

Software Trustworthiness Evaluation using Structural Equation Modeling

Rumei Deng^{a,*}, Yixiang Chen^b, Hengyang Wu^a, and Hongwei Tao^c

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

^bShanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai, 200062, China

^cSchool of Computer and Communication Engineering, Zhengzhou University of Light Industry, Zhengzhou, 450002, China

Abstract

Software trustworthiness evaluation results can provide guidance for existing software development. Therefore, it is of great significance to study software trustworthiness evaluation. Weights and the relationship of trustworthy attributes have a huge effect on the results of software trustworthiness evaluation. However, few studies have considered weights and the quantitative relationship among trustworthy attributes at the same time. Structural equation modeling (SEM) provides a more scientific and reasonable method to obtain weights, where the objectification of subjective weights is guaranteed. In this paper, we propose a SEM-based method to evaluate software trustworthiness. Firstly, we establish the trustworthy evaluation indicator system for software. Based on the survey data, we construct the SEM for software trustworthiness evaluation to obtain the weights of trustworthy attributes and the relationship among them. Lastly, we apply the trustworthy measurement model to calculate the trustworthiness value of software to be surveyed. These trustworthiness values demonstrate the reasonability of our method.

Keywords: structural equation modeling (SEM); software trustworthiness; trustworthy attribute; weight

(Submitted on June 20, 2019; Revised on July 14, 2019; Accepted on August 6, 2019)

© 2019 Totem Publisher, Inc. All rights reserved.

1. Introduction

With the rapid development of the social electronic information industry, software is used in various fields, such as aerospace, rail transit, finance, and medicine. Although software development technology is mature, it can still cause errors during operation, such as when the input attributes are not checked. Serious incidents such as airplane crashes and train derailment may even endanger the safety of human lives and property. For example, the crash of the JAS39 fighter in Sweden was due to the failure of the software design [1]. Thus, people are paying more and more attention to software trustworthiness.

There are still many challenges in software trustworthiness evaluation. Most research on the relationship among attributes is based on experience from a qualitative point of view. The interaction between trustworthy attributes may have a significant influence on the trustworthiness of the whole software system. In the existing methods or models for software trustworthiness evaluation, the weights are mostly obtained through expert experience, information entropy, or rough set theory. Shi used a fuzzy comprehensive evaluation model based on entropy weight to measure software dependability [2]. Li established an improved comprehensive evaluation model of software dependability based on rough set theory [3]. Shi developed an approach for software trustworthiness by using combination weights and TOPSIS [4]. SEM is a confirmatory analysis method for analysing the relationship between variables. Compared with the classical methods, the advantages of SEM are that it: (1) allows for the existence of measurement errors, (2) considers the correlation and causality between variables, (3) analyses the rationality of the assumed model by path coefficients and gives revision suggestions, (4) lowers the subjectivity of the indicator weight, and (5) quantifies the relationship between variables. So far, there has been no research on applying SEM to software trustworthiness evaluation. Yu constructed an exclusive SEM to study the relationship between software process trustworthiness and software product trustworthiness [5]. Dai applied SEM to study

* Corresponding author.

E-mail address: 51174500080@stu.ecnu.edu.cn

the performance evaluation of sustainable supply chains [6].

Aiming at the trustworthiness evaluation of software, domestic and foreign scholars have carried out a series of research and obtained some achievements. Wang et al. proposed a software trustworthiness measurement model, where the weights of each trustworthy attribute are scored by experts [7]. The National Institute of Standards and Technology proposed a preliminary framework of credible evaluation for the quantification of software trustworthiness. The factors affecting software trustworthiness include reliability, availability, safety, and other software attributes [8]. Oldenburg provided a holistic view of software trustworthiness that considers system architecture, evaluation, and certification in different fields [9]. The software attributes include availability, safety, reliability, performance, and so on. Liu et al. proposed that trustworthiness is based on many concepts such as software correctness, reliability, safety, timeliness, completeness, and availability [10]. Tao divided the attributes that affect software trustworthiness into key attributes and non-key attributes [11]. A key attribute refers to an attribute that is necessarily required for trustworthy software, and the set of key attributes includes functionality, reliability, maintainability, and survivability. A non-key attribute refers to an attribute that is not a key attribute but also required according to the characteristics of software, such as predictability, timeliness, and real-time.

Recently, we have been working on a project to evaluate the trustworthiness of software for rail transit computer interlocking products. According to the project requirements, we choose four trustworthy attributes, including functionality, reliability, safety, and maintainability. Therefore, we mainly focus on these attributes in this paper.

Based on this background, we establish a trustworthiness evaluation indicator system for software. Then, we construct the SEM for the software trustworthiness evaluation to obtain the weights of trustworthy attributes and finally calculate the value of software trustworthiness.

2. Establishment of Software Trustworthiness Evaluation Indicator System

2.1. Principles for Establishing an Evaluation Indicator System

Since the factors considered in the evaluation of software trustworthiness are dynamic and many-sided, the scope of the evaluation indicators is wide and the content is complex. According to the complex system theory and management decision theory, the software trustworthiness evaluation indicator system should follow the principles of science, system, comparability, practicability, independence, and purpose [12].

2.2. Construction of the Trustworthy Evaluation Indicator System

We choose functionality, reliability, safety, and maintainability to evaluate the software trustworthiness. The evaluation indicator for each attribute refers to the quality model defined in ISO/IEC 9126 [13-14]. The specific trustworthy evaluation indicator system is shown in Table 1.

Table 1. Trustworthy evaluation indicator system

Evaluation target	Primary indicator	Secondary indicator	Symbol
Software trustworthiness	Functionality	The adequacy of function definition	X1
		The degree of adaptation of function definition	X2
		The correct degree of function definition	X3
		The integrity of definition of interface relationship, interface protocol, and interface data	X4
		Interface extensibility	X5
	Reliability	The adequacy of maturity requirement definition	X6
		The degree of demand stability	X7
		The identification of error handling rules	X8
		The clarity of failure and post-failure treatment	X9
	Safety	The adequacy of definition of general safety requirements and specific safety requirements	X10
		The adequacy of definition of safety function requirement	X11
		The adequacy of anti-hazard requirements definition	X12
		The clarity of hazard log	X13
	Maintainability	The clarity of two-way tracking relationship	X14
		The ability of problem analysis and positioning	X15
		The clarity of qualification review requirements	X16
		The ability to reconfirm change	X17

We assume that the indicators are independent. However, in the actual software system, functionality, reliability, safety, and maintainability are interactive. For example, for a rail transit software, safety and functionality are inter-linked in the sense that a weakness in either or a mismanagement of conflicts between safety and functionality requirements may prevent the achievement of a dependable system. The safety and functionality can only be achieved by meeting all reliability and maintainability requirements as well as controlling the ongoing, long-term maintenance and operational activities and the system environment [15].

3. Construction of SEM and Software Trustworthiness Evaluation

The structural equation model (SEM), also known as the latent variable model (LVM), is a confirmatory analysis method that can deal with measurement and analysis problems simultaneously. It attaches great importance to the use of multiple statistical indicators [16], which can analyse the relationship between multiple variables and the degree of consistency between the variables and the observed indicators. Software trustworthiness evaluation involves functionality, reliability, safety, and maintainability in this paper. There are intrinsic links between the four trustworthy attributes. Therefore, SEM is a more reasonable choice, and this method also allows for the existence of measurement errors. The model can also solve the problem that trustworthy attributes cannot be directly observed or measured as abstract concepts, nor can they be presented in terms of data quantification. It is advisable in the actual evaluation of software trustworthiness.

3.1. Model Construction

SEM consists of two parts: the structural model and the measurement model. The structural model describes the causal relationship between the latent variables. The measurement model reflects the relationship between the latent variables and the observed variables. The latent variable is a constructive factor, which cannot be measured or observed directly. The latent variables can be divided into exogenous latent variables and endogenous latent variables. Exogenous latent variables refer to variables in the model that are not affected by any other variables but directly affect other variables. Endogenous latent variables are variables that will be affected by either type of variable. Observed variables, also known as manifest variables or indicator variables, can be directly observed or measured, and the data obtained can be quantified [16]. In the figure below, functionality, reliability, safety, and maintainability are exogenous latent variables, and software trustworthiness is an endogenous latent variable. A single arrow indicates the causal relationship in one direction, with the starting point of the arrow as a dependent variable and the arrow pointing to the result variable. The double arrow indicates the relationship between the two variables, and SEM defaults to the double arrow between exogenous latent variables [16]. In measurement model, the higher the factor loading of each observed variable, the greater the intensity of the influence of the latent variable. For functionality, the observed variables are denoted by X1 to X5. The observed variables of reliability are represented by X6 to X9. The safety observation variables are denoted by X10 to X13. The observed variables of maintainability are represented by X14 to X17. For software trustworthiness, the user feedback of the software is regarded as an observed variable, including the satisfaction degree of the software function, the stability of the software running, and the friendly degree of the user interaction, which are represented by Y1, Y2, and Y3, respectively. e1 to e21 are error terms.

We use amos25.0 to draw the path diagram, and the SEM of software trustworthiness evaluation is shown in Figure 1:



Figure 1. SEM of software trustworthiness evaluation

3.2. Data Collection and Model Test

We collect 170 valid questionnaires, where the observation data of functionality, reliability, safety, and maintainability are derived from those involved in software development. The observed data of software trustworthiness comes from user feedback. For example, Table 2 shows the data surveyed for six of the 170 pieces of software.

Table 2. The survey data for six pieces of software

Observed variable Score Software number	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	X13	X14	X15	X16	X17	Y1	Y2	Y3
No79	9	10	10	10	9	9	9	9	8	9	9	9	10	9	10	8	9	9	8	10
No142	9	9	10	8	10	10	9	9	9	10	9	10	10	8	9	7	8	10	9	10
No116	8	7	9	8	10	8	9	7	7	9	8	9	10	10	9	8	7	9	9	8
No161	6	9	10	7	8	7	8	7	10	7	8	7	7	8	6	8	7	6	7	9
No11	7	6	6	6	6	6	7	7	7	7	5	6	6	6	5	4	6	6	6	6
No167	10	6	6	4	3	3	6	7	5	5	3	7	6	7	5	4	6	4	5	6

3.2.1. Reliability Test

Reliability refers to the degree of consistency of the results obtained from repeated measurements of the same object using the same method. Online SPSS analysis software is used to test the reliability. The reliability test usually takes Cronbach's alpha coefficient as the standard. Generally, the value of the α coefficient is between 0 and 1. If the α coefficient is not more than 0.6, it is generally considered that the internal consistency is insufficient. When it reaches 0.7-0.8, the reliability of the scale is quite good, and 0.8-0.9 indicates that the scale is very reliable. The results of the reliability analysis of the data in this paper are shown in Table 3. The reliability of each measurement indicator is greater than 0.8, which indicates that the reliability of the scale is very good.

Table 3. Reliability analysis table

Measurement indicator	Cronbach's alpha coefficient
Functionality	0.865
Reliability	0.896
Safety	0.882
Maintainability	0.906
Software trustworthiness	0.864

3.2.2. Validity Test

Validity refers to the extent to which a measuring tool or means can accurately measure what is required to be measured. We use amos25.0 for the validity test. Validity is divided into content validity, criterion validity, and construct validity. Content validity is mainly based on subjective judgment, and it is difficult to choose a suitable criterion for criterion validity. Therefore, the construct validity test is conducted in this paper. We calculate the average variance extracted (AVE) of the latent variable. If the value of AVE is more than 0.5, the observed variable can reflect its latent variable effectively. The validity analysis of the data can be found in Table 4.

All factor loading values are between 0.5 and 0.95. The AVE values of functionality, reliability, safety, and maintainability are all greater than 0.5, indicating that the scale has high construct validity. The fit indicators of the model are shown in Table 5.

CMIN/DF, IFI, TLI, and CFI all meet the standard, RMSEA is equal to the standard value, and NFI and RFI are more than 0.8 and close to 0.9, which is acceptable as a whole.

3.3. Software Trustworthiness Value Calculation

Based on the standardized result of the SEM of software trustworthiness evaluation in Figure 1, we normalize the factor

loading of each observed variable to obtain the proportion of the observed variable to the latent variable and calculate the values of the four trustworthy attributes. The same method is used to calculate the weights of the trustworthy attributes and evaluate the software trustworthiness.

Table 4. Validity analysis table

Evaluation indicator			Factor loading	AVE (average variance extracted)
Primary indicator	Secondary indicator	Symbol		
Functionality	The adequacy of function definition	X1	0.64	0.56
	The degree of adaptation of function definition	X2	0.70	
	The correct degree of function definition	X3	0.83	
	The integrity of definition of interface relationship, interface protocol, and interface data	X4	0.82	
	Interface extensibility	X5	0.76	
Reliability	The adequacy of maturity requirement definition	X6	0.79	0.64
	The degree of demand stability	X7	0.76	
	The identification of error handling rules	X8	0.82	
	The clarity of failure and post-failure treatment	X9	0.84	
Safety	The adequacy of definition of general safety requirements and specific safety requirements	X10	0.85	0.63
	The adequacy of definition of safety function requirement	X11	0.79	
	The adequacy of anti-hazard requirements definition	X12	0.85	
	The clarity of hazard log	X13	0.68	
Maintainability	The clarity of two-way tracking relationship	X14	0.85	0.69
	The ability of problem analysis and positioning	X15	0.83	
	The clarity of qualification review requirements	X16	0.85	
	The ability to reconfirm change	X17	0.80	

Table 5. Fit indicators of model

Indicator to be tested	CMIN/DF	NFI	RFI	IFI	TLI	CFI	RMSEA
Fit standard	<3	>0.9	>0.9	>0.9	>0.9	>0.9	<0.08
The data of test result	2.074	0.880	0.842	0.934	0.912	0.933	0.080

The formula for calculating the weight of X1 is as follows:

$$\text{The weight of } X1 = \frac{0.64}{0.64+0.70+0.83+0.82+0.76} = 0.17 \quad (1)$$

We use the software trustworthiness measurement model proposed by Wang [7] to calculate the values of functionality, reliability, safety, maintainability, and software trustworthiness. The model satisfies the Cannikin Law. In order to achieve a good result, the value of each item must reach a certain limit. The weights of observed variables are shown in Table 6.

Table 6. The weights of observed variables

Trustworthy attribute	Observed variable	Weight
Functionality	The adequacy of function definition	0.17
	The degree of adaptation of function definition	0.19
	The correct degree of function definition	0.22
	The integrity of definition of interface relationship, interface protocol, and interface data	0.22
	Interface extensibility	0.20
Reliability	The adequacy of maturity requirement definition	0.25
	The degree of demand stability	0.24
	The identification of error handling rules	0.25
	The clarity of failure and post-failure treatment	0.26
Safety	The adequacy of definition of general safety requirements and specific safety requirements	0.27
	The adequacy of definition of safety function requirement	0.25
	The adequacy of anti-hazard requirements definition	0.27
	The clarity of hazard log	0.21
Maintainability	The clarity of two-way tracking relationship	0.26
	The ability of problem analysis and positioning	0.24
	The clarity of qualification review requirements	0.26
	The ability to reconfirm change	0.24

$$\text{The value of functionality} = X1^{0.17} \times X2^{0.19} \times X3^{0.22} \times X4^{0.22} \times X5^{0.20} \quad (2)$$

The formulas for calculating the values of reliability, safety, and maintainability are similar to that of functionality.

The formula for calculating the software trustworthiness based on four trustworthy attributes (The weights of trustworthy attributes are shown in Table 7) is as follows:

Table 7. The weights of trustworthy attributes

Trustworthy attribute	Weight
Functionality	0.32
Reliability	0.05
Safety	0.41
Maintainability	0.22

$$\text{The value of software trustworthiness} = \frac{\text{the value of functionality}^{0.32} \times \text{the value of reliability}^{0.05} \times \text{the value of safety}^{0.41} \times \text{the value of maintainability}^{0.22}}{\quad} \quad (3)$$

According to the formula for calculating weight, the weights of Y1, Y2, and Y3 are 0.33, 0.33, and 0.34, respectively, and the formula for calculating the value of software trustworthiness based on Y1, Y2, and Y3 is as follows. It is only used to verify the validity of the software trustworthiness value based on the four attributes.

$$\text{The value of software trustworthiness} = Y1^{0.33} \times Y2^{0.33} \times Y3^{0.34} \quad (4)$$

We take the six pieces of software mentioned as an example. The survey data for the software are shown in Table 1.

According to the formula, the values of the four trustworthy attributes and software trustworthiness are shown in Table 8.

Table 8. The values of trustworthy attributes and software trustworthiness

Software number	Trustworthy attribute and software trustworthiness Value	Functionality	Reliability	Safety	Maintainability	Software trustworthiness value based on the four attributes	Software trustworthiness value based on user feedback
No79		9.62	8.73	9.20	8.95	9.25	8.97
No142		9.17	9.24	9.74	7.95	9.11	9.66
No116		8.37	7.69	8.93	8.45	8.58	8.65
No161		7.95	7.93	7.24	7.23	7.49	7.25
No11		6.16	6.74	5.98	5.17	5.88	6.00
No167		5.21	5.00	5.01	5.38	5.15	4.94

Figures 2-3 describe the comparison of trustworthiness values of the 170 pieces of software calculated in these two ways.

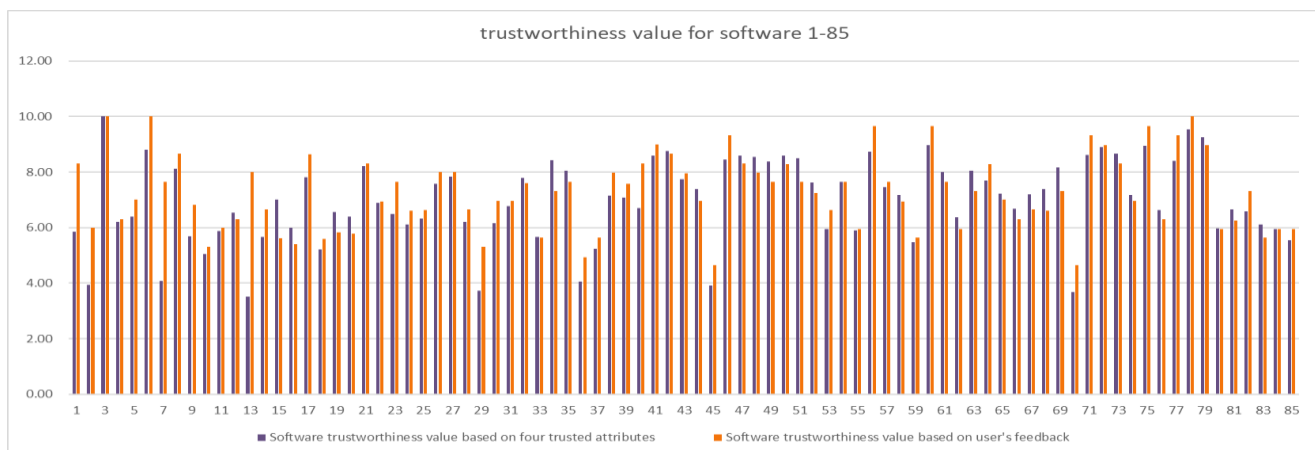


Figure 2. Trustworthiness value for software No1-No85

There are 94 pieces of software with error less than 0.5 and 148 pieces of software with error less than 1 in the software trustworthiness values calculated by these two methods. This result verifies the correctness of the formula. The

trustworthiness values of software No79 and No142 are 9.25 and 9.11, respectively. These two have 3/4 observed variables with scores greater than 9. Software No116 and No161 have trustworthiness values of 8.58 and 7.49, respectively. For No116, half of the observed variables scored below 9, and for No161, half of the observed variables scored less than 8. The trustworthiness values of software No11 and No167 are both less than 6, and their scores of the observed variables are almost less than or equal to 6. This proves that our method is scientific and rational.

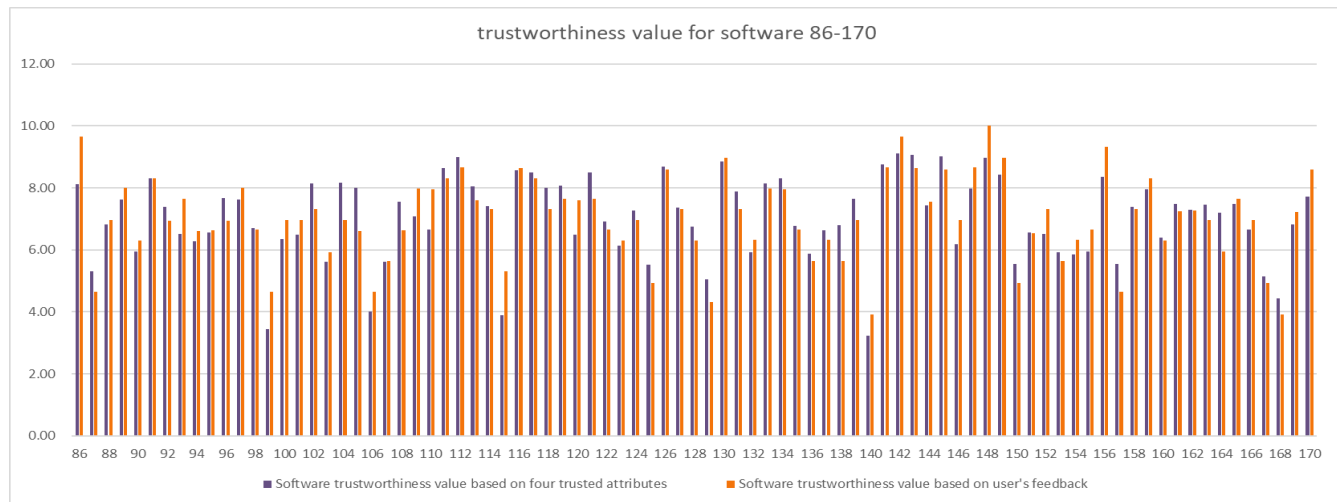


Figure 3. Trustworthiness value for software No86-No170

4. Conclusions

With the wider application of software and the increasingly complex application environment, the requirements for software quality has become higher and higher. A software trustworthiness evaluation method with higher accuracy is necessary. We chose four trustworthy attributes of functionality, reliability, safety, and maintainability to establish a software trustworthiness evaluation indicator system. Then, we constructed the SEM for software trustworthiness evaluation to obtain the weight. Finally, we combined the software trustworthiness measurement calculation model to calculate the values of the four trustworthy attributes and the software trustworthiness. This provides a new way of thinking for weight calculation and provides a reference for quantitative research on attribute relationships.

Studying the relationship between attributes is very important for the attribute-based software trustworthiness evaluation, the trade-off analysis of the attributes, and the improvement and evolution analysis of the software trustworthiness [17]. However, limited research has been conducted. We only considered the effect of some trustworthy attributes on the software trustworthiness. The design of the observed variables was not comprehensive enough, and only the software developers were involved in the investigation. The value of the software trustworthiness may have had errors. Future work will consider broadening the scope of the survey, increasing the number of samples, and adjusting the number and content of trustworthy attributes and observed variables.

Acknowledgements

This work is supported by Fitsco. The authors would also like to express their deepest gratitude to K. L. He for his revision of this paper.

References

1. Z. H. Jin and L. L. Ye, "Fatal Bug: Disaster and Enlightenment of Software Errors," Posts and Telecom Press, 2016
2. H. Shi, J. Ma, and F. Y. Zou, "A Fuzzy Comprehensive Evaluation Model for Software Dependability based on Entropy Weight," in *Proceedings of International Conference on Computer Science and Software Engineering*, IEEE Computer Society Press, pp. 683-685, 2008
3. B. Li and Y. Cao, "An Improved Comprehensive Evaluation Model of Software Dependability based on Rough Set Theory," *Journal of Software*, Vol. 4, No. 10, pp. 1152-1159, 2009
4. L. Shi, S. L. Yang, K. Li, and B. G. Yu, "Developing an Evaluation Approach for Software Trustworthiness using Combination Weights and TOPSIS," *Journal of Software*, Vol. 7, No. 3, pp. 532-543, 2012
5. B. H. Yu, "Measurement Theory and Method of Trustworthy Software," Science Press, 2014

6. D. Jun, J. Qi, X. Li, and W. Jing, "Research of Performance Evaluation on Sustainable Supply Chain based on Structural Equation Modeling," *Ecological Economy*, Vol. 31, No. 4, 2015
7. J. Wang, Y. X. Chen, B. Gu, X. Y. Guo, B. H. Wang, S. Y. Jin, et al., "An Approach to Measuring and Grading Software Trust for Spacecraft Software," *Science China Technological Sciences*, Vol. 45, No. 2, 2015
8. T. Boland, C. Cleraux, and E. Fong, "Toward a Preliminary Framework for Assessing the Trustworthiness of Software," NIST Interagency Report 7755, 2010
9. W. Hasselbring and R. Reussner, "Toward Trustworthy Software Systems," *Computer*, Vol. 39, No. 4, pp. 91-92, 2006
10. K. Liu, Z. G. Shan, J. Wang, J. F. He, Z. T. Zhang, and Y. W. Qin, "The Trustworthy Software Basic Research Summary of Major Research Plan," *Bulletin of National Natural Science Foundation of China*, Vol. 22, No. 3, pp. 145-151, 2008
11. H. W. Tao, "Research on the Measurement Models of Software Trustworthiness based on Attributes," Doctoral thesis of East China Normal University, 2011
12. Y. W. Du, "An Approach to Optimize the Project Portfolio of Venture Capital and Its Application Research," Jilin University, Changchun, 2007
13. ISO, "Software Engineering - Product Quality, Part 2: External Metrics," International Organization for Standardization, ISO 9126-1:2001, 2001
14. ISO, "Software Engineering - Product Quality, Part 3: Internal Metrics," International Organization for Standardization, ISO 9126-1:2001, 2001
15. Railway Application, "The Specification and Demonstration of Reliability, Availability, Maintainability and Safety (RAMS)," GB-BSI, 1999
16. M. L. Wu, "Structural Equation Modeling -- Operation and Application of AMOS," Chongqing University Press, Chongqing, 2010
17. W. X. Zhang, W. H. Liu, and X. Wu, "Quantitative Evaluation Model, Method and Implementation of Software Trustworthiness," Tsinghua University Press, 2015