A Minimal Test Cases Calculation Method for Software Trustworthiness Test

HongXing Kan
School of Medical Information Technology, Anhui University of Traditional Chinese Medicine,
Hefei, Anhui, China
Email: ffdkhx@ahtcm.edu.cn

JiLi Hu*, Li Jin, LuYao Zhang
School of Medical Information Technology, Anhui University of Traditional Chinese Medicine,
Hefei, Anhui, China
Email: yymozly@gmail.com

***Abstract*—the credibility of software needs to be tested thoroughly with many testing cases before being used. However, more test cases will increase the total software's developing expenditure. In order to cut the expenditure of software testing as much as possible, we present a method to calculate the minimal test cases for software trustworthiness properties test. First, according to the quality characteristics and specific application background of the software, we determine the composition of software trustworthiness properties and complex interaction between trustworthiness properties with the method of AHP, and then construct the hierarchical structure of trustworthiness properties. Second, based on the hierarchical structure of trustworthiness properties, a top-down comparison matrix was constructed through pair-wise comparison. The eigenvectors of the matrix was calculated and consistency test was done to get the software's different trustworthiness properties weights. Third, the ratio relationship among trustworthiness properties was calculated and the minimal testing cases at certain confidence level were obtained based on classical statistical hypothesis theory. Finally, we provide an application case to elaborate the process and effectiveness of the method.**

***Index Terms*—trustworthiness properties, test, AHP, statistical hypothesis, minimal test cases**

I. INTRODUCTION

Nowadays computer has been used in very field, such asocial society, economy and national security. All of those applications depend on the credibility of software. However, the credibility of software is unreliable sometimes and any unsure software fail can lead to adverse effect or even disaster to people's life. For example, The Explosion of Ariane 5[1], Therac-25 radiation therapy accident [2] and The Chinook Helicopter Disaster are the lessons for the entire society [3]. By reviewing those accidents, people pay more attention to correctness, reliability, security and fault tolerance properties of the software. In the United States, DARPA, NSF, NASA, NSA, NIST, FDA, FDA and DoD agencies are actively involved in research and development of trustworthy software system. NSTC has formed a series of research reports [4-7]. Trusted software means that the operating behavior and result of the software system is always in line with our expectations, meanwhile, the trusted software can also provide continuous service with presence of disturbance [8]. Software trustworthiness properties often include reliability, reliable safety, survivability, faulty tolerance and real-time [9].

During software development, rigorous test of software by executing the software to determine whether the software possess desired attributes is an effective, essential and objective method to evaluate the trustworthiness of software [10]. Software testing is a labor extensive and expenditure expensive step. Presently, most software development institutions spend more than 40% of research fund on software testing. In some circumstance the percentage may increase to more than 50% for software with high reliability requirement [10]. Software testing is a comprehensive testing which not only includes functional and performance testing, but also contains reliable properties and running environment.

The comprehensive testing of a trusted software does not mean to test all reliability properties in detail equally. Otherwise the expenditure will be increase enormously or impossible to achieve it. The software with different applications has different focus of the testing varies with different trustworthiness required[11]. For some Safety Critical Systems, such as flight control system, nuclear reaction monitoring system and critical patient care system, the whole system will be in severe crisis once the software failed. Therefore, in addition to basic functions and performance testing, reliability and safety properties should be tested for those kinds of software[12]. For some management information systems operating under different environments, trusted properties testing should focus on fault-tolerance of data and environmental compatibility. In fact, differentiated and focus-specific

This work is supported by the Natural Science Foundation of Anhui Province (No.11040606M187), Talent Development Fund ((No.2009z046) and Department of Education Fund ((No.KJ2010A346).
*Corresponding author : JiLi Hu , yymozly@gmail.com

doi:10.4304/jsw.9.8.2087-2092

testing of trusted software is one of the strategies to save the high cost of test.

This article studies the differentiated and focusing-specific testing strategy of software with AHP method and classical statistical hypothesis theory. We proposed a new method to find the minimum number of test cases needs to test the trustworthiness of software so as to reduce the expenditure of testing, and at the same keep the expected level of trustworthiness.

II. The Minimum Number Of Test Case Calculation

A. Analytical Hierarchy Process

In 1970s, Saaty (1978, 1979, and 1980) proposed an Analytical Hierarchy Process (AHP) to offer the qualitative and quantitative factor for decision makers in Multiple Criteria Decision Making (MCDM) [13]. Users of the AHP first decompose their decision process into a hierarchy of more easily comprehended sub-levels, each of which can be analyzed independently. The hierarchy is often structured in at least three levels:

The goal: what will AHP measure, e.g., prioritize organisms for survey activities

The criteria: elements integral to attaining the goal, e.g., biological effects, economic effects, etc.

The alternatives: the organisms of concern.

After classifying the hierarchy, layer factors can be compared in pairs to build comparison and judgment matrix. Assuming that element A_k in the layer A and element B_i (i = 1, 2... n) in the lower layer B are related, then we will compare elements in pairs in B and build A - B judgment matrix M. $m_{i,j}$ is the value of judgment matrix, which can be viewed as the judgment value of relative importance of B_i on B_j for A_k. These values are often given by domain experts based on their experiences and background knowledge.

After all matrices are developed, eigenvectors and the maximum eigenvalue (λ_{max}) for each matrix are calculated. The λ_{max} value is an important validating parameter in AHP. It is used for calculating the Consistency Ratio (*CR*)(Saaty, 2000) of the estimated vector in order to validate whether the pair-wise comparison matrix provides a completely consistent evaluation. The consistency ratio is calculated as following:

Step1: Calculate the eigenvector or the relative weights and λ_{max} for each matrix of order *n*.

Step2: Compute the consistency index for each matrix of order *n* by the formulae(1):

$$CI = (\lambda_{\max} - n)/(n-1) \tag{1}$$

Step3: The consistency ratio is then calculated using the formulae(2):

$$CR = \frac{CI}{RI} \tag{2}$$

where Random Consistency Index (*RI*) varies depending upon the order of matrix. Tables 1 shows the value of the Random Consistency Index (*RI*) for matrices of order 1 to 10 obtained by approximating random indices using a sample size of 500[14].

TABLE I.
AVERAGE RANDOM INDEX (RI) BASED ON MATRIX SIZE (SAATY, 2000)

S. No.	Size of Matrix (n)	Random Consistency Index (RI)
1	1	0
2	2	r
3	3	0.52
4	4	0.89
5	5	1.11
6	6	1.25
7	7	1.35
8	8	1.40
9	9	1.45
10	10	1.49

The *CR* is acceptable if it does not exceed 0.10. If *CR* is more than 0.10, inconsistency of judgments within that matrix has occurred and the evaluation process should therefore be reviewed, reconsidered and improved. An acceptable consistency ratio helps experts to determine the priorities of a set of criteria with high reliability.

After consistency testing of λ_{max} , corresponding eigenvector W' can be calculated using the following formulae (3).

$$(M' - \lambda_{\max} I)W' = 0 \tag{3}$$

Where I denote the unit matrix and W' can be expressed as formulae (4):

$$W' = (w_1', w_2', \ \dots\ , w_n') \tag{4}$$

Weights can also be normalized (put on a 0 to 1 scale).

After normalization of W', we get W''.

$$W'' = (w_1'/\sum_{i=1}^{n} w_i', w_2'/\sum_{i=1}^{n} w_i', \ \dots\ , w_n'/\sum_{i=1}^{n} w_i') \tag{5}$$

Getting normalized eigenvector, hierarchical ranking should be done for each layer and then different layers. Finally, we will get all ranking weigh for each minimum layer scheme relative to the target. Consistency test should be carried out for both ranking within single layer and ranking for all levels to distinguish whether they have a satisfactory consistency. If not, we need to adjust level judgment matrix until achieve consistency requirements. Using this AHP method, we can get relative importance degree of trustworthiness properties of software. Also, it provides the basis for the distinction and targeted software testing.

B. The establishment of the hierarchy of the trusted properties

According to the description in ISO/IEC 9126, trusted properties are constituted by many composition elements. Such as MTTF, integrity, stability reflects the reliability of software; MTTUF, integrity, and confidentiality represent the security properties; Analyzability, changeability, and stability represent the maintainability properties. Of course, some elements reflect not only just one trusted property; they may also have some intercross. For example, software integrity not only reflects the reliability properties, but also the security properties; Stability reflects both maintainability properties and the reliability of the software. Therefore, the hierarchy model of trustworthy software can be established by the following method:

- Goal (A): evaluate all trusted properties of trusted software.
- Criterion (B): factors constituted the trusted properties of software such as MTTF, MTTUF, integrity and privacy and so on, which represent by B_1, B_2, B_3... B_n.
- Alternatives (C): the reliability of software, such as reliability, safety, compatibility, and fault tolerance properties of software and so forth. Marked withC_1, C_2, C_3... C_n.

Based on a hierarchical model, we can get *n+1* judgment matrixes including A-B, B_1-C, B_2-C, B_3-C and so on. Then according to experience data and score by experts, we get the value of individual judgment matrix. Through calculating the matrix's maximum eigenvector and the normalized eigenvector, we can get the hierarchy of object layer C level. Then we finally obtain ranking among different trusted properties by weight, laying the foundations of exterminating minimum number of test cases.

C. Determination of the minimum number of test cases

According to the theory of classical statistical hypothesis testing [16-18] method proposed by David, Howden and Parnas et al., when one trusting property is being tested and the failure probability is *P*, and each operation meets the statistical independence of the Bernoulli (Bernoulli) experiment. Then, failure probability of *R* out of *n* tests meets the binomial distribution (See Formulae (6)).

$$P(R=r)=C_n^r P^r (1-P)^{n-r} \tag{6}$$

Under the above hypothesis, we can use classical statistical hypothesis testing method to determine test number *N* in engineering practice. The classical statistical hypothesis method set two opposite hypothesis for some prescribed trustworthiness index (P_0, *C*).

$$\left.\begin{aligned} H_0 &: P \le P_0 \\ H_1 &: P > P_0 \end{aligned}\right\} \tag{7}$$

Set the test significance level as $\alpha = 1-C$.Usually, if *r* times failure in some trusted test is allowed, the number of test case, *N*, should be the minimum value *n* to meet following formulae (8).

$$\sum_{i=r+1}^{n} C_n^i P_0^i (1-p_0)^{n-i} \ge 1-\alpha \tag{8}$$

Specifically, when *r=0*, the test intolerance is VALID and then

$$N=\left\lceil \frac{\ln(1-C)}{\ln(1-P_0)} \right\rceil \tag{9}$$

Using the above results analyzed by AHP method and combining with formulae (9), we will get minimal test cases for different trustworthiness test requirements.

III. METHOD APPLICATION

Liquid metal detection software (LMDS) is real-time online intelligent software which service specifically for foundry industry. It has the real-time online management capability of quality control of industrial products and greatly enhances the market competence of the equipment manufacturing industry. Take the Liquid Metal Detection Software (LMDS) [19] as the research object. Based on the above method, we obtain minimum number of test cases among different trustworthiness properties of software.

A. The determinateness of LMDS trustworthiness properties weight

LMDS has the capability of rapidly testing 4 categories of 23 kinds' different marks molten iron with 18 material parameters. It has highly requirement for software reliability. LMDS monitors the key parameters in casting cycle under high temperature and high pressure environment. The strict requirement of software security property for LMDS is indispensable. Casting process involves a variety of casting techniques and materials. The environment is complex and variable. It requires LMDS owns highly fault tolerance ability. In addition, casting process cannot be randomly interrupted and it requires software failure can quickly restore. At this point the software must be able to be maintained in timely. So, it requires maintainability property. Therefore, LMDS system has highly requirements in software reliability, security, maintainability and fault tolerance properties. Comprehensive test of these four properties are extremely important.

Based on AHP method, LMDS trustworthiness evaluation can be set as the target layer. Affecting these trusted properties' elements can be obtained according to the description of the ISO / IEC 9126, completeness, stable, and easy of change, easy of recovery, MTTF and so on. These elements will constitute the criteria layer. Object layer is defined as the software reliability, safety, fault-tolerance and maintainability four trustworthiness properties. Figure 1 shows the relationships of all levels.

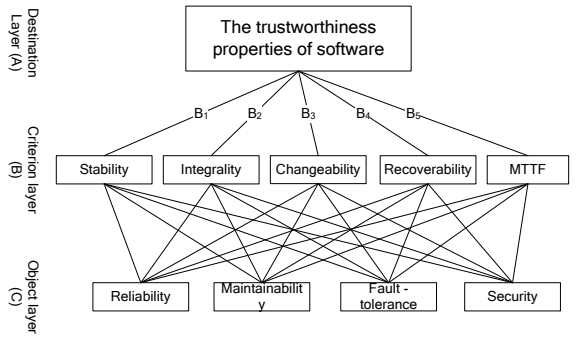


Figure1. LMDS trustworthiness properties hierarchical chart.

According to the Figure 1 hierarchical structure, we build A-B judgment matrix named *M* at first. For example:

$$M=\begin{bmatrix} A & B_1 & B_2 & B_3 & B_4 & B_5 \\ B_1 & 1 & \frac{1}{2} & 4 & 3 & 3 \\ B_2 & 2 & 1 & 7 & 5 & 5 \\ B_3 & \frac{1}{4} & \frac{1}{7} & 1 & \frac{1}{2} & \frac{1}{3} \\ B_4 & \frac{1}{3} & \frac{1}{5} & 2 & 1 & 1 \\ B_5 & \frac{1}{3} & \frac{1}{5} & 3 & 1 & 1 \end{bmatrix}$$

Using Matlab, we calculate the maximum eigenvalue $\lambda_{\max} = 5.073$ of judgment matrix *M*, and the normalized eigenvector.

$$L_A = (0.263, 0.475, 0.055, 0.099, 0.110)^T$$

$$CR = \frac{0.018}{1.12} = 0.016 < 0.1$$

The results meet the consistency requirements. If the results do not meet the consistency requirements, we have to modify judgment matrix relative importance ratio scale.

We can use the same methods to get judgment matrix of B_1-C、B_2-C、B_3-C、B_4-C、B_5-C、B_6-C and those maximum eigenvalue feature factors. After consistency checking, we can get the level total sequencing of object C according to above vector level sort results.

C_1(reliability) (0.328)> C_4 (security) (0.321)> C_3 (fault tolerance) (0.301)> C_2 (maintainability) (0.291).

B. Solving minimum number of test cases

Assuming that the trustworthiness property C_1 (reliability) reached 99.9%, which is given by customs or domain experts, then, we can get other properties proportionally based on those properties weight. For example, security property C_4 is 97.8% (0.999×0.321/0.328 =0.978), faulty and tolerance property C_3 is 91.7% (0.999×0.301/0.328=0.917), maintainability property C_2 is 88.6% (0.999×0.301/0.328=0.917). According to these trustworthiness requirements, using the formulae (9), we can make the different trustworthiness properties in different significant level of the minimum test number. As shown in Table 2.

TABLE 2
THE MINIMUM NUMBER OF TEST CASES WITH DIFFERENT TRUSTWORTHINESS REQUIREMENT

R_0=1-P_0	C=1-α			
	0.90	0.95	0.99	0.999
reliability（0.999）	2302	2995	4603	6905
security（0.978）	104	135	207	311
fault tolerance（0.917）	27	35	53	80
maintenance（0.886）	19	25	38	57
…	…	…	…	…

As can be seen in the Table 2, at the same degree of confidence, the required number of test cases greatly reduced as the dependability requirements decrease. For example, we need at least 2302 test cases under 90% confidence levels, when reliability value reached 99.9%, while 97.8% security tests at only 104 test cases. The trustworthiness of the testing requirements reduces 2.10%, yet the test cases number has decreased by 95.5%. Figure 2 shows the trend of the minimum number of test cases in different degrees of confidence and the different trustworthiness requirements (for comparison, the ordinate is logarithmic coordinates in Figure 2).

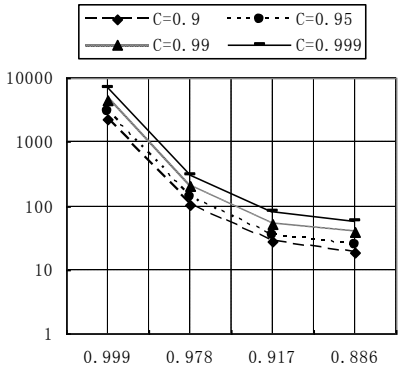


Figure 2. Minimum number of test cases change curve.

Figure 2 shows that, no matter what confidence, a slight change of software trustworthiness requirements will lead to large fluctuations in the number of test cases. So in the trusted software testing, we must distinguish those trustworthiness properties and have pertinently test. A tiny difference will save a lot of test cases.

IV. CONCLUSION

The testing of trusted software is the comprehensive test of credible natures and operation environment, which is very complex and needs many test cases. In present paper, based on the ideology of distinctiveness and pertinence, we discuss the method of using minimum number of test case to examine the trustworthiness properties of software. Furthermore, we take the LMDS as the example to explain the validity of the method. The validity of trusted software test not only depends on the number of test cases, but also lies on its designed quality.

In further study, we will discuss how to design the high-quality test cases under the limit of minimum quantity.

Moreover, the accurate degree of solving the minimum test cases number depends on the construction and value of the judgment matrix, while the values depend on the experts' experience and relevant historical data. So the method discussed in the paper applies only to those organizations that has qualified experts and associated database. For those conditions without the associated experience data and experts, the method here may provide a reference. In future based on the method of group decision-making, we will establish an adaptive system based on the evolution characteristics [20] of software requirements, which will be used as the expert-storehouse and knowledge library to serve as a spur for test institution to use the method more correctly and accurately.

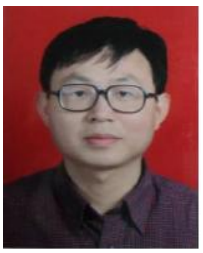
ACKNOWLEDGMENT

We thank the following people for their comments on previous versions of the paper: Shanlin Yang, Shuai Ding,and Xijun Ma.

Part of this work is supported by the Natural Science Foundation of Anhui Province, (No.11040606M187), Talent Development Fund (No.2009z046) and Department of Education Fund (No.KJ2010A346).

REFERENCES

[1] Selding P B. Faulty software caused Ariane 5 failure [J]. Space News, 1996, 25 (7): 24 - 30.
[2] Leveson N.G., Turner C. S. An investigation of t he Therac225 accident [J]. IEEE Computer, 1993, 26 (7): 18 - 41.
[3] http://en.wikipedia.org/wiki/Software_bug
[4] National Science, Technology Council (NSTC). America in the Age of Information: A Forumon Federal Information and Communications R&D[R] .Bethesda, Maryland, July 6 - 7, 1995.
[5] NSTC. Research challenges in high confidence systems [A]. Proceed2ings of the Committee on Computing, Information, and CommunicationsWorkshop [C]. USA: Http://www.hpcc.gov/pubs/hcs2Aug97/ ntro.html, August 6 - 7, 1997.
[6] High Confidence Systems Working Group, NSTC. Setting an interagencyhigh confidence systems (HCS) research agenda [A]. Proceedings ofthe Interagency High Confidence Systems Workshop [C]. Arlington, Virginia, 25 March 1998.
[7] High Confidence Software and Systems Coordinating Group. High Confi2dence Software and Systems Research Needs[R]. USA: Http://www.ccic.gov/pubs/ hcss2research. pdf ,January 10 ,2001.
[8] Liu Ke, Shan Zhi Guang, Wang Ji. Overview on major research plan of trustworthy software [J]. Bulletin of National Natural Science Foundation of China, 2008, 22(3):145-151 (in Chinese).
[9] Yang Shan-lin, Ding Shuai, Zhu Wei.A trusted software evaluation method based on utility and evidence theory[J].Journal of Computer Research and Development, 2009, 46(7): 1152-1159
[10] CHEN Huo-wang, WANG Ji, Dong Wei. High Confidence Software Engineering Technologies [J]. Chinese Journal of Electronics, 2003, 31 (12):1933-1938.
[11] Dozelli P, Basili V. A Practical Framework for Eliciting and Modeling System Dependability Requirements: Experience from the NASA High Dependability Computing Project [J].The Journal of Systems and Software , 2006, 79 :1072119
[12] Tan Zhi-dong, Lei Hang, Sang Nan.Study of Security and Critical Software Reliability Demonstration Testing Method [J]. Chinese Journal of Aeronautics, 2005, 26(3):334-339.
[13] T.L.Satty. Fundamentals of Decision Making and Priority Theory with the Analytic Hierarchy Process [M]. RWS Publications, 2000.
[14] Sanjay Kumar, Neeraj Parashar, Abid Haleem. Analytical Hierarchy Process Applied to Vendor Selection Problem: Small Scale, Medium Scale and Large Scale Industries [J]. Business Intelligence Journal, 2009, 2(2):355-362.
[15] H W Jung, S G Kim, C S Chung. Measuring Software Product Quality: A Survey of ISO/IEC 9126[J]. IEEE Software, 2004, 21(5): 88-92.
[16] David L P, John A, Kwan S P. Evaluation of safety critical software [J]. Communication of ACM, 1990, 33 (6):636 - 648.
[17] Parnas D L, Asmis GJ K, Madey J. Assessment of safety critical software in nuclear power plants [J]. Nuclear Safety, 1991, 32 (2): 189 - 198.
[18] Howden W E. Good enough versus high assurance soft ware testing and analysis methods [A]. In: Regina S Sed. Proceedings of the Third IEEE International High Assurance Systems Engineering Symposium [C]. Washington D C: IEEE Computer Society, 1998. 166 - 175.
[19] Zhan Hui, Zhang Qicai. Online liquid cast iron material parameters intelligent detection and quality control system [J]. China Awards for Science and Technology, 2007, 3: 38-41 (in Chinese)
[20] Ding Shuai,Yang Shan-lin, Kan Hong-xing. The Credibility Assessment Adaptive Model for Software Evolution [J]. Geomatics and Information Science of Wuhan University, 2010, 35(5):542-545.



HongXing Kan was born in Hefei, Anhui province in 1972.He is an associate professor and PhD supervisor with School of Medical Information Technology at Anhui University of Traditional Chinese Medicine. His main research areas include Trusted Computing, Software Engineering, and Decision Support System.

JiLi Hu was born in Huainan, Anhui province in 1980.He is a lecturer and master of software engineering with School of Medical Information Technology at Anhui University of Traditional Chinese Medicine. His main research areas include Data Mining, Image Recognition, and Software Engineering.

Li Jin was born in Hefei, Anhui province in 1974.He is a lecturer and master of engineering with School of Medical

Information Technology at Anhui University of Traditional Chinese Medicine. His main research areas include analysis & design of algorithm , Diagnosis and treatment of traditional Chinese Medicine Digital, Electronic Design Automation etc.

LuYao Zhang is a student at Anhui University of Traditional Chinese Medicine. Now, she is working for M.S. degree in Computer Technology. Her current interests include Software Development and Image Processing.