-1186.

[35] Alhazmi O H, Malaiya Y K. Quantitative vulnerability assessment of systems software. In *Proc. Reliability and Maintainability Symposium*. Jan. 2005, pp. 615620.

[36] Anderson R. Security in open versus closed systems — The dance of Boltzmann, Coase and Moore. In *Proc. the Conf. Open Source Software Economics*, Jul. 2002, pp.1-15.

[37] Musa J D. Software reliability data. Technique Report, Data and Analysis Center for Software, Rome Air Development Center, Rome, 1979, pp.9-10.

[38] Rescorla E. Is fining security holes a good idea? *IEEE Security & Privacy*, 2005: 3(1): 14-19.

[39] Zeng D F. Software trustworthiness evidence assessment framework research based on life cycle [Master Thesis]. Beijing: The Key Laboratory for Information System Security, Software School, Tsinghua University, 2011. (in Chinese)

[40] Lang B, Liu X D, Wang H M *et al.* A classification model for software trustworthiness. *Journal of Frontiers of Computer Science and Technology*, 2010, 4(3): 231-239.

[41] Lu G, Wang H M, Mao X G. A cognitive-based evidence model for software trustworthiness evaluation. *Journal of Nanjing University* (*Natural Sciences*), 2010, 46(4): 456-463.

[42] Immonen A, Palviainen M. Trustworthiness evaluation and testing of open source components. In *Proc. the 7th International Conference on Quality Software*, Oct. 2007, pp.316-321.

[43] ISO/IEC FDIS 9126-1:2001. Information technology — Software product quality — Part 1: Quality Model. Published Standard. ISO/IEC, June 2001.

[44] Tao H W, Chen Y X. A metric model for trustworthiness of softwares. In *Proc. 2009 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, Sept. 2009, pp.69-72.

[45] Tao H W, Chen Y X. Another metric model for trustworthiness of softwares based on partition. In *Quantitative Logic and Soft Computing*, Cao B Y, Wang G J, Chen S L *et al.* (eds.), Springer, 2010, pp.695-705.

[46] Tao H W, Chen Y X. A new metric model for trustworthiness of softwares. *Telecommunication Systems*, 2012, 51(2): 95-105.

[47] Kamiya T, Kusumoto S, Inoue K. CCFinder: A multilinguistic token-based code clone detection system for large scale source code. *TSE*, 2002, 28(7): 654–670.

[48] Li Z, Lu Z, Myagmar S, Zhou Y. CP-Miner: A tool for finding copy-paste and related bugs in operating system code. In *Proc. OSDI*, Dec. 2004, pp.289–302.

[49] Lin C. Research on code clone detection system based on fingerprint [Master Thesis]. Beijing: The Key Laboratory for Information System Security, Software School, Tsinghua University, 2016. (in Chinese)

[50] Li J J. Research on code clone detection system based on fingerprint [Master Thesis]. Beijing: The Key Laboratory for Information System Security, Software School, Tsinghua University, 2015. (in Chinese)

Xi Yang is a Ph.D. candidate of Tsinghua University, Beijing, and an associate professor of Fuzhou University, Fuzhou. Yang received her Bachelor's degree in information science at University of Shanghai for Science and Technology, Shanghai. She obtained her M.S. degree at Yunnan University, Kunming. She is a member of CCF and ACM. Her research interests include software trustworthiness, information security and software system theory.

Gul Jabeen is a Ph.D. candidate of Tsinghua University, Beijing. Jabeen received her Bachelor's degree from Karakoram International University, Pakistan. After completing the undergraduate degree she got her Master's degree from International Islamic University, Islamabad, in computer science. Her research interests include software and information security, software reliability and vulnerability prediction modeling.

Ping Luo is a professor of applied mathematics at the Key Laboratory for Information System Security and the School of Software, Tshinghua University, Beijing. He received his Ph.D. degree at Institute of Systems Science, Chinese Academy of Sciences, Beijing, majored in applied mathematics. In 1998, he completed his postdoctoral program at Institute of Mathematics of Chinese Academy of Sciences, Beijing. His research interests include information security, applied mathematics and software system theory.

Xiao-Ling Zhu is a lecturer at Hefei University of Technology, Hefei. Zhu received her Ph.D. degree in computer application technology of Hefei University of Technology in 2013. Her research interests include software trustworthiness and privacy protection.

Mei-Hua Liu is an associate professor of applied linguistics at the Department of Foreign Languages and Literatures, Tsinghua University, Beijing. Her research interests mainly include English as a Foreign Language (EFL) teaching and learning in the Chinese context, reticence and anxiety, EFL writing, and international education.