A Survey of Software Trustworthiness Measurement Validation

Hongwei Tao\textsuperscript{a} and Yixiang Chen\textsuperscript{b,*}

\textsuperscript{a}School of Computer and Communication Engineering, Zhengzhou University of Light Industry, Zhengzhou, 450002, China
\textsuperscript{b}MoE Engineering Center for Software/Hardware Co-Design Technology and Application, East China Normal University, Shanghai, 200062, China

Abstract

The software trustworthiness measurement is an essential research subject in trustworthy software. Software trustworthiness measurement validation can show whether the measurement is adequate for measuring the software trustworthiness. There are many research results in software trustworthiness measurement validation. In this paper, we survey software trustworthiness measurement theoretical validation and empirical validation. The research states of software trustworthiness measurement theoretical validation are summarized from the view of validation based on measurement theory and validation based on axiomatic approaches, and state-of-the-art empirical validation methods are studied through case studies, surveys, and experiments. Lastly, we analyze the challenges faced in software trustworthiness measurement validation.

Keywords: software trustworthiness measurement validation; theoretical validation; empirical validation

(Submitted on May 30, 2018; Revised on July 27, 2018; Accepted on August 8, 2018)

© 2018 Totem Publisher, Inc. All rights reserved.

1. Introduction

Software trustworthiness reflects the ability of software behaviors and results to meet users’ expectations. Its measurement is one of the core scientific problems in the research of trustworthy software [1]. Measurement models are reasonable and effective only if they have been validated to accurately characterize the attributes to be measured. Otherwise, there are no guarantees that the decisions made based on these measurement models will achieve the desired effects. Therefore, the validation of the measurement models is very important. In a sense, people verify the correctness of the measurement models based on validation. Software trustworthiness measurement is an important software measurement, and its validation is also very necessary.

The measurement validations in the field of software engineering include theoretical validation and empirical validation. Theoretical validation methods can be further divided into two categories [2]: one is based on the measurement theory [3-6], and the other is based on axiomatic approaches [7-9]. If a measurement model satisfies the homomorphism between the empirical relational system and the numerical relational system, it is considered to have passed the measurement theory validation [2-5]. The axiomatic approaches can formally describe the empirical understandings of software attributes by defining the expected properties of the numerical relation system in software measurements, which can provide a formal basis for the quantification of software attributes. If a measurement model satisfies these expected properties, it is considered to have passed the axiomatic approaches validation [5-6]. The main methods of empirical validation are case studies, surveys, and experiments [10-11]. Case studies are conducted on single or multiple entities or phenomena in real environments. They involve comparisons of results produced by multiple methods or tools [12]. There are usually three strategies for selecting a comparison object: a sister project, a baseline, or a random selection [10]. Surveys are retrospective studies. The main means of collecting data in surveys are interviews or questionnaires [13]. Experiments in software engineering are empirical methods that identify and manipulate key factors in experimental scenarios [11]. The key difference between case studies and experiments is the degree of control over the environment [14]. When experimental studies are used, they are intended to perform operations in different scenarios to find out the differences between them. However, the scenarios in case studies are controlled by the actual projects.

* Corresponding author.

E-mail address: yxchen@sei.ecnu.edu.cn
At present, the review of the study of software trustworthiness measurement validation has not been seen yet. In this paper, software trustworthiness measurement validation is discussed based on the various validation methods mentioned above.

2. Theoretical Validation of Software Trustworthiness Measurement

2.1. Validation based on Measurement Theory

There are two basic problems in measurement theory. One is the representation theory problem, which explains the necessary and sufficient conditions for the existence of homomorphic mapping from an empirical relational system to a given numerical relational system. The other is the uniqueness problem, which describes what scale type can be given for a given metric system [4-5, 15-16]. The validation based on the axiomatic approaches does not address the scale types, which is often criticized by researchers [3]. The scale types are very important for the correct application of statistical operations; for example, it makes no sense to apply comparisons of arithmetic to ordinal type. There are two kinds of measurement structures commonly used in the field of software measurement: proximity structure and extensive structure. The proximity structure is composed of a measurement object set and a binary relation on the measurement object set. The binary relation must satisfy the completeness, transitivity, positive, positivity, minimality, and symmetry [15]. This structure is mostly used for the measurement of internal attributes, such as software complexity [4-5], and the representative scholars are Geert Poels and Guido Dedene. The extensive structure highlights the operations between elements and consists of a measurement object set, a binary relation on the measurement object set, and a closed binary operation on the measurement object set. The binary relation must satisfy the completeness and transitivity, weak associativity, monotonicity, and Archimedian. This structure is also mainly applied to measure internal attributes, such as software complexity [6], and the representative scholars are Horst Zuse and Peter Bollmann.

In the field of software trustworthiness measurement, it was found that software trustworthiness measurement models based on extensive structure have been established only in [17]. We have not yet seen the publication of the proximity structure used to measure software trustworthiness. First, an extensive structure about software trustworthiness is constructed in [17]. Then, an additive software trustworthiness measurement model based on this extensive structure and a non-additive software trustworthiness measurement model based on this extensive structure are established respectively. It is also proven that the non-additive measurement model can be used as a ratio scale. Finally, the developed models are validated by use of axiomatic approaches [17].

2.2. Validation based on Axiomatic Approaches

Validation based on axiomatic approaches, also known as property-based validation, is mostly used in validations of internal attributes measurements, such as software complexity, cohesion, and coupling [7-9]. The properties of classical measurements related to internal attributes measurements include Briand’s properties [7], Kitchenham’s properties [8], and Weyuker’s properties [9].

In the field of software trustworthiness measurement, the team led by Prof. Y. X. Chen applies the axiomatic approaches to validate attribute-based software trustworthiness measurements. Four desirable properties of the attribute-based software trustworthiness measurement are first proposed in [18], that is, monotonicity, acceleration, sensitivity, and substitutivity. Reference [19] extends the above property set and introduces stability. This property describes that the software trustworthiness value must be greater than or equal to the minimum trustworthy attribute value and less than or equal to the maximum trustworthy attribute value. Reference [20] improves the stability and puts forward the expectability, which is related to the human subjectivity. Expectability states that if all the trustworthy attributes meet the users’ expectations, the final trustworthiness will also satisfy users’ expectations. Reference [21] further improves the above property set and adds three new properties: non-negativity, nullability, and appropriateness of the ratio of trustworthy attributes. Non-negativity requires that software trustworthiness is non-negative. Nullability asserts that software trustworthiness is the result of the combination of trustworthy attributes and cannot be obtained from a single trustworthy attribute. Appropriateness of the ratio of trustworthy attributes means that each trustworthy attribute should have an appropriate proportion assumption. Reference [22] extends the above works to measure software trustworthiness in the view of the decompositions of trustworthy attributes by use of axiomatic approaches, and it also proposes the desirable measure properties from the point of decomposition of trustworthy attributes. Reference [17] further extends the above works and establishes a set of properties that the module-based software trustworthiness measurement models should satisfy. These above properties have been applied successively to the theoretical validation of the software trustworthiness measurement models constructed in [17-24].
3. Empirical Validation of Software Trustworthiness Measurement

There are three types of software trustworthiness measurements: attribute-based software trustworthiness measurements, behavior-based software trustworthiness measurements, and process-based software trustworthiness measurements. The first two categories measure the software trustworthiness based on the analysis results of software attributes or software behaviors. The last category analyzes whether the software development process follows a series of guarantees related to the software trustworthiness or data collected in the software process to evaluate the software trustworthiness. The empirical validation of these three types of measurements involves case studies, surveys, and experiments. However, according to the differences in methods and techniques used in the measurements, there are differences in empirical validation methods.

3.1. Case Study

For the most part, examples are given to illustrate how to use the newly proposed model to measure software trustworthiness in the case studies of software trustworthiness measurement. There are also individual references involving comparisons of similar trustworthiness measurement models, but the baseline for the software trustworthiness measurement has not been established yet. The following is an overview of the case studies for software trustworthiness measurements based on attributes, behaviors, and processes.

The general framework of the attribute-based software trustworthiness measurement is as follows. The attributes/indexes that affect the software trustworthiness are firstly selected, which are described as trustworthy attributes/indexes. Then, measurement models are established to measure these selected trustworthy attributes/evidence. Finally, models and methods are proposed to measure software trustworthiness based on measurement results of trustworthy attributes/indexes. Typical models and methods include uncertainty theory [25-34], axiomatic approaches [18-24], defect analysis and test review [35] and so on.

The uncertainty theory commonly used in software trustworthiness measurements includes fuzzy set theory, rough set theory, analytic hierarchy process, grey number system theory, TOPSIS, and evidence theory. Reference [36] proposes a model of using a fuzzy comprehensive evaluation method to measure the software trustworthiness. The index set is established based on existing research. The index weights are obtained through expert experience. A case of aerospace software is given to illustrate how to use the proposed model to measure the software trustworthiness. Reference [31] also uses the fuzzy comprehensive evaluation method to measure the software trustworthiness, but its index set is determined by inviting experts. The index weights are calculated through information entropy to overcome the subjectivity in the determination of index weights. This model is applied to measure an e-commerce software trustworthiness to demonstrate its usability. Reference [29] also measures software trustworthiness based on the fuzzy comprehensive evaluation method. The index set is given by the author in advance. The index weights are obtained by the combination of objective weights and subjective weights, where the objective weights are calculated by the rough set theory and the subjective weights are obtained by means of expert opinion. Finally, the proposed method is applied to measure an aerospace software trustworthiness, and it is compared with the software trustworthiness measurement model based on fuzzy comprehensive evaluation method whose index weights are obtained by the questionnaires. The comparison results show that the model given in [29] can effectively avoid the subjectivity in the weight coefficients. The software trustworthiness measurement method proposed in [32] also involves the combination weights determined by the objective and subjective weights. The objective weights are calculated by the information entropy and the subjective weights are obtained by using the fuzzy analytic hierarchy process based on expert opinions. With the help of this combined weights, the improved TOPSIS method is used to measure the software trustworthiness. This method is applied to select the product lifecycle management software based on the software trustworthiness. Reference [27] combines the grey number system theory, Minkowski distance, and fuzzy clustering to measure software trustworthiness. It uses this method to perform ten rounds of trustworthiness measurement for a room adaptive temperature control software. The evaluation results are compared with the results of ten rounds of trustworthiness assessment for the same software with the method given in [28]. If the average absolute deviation is used as an indicator of the accuracy of the assessment, the comparison results show that the method presented in [27] is more accurate. Evidence theory has the ability to uniformly model the exact data and subjective judgments. It is also widely used in software trustworthiness measurement. Yang et al. propose a software trustworthiness measurement method based on utility theory and evidence theory [33] and take a metal liquid detection system as the research object to validate the effectiveness of the proposed method. They further study the application of evidence theory and utility theory in the software trustworthiness measurement. Based on the model given in [33], a discount factor is introduced to solve the unreliability problem of evaluation [26]. The metal liquid detection system mentioned in [33] is taken as a case to illustrate the effectiveness of the measurement method. In order to solve the problem of the software trustworthiness evaluation with uncertain qualitative and quantitative trustworthy indicators, Yang et al. calculate the indicator weights used in evidence reasoning based on uncertain measures [25]. A team composed of commercial hospital information system (HIS)
developers, software engineering experts, and selected testers use the proposed method to evaluate the HIS trustworthiness to validate the rationality of the model. Shen et al. establish a generic software trustworthiness assessment model consisting of an evidence model, attribute model, and index system [30]. The feasibility of the model is demonstrated by taking the tax-oriented software trustworthiness assessment as an example.

Chen et al. establish the trustworthiness measurement method for aerospace embedded software based on the axiomatic approaches [24]. The software special review expert group composed of ten experts applies the method to evaluate 23 aerospace embedded software’ trustworthiness. Reference [24] selects 11 typical software among the 23 software to make an analysis. The analysis results show that this method can effectively evaluate the trustworthiness of the aerospace embedded software and identify the areas that need to be improved. Li et al. focus on five aspects of trustworthiness, that is, software correctness, reliability, fault-tolerance, security, and privacy. Then, they assess software security, privacy, and fault tolerance based on weakness analyses, and they evaluate software correctness and reliability based on testing assessment [19]. Reference [35] shows the validity and rationality of the assessment method through a detailed analysis of an open source project MeyboJmail.

The general framework of the behavior-based software trustworthiness measurement model is as follows. First, the source codes are analyzed or the running statuses of the software are dynamically detected under the trustworthy environment, software behavior characteristics are acquired, and the software normal behavior model is established. Then, at the checkpoint (also known as the monitoring point), the actual operating statuses of the software are monitored, their actual behavior characteristics are extracted, and the software actual behavior model is built. Finally, the actual behavior is compared with the normal behavior to evaluate software trustworthiness. The focuses of this kind of measurement are behavioral modeling and the comparative study of normal behaviors and actual behaviors. The commonly used behavioral modeling methods include system call sequence, finite automata, concurrency theory, and program slicing. Typical comparison methods include rule calculation, abnormal value threshold comparison, risk assessment, and quantitative bisimulation.

Tian et al. introduce software running trajectories and function trajectories to describe the software behavior. The software running trajectories describe execution paths of the software, and the function trajectories depict functions performed by the software [37]. The attribute information involved in these trajectories are collected by implanting sensors at checkpoints. Reference [37] conducts an experiment on the editor software VI 16.1 with TOCTTOU (time of check to time of use) vulnerability under the Red Hat 9 Linux operating system. First, the software trustworthy behaviors are obtained from static analysis and dynamic analysis at checkpoints. When the software is running, attacks against the above vulnerabilities can be successfully detected by comparing with a given software abnormal value threshold [37]. Tian et al. also classify the attributes of software monitoring point into the control flow level attributes, access control level attributes, and scene level attributes, and they propose a behavior-based software trustworthiness measurement model for software monitoring point in the view of the classification attributes [38]. Reference [38] also carries out experiments on the editor software VI 6.1 under the Red Hat 9 Linux operating system. The access control level attributes and scene level attributes are used as examples for analysis. The experimental results show that the model can accurately evaluate the trustworthiness of monitoring points. Reference [39] establishes a software dynamic trustworthiness measurement model from the point of the checkpoint’s classification attributes. Experiments are carried out on the vulnerable software KON 20.3.9 and VI 6.1 under the Red Hat 9 Linux operating system. Attacks on the control flow level, access control level, and scene level are designed respectively. The model can detect attacks on the above three levels. Moreover, compared with the method proposed in [40-41], it can detect more types of attacks. Reference [42] also uses software running trajectories and function trajectories to describe the software behavior, but it evaluates the software behavior trustworthiness based on the identification evaluation rules and scene evaluation rules of checkpoints. In order to verify the accuracy and efficiency of the model, experiments are performed on the software GZIP, EJECT, and PS under the Linux operating system [42]. Experimental results show that the model presented in [42] is superior to the Dyck model given in [43] in terms of both accuracy and efficiency. In order to verify the validity of this model, WU-FTPD-2.6.0 with vulnerabilities in the Linux operating system is selected as the target software and these vulnerabilities are attacked. The experimental results indicate that this model can detect both control flow and data flow attacks [42]. Reference [44] further considers the suspected risk at the checkpoint and calculates software behavior trustworthiness through a reward and punishment mechanism. Simulation experiments are performed under the Linux operating system. The experimental method is similar to that in [42], but the difference is that five cycles of normal training are carried out and the checkpoint risk values and software behavior trustworthiness value are calculated by processing the information collected during the training period. Reference [34] tries to describe the software behavior with behavior trajectories and checkpoint scene. The rationality and effectiveness of the measurement model are illustrated by taking four common software GZIP, CAT, FIND, and TAR under the Linux operating system as examples. The above models only calculate the trustworthiness of the software running at one time. Reference [37] studies software and its modules trustworthiness measurements after running for a period of time. The Markov model
and the weights of checkpoints are used to reflect the trustworthy statuses of the checkpoints, and finally the trustworthy statuses of the checkpoints are synthesized to measure the overall trustworthiness of the software [45]. In order to verify the rationality and validity of the model, experiments are performed with GZIP as the target software in the Linux operating system. The results show that compared with the evaluation model based on the checkpoint proposed in [24], the data obtained from this model are less affected by the pre-set parameters. For the abnormal situation, the typical vulnerabilities of GZIP are attacked, and GZIP is run many times to collect scene information at each checkpoint at the same time. Then, checkpoints and software trustworthiness are also computed according to the proposed model, but this time, the corresponding checkpoints and software trustworthiness become very low. Therefore, the model can accurately discover the software’s vulnerability.

Mitra Nami and Witold Suryn propose a software trustworthiness measurement model based on scenarios and finite state machines [1, 46]. First, interaction scenarios between a software and its users and the environments are extracted according to users’ requirements, and a finite state machine that is used as an expected behavior is generated. Then, according to the actual operation scenario, a finite state machine for actual behavior is established. Finally, the software trustworthiness is evaluated according to the consistency of the two behaviors. In order to illustrate the availability of the model, it is used to evaluate the trustworthiness of an education center online course registration system [1, 46].

Dr. F. Zhang et al. take system calls as a modeling object and use the action of Calculus of Communication System to describe software dynamic behavior. Based on the non-interference model, a software dynamic behavior trustworthiness analysis method is proposed. An example is given to illustrate the working process of the proposed software dynamic behavior modeling and trustworthiness measurement [47]. Wang et al. decompose the software running behavior into parallel or series action sequences based on Pi calculus [48]. Then, the software trustworthiness is measured according to the fitting degree between the real running results and the expected results. The effectiveness of the measurement process is illustrated with the MWB software. Zhou et al. combine Q algebra with the ability to describe multi-dimensional trustworthy features and Pi calculus, and they propose QPi. The similarity of the QPi process with respect to trustworthy features is studied by means of quantitative bisimulation, and the switching bit protocol is modeled by QPi to illustrate its descriptive ability [49].

Wang et al. apply program slicing technology to study software interaction behavior trustworthiness. The software interaction behavior expected model is established by extracting program slices that rely on input data. The actual behavior of the software is dynamically acquired by monitoring the running conditions of the software, and the comparison results of the two behaviors are taken as the trustworthiness measurement value [50]. A program with heap buffer overflow under the Linux operating system is taken as a test example. The test results show that the proposed model can find both the attacks on the control variables and the attacks on non-control variables [50].

According to different evaluation methods and categories of evidence, software process trustworthiness evaluation can be divided into evaluation based on expert experience and evaluation based on process data [51].

Software process trustworthiness evaluation based on expert experience measures the software trustworthiness by analyzing whether the whole life cycle of software follows the trustworthy principles. Amoroso et al. put forward TSM (Trusted Software Methodology) from the perspective of the software development process [52]. TSM first gives a series of trusted principles that the software development process should follow. They are mainly well-known and widely accepted security principles and software development principles. Then, the software trustworthiness is measured by determining whether the software development method is consistent with these trusted principles. The development process of the V/MLS operating system, the development process of the STAT prototype tool of Bell Labs, and the re-development process of the Advanced Interactive Monitor software system written in Ada language are taken as examples to demonstrate the rationality of the measurement method [52-53].

The software process trustworthiness evaluation method based on process data uses the collected data that describes the behaviors and attributes of the process as evidence to measure software trustworthiness. Wang et al. make use of the fuzzy set theory and analytic hierarchy process techniques to measure software processes trustworthiness on the basis of their Trustworthy Process Management Framework (TPMF). This method is applied to the review and testing process trustworthiness evaluation of the eight versions of SofTPM [54]. Moreover, Wang et al. take TSM and evidence theory as the foundation and use the trustworthy process data as evidence to evaluate the software process trustworthiness [55]. Taking a requirement management software as an example, reference [55] explains how to use the proposed method to evaluate the software process trustworthiness. At the same time, this method is compared with the existing software process trustworthiness evaluation method. The comparison results show that it not only avoids the deviation of the evaluation results caused by subjective judgment in the existing evaluation methods, but also solves the problem of the non-
comprehensiveness of the evaluation results caused by a single judgment [55]. A software process trustworthiness measurement index system composed of software process products, software process activities, and software entities is established in [56], and the corresponding measurement models are built for each index from different perspectives. The development process of an insurance business integrated management information system is used as an example to illustrate the usability of the measurement model [56]. The software process trustworthiness measurement index system established in [57] is composed of schedule trustworthiness and cost trustworthiness. The effectiveness of the proposed method is analyzed by taking the software analysis process of a software company in Shandong Province as the research object [57].

3.2. Survey

The survey methods used in the software trustworthiness measurement include interviews, expert consultations, questionnaires, and literature reviews. The primary purposes of the first three methods are to obtain the candidate software to be measured, the set of measure indexes, the weights of the indexes, and other data that are useful for the software trustworthiness measurement. For example, in the references [29, 32] cited in the previous case studies, there are cases where candidate software, the set of measure indexes, and the weights of the indexes are obtained by expert consultations or questionnaires. The main purpose of the literature review is to summarize the existing trustworthiness measurement models, trustworthiness evaluation methods, or trustworthiness measurement tools [58-59], but the review of the validation of trustworthiness measurement models, trustworthiness evaluation methods, or trustworthiness measurement tools has not yet been seen.

The following are two classic examples using the survey methods in the field of software trustworthiness measurement. Taibi et al. first obtain the factors that affect the open source software trustworthiness through literature reviews, interviews, and questionnaires. Then, with the help of the self-developed tool MacXim, they obtain the objective measurements of the complexity, cohesion, modularity, and so on of 22 open source software written in Java language and 22 open source software written in C/C++ language. The subjective evaluations of trustworthiness of the above-mentioned software are obtained through more than 500 questionnaires. Finally, a multivariate regression model is established to study the relation between subjective evaluations and objective measures [60]. Li et al. propose a software process model with a risk management module and a cost control module, and a process risk and software trustworthiness measurement model is established based on this model. According to a survey of 93 project managers in five software companies, the relevant information of 63 indicators involved in this measurement model are obtained [61]. Case simulations are conducted for a variety of software environments, and software process risks are analyzed in various software environments. The results show that risk management is very important to enhance software trustworthiness [61].

3.3. Experiments

Experiments need to identify and control the factors or variables in the research scenario. In this sense, many of the aforementioned cases of behavior-based software trustworthiness measurements can be understood as experiments. For example, there are measurements for the normal situation and measurements for the abnormal situation in the case studies of the measurement models mentioned in [21, 34, 37-39, 42, 44-45]. The process-based software trustworthiness measurement model proposed in [61] also involves case simulations for a variety of software environments and can be considered as an experimental study.

The following are two classic experiments of the attribute-based software trustworthiness measurement models that are both related to the Bayesian network. Considering that most of the current Internetware trustworthiness evaluations do not involve the system structure and the evaluation index is single, Xu et al. propose a trustworthiness evaluation model of Internetware based on Bayesian network. It can solve the problem mentioned above [62]. They conduct an experiment on a practical enterprise e-business application system to verify the rationality of the measurement model. The experiment involves five indexes: quality of test, normal programming, detailed document, skilled staff, and process management. According to the different degree of satisfaction of the evaluation indexes, the corresponding calculations are carried out. The experimental results show that this model can objectively and comprehensively evaluate the Internetware trustworthiness [62]. In view of the lack of dynamic adaptive capability of the current trustworthiness measurement model, Wu et al. propose a trustworthiness measurement model based on dynamic Bayesian network [63], which is related to time factor. Reference [63] uses Genie, a simulation modeling tool developed by the Decision Systems Laboratory of the University of Pittsburgh, to conduct two simulation experiments. The experimental process includes the extraction of variables, the determination of dependent relation between variables, and the determination of the conditional probability distribution among variables. The first simulation experiment verifies the validity of the model. The second simulation experiment shows the superiority of this model over the traditional Bayesian network model and the trustworthiness measurement model based on dynamic Bayesian network proposed in [64].
4. The Challenges in Validation of Software Trustworthiness Measurement

Software trustworthiness measurement validation has achieved some good research results, but there are still some problems that must be addressed.

4.1. Theoretical Validation

At present, the theoretical validation of software trustworthiness measurements is mainly focused on the attribute-based software trustworthiness measurements, and the research in the theoretical validation of software trustworthiness measurements based on behavior and process has not been launched yet. For the validation based on the measurement theory, only the software trustworthiness measurement models based on the extensive structure have been constructed, and the software trustworthiness measurement models based on the proximity structure or other measurement structures have not been established yet.

4.2. Empirical Validation

The case studies of empirical validation of software trustworthiness measurements mainly demonstrate the availability of measurement models. The researches of the comparison of similar software trustworthiness measurement models based on case studies are insufficient. Many cases are artificially constructed to verify the rationality of the models. Even if there are real cases, their scales are small and their representativeness are not obvious. Furthermore, the baselines for the software trustworthiness measurements based on attributes, behaviors, and processes have not been formed. Therefore, there are no benchmarks available for comparison with newly constructed models.

The survey methods of empirical validation of software trustworthiness measurements mainly obtain the candidate software to be measured, the set of measure indexes, and the weights of the indexes. There are few researches on the users’ subjective evaluation of software trustworthiness. Software trustworthiness is a subjective reflection of software quality in the users’ mind; therefore, it is very important to obtain users’ subjective evaluation of software trustworthiness. In addition, research on the rationality of the design of questionnaires about software trustworthiness is also insufficient.

Experimental researches of empirical validation of software trustworthiness measurements mostly concentrate on the control of a few variables or factors, and the experiments mainly focus on simulations scenarios or laboratory scenarios. There are few experiments on large-scale or online experiments.

5. Conclusions

In this paper we introduce the main research results of the theoretical validation and empirical validation of software trustworthiness measurements. We discuss the theoretical validation of software trustworthiness measurement from two aspects, namely, the validation based on measurement theory and the validation based on axiomatic approaches. The progress of empirical validation of software trustworthiness measurements is described from three aspects: case studies, surveys, and experimental studies. Finally, the challenges of the validation of software trustworthiness measurements are summarized.

Acknowledgements

This work was supported by the Innovation Group Project of National Natural Science Foundation (61321064), National Key R&D Plan Inter-Governmental S&T Cooperation Subproject (2016YFE0100600, 2016YFE0100300), Doctoral Research Fund of Zhengzhou University of Light Industry (13501050045), and Sino Foreign Cooperation Foundation of the Henan Provincial Science and Technology Department (162102410076).

References


