A Measurable SocialToTech Software Trust Framework

Xi Yang, Ping Luo and Gul Jabeen
The Key Laboratory for Information System Security, School of Software, Tsinghua University, Beijing, China
x-yang14@mails.tsinghua.edu.cn

Abstract. With the rapid development of Internet, software trustworthiness has been increasingly important. How to measure this property has become a real challenge. Based on Sociology Trust system, this paper presents a framework which could be measurable, called as SocialToTech Software Trust Framework (SSTF). The SSTF includes three parts of software trustworthiness and this paper proposes a whole metric solution through a generalized indicator Cost-Loss.

1. Introduction
In the current open cyber environment, traditional reliability, security and quality metrics of software are no longer sufficient. Instead, when choosing software, users are more concerned about the software trustworthiness which plays a key role. Since 1983, important authorities of the world have been working on defining the term “trust” and providing frameworks for research on trust [1-3]. Meanwhile, researchers have advanced a number of definitions, methods and models of trust [3]-[4].

Even so, software trust, whose core concept should be definite and universal, is much debated and without an established definition. For example, a) in many studies, trustworthiness is subdivided into more software quality attributes [4], as done in traditional software quality research [5], which excluded application problems of software under the environment of Internet. b) Varieties of trustworthiness concepts and terminologies lead to greater confusions such as trustworthiness, dependability, confidence and so on, which cover different aspects of trusted computing and have overlapping as well as differences with one another. c) All the relevant research does not present a bounded, consistent framework for trustworthiness, which makes it difficult to quantify the value of trustworthiness. In reality, as an extremely important software feature [1]-[3], the quantification and evaluation of trustworthiness are as desperately needed. Traditional research has usually quantified software indicators based on reliability models [6]. However, a single attribute measurement of reliability fails to satisfy the evaluation requirements of complex software. Hence, in recent years, researchers worldwide have been researching to find a metrics method of overall credibility in software and present software trustworthy metrics models [7] from various angles. It is shame yet that all the currently available models are far from perfect: a) Some depend on artificial and subjective evaluation but lack objectivity; b) some take trustworthiness as a single attribute, without taking into account the bad effects of affecting the software’s overall credibility; and c) different trustworthy attributes have different metric standards and indicators, which makes it difficult to compare one another.

These issues can all be largely attributed to the immaturity and rapid development in the field. Consequently, to present a standard, measurable and integral trustworthy framework has become the focus of extensive and ongoing research in the current credible system.

2. SocialToTech Trust Framework
The word “trustworthy” which originated from sociology [4] is regarded as a virtue, a kind of moral
value. It is concerned with how the human brain reacts to the objective reality and evaluates others from social contract relationship. Thus, as a product of human thinking, software is similar to human society. Hence the question arises: is software trust the same as human trust in nature? For years, all the research on software trust has been using diverse definitions, methods, and metrics models, which results in wide debate. Naturally, no agreement has ever been reached and little drastic progress has been made concerning software trust. We believe that this is predominantly because the current research fails to grasp the nature of software trust and fails to propose a metrics system as axiomatic as mathematic theories. Therefore we believe that a credible and axiomatic software trust framework should be situated in sociology as well. Only based on this, could it possible to reach and widespread agreement on software trust and further explore its various attributes, characteristics, future modeling, and quantification, etc.

Our research team has long been utilizing sociology theory to construct a strict and complete trust axiom system [8] for effective quantification and evaluation, aiming to reveal the essence of credibility. We established a practical model of trustworthy software in the cyber environment. In order to facilitate the software trustworthiness measurement, this paper consummate this model to a “SocialToTech trust framework”, as shown in Figure 1.

![Figure 1. SocialToTech Software Trust Framework.](image)

In this framework J is the third-party authority, accredited by the government with rich knowledge and powerful testing tools. It evaluates the trustworthiness of software system T(a) on behalf of regular judge Ji, providing evaluate solution for Ji to decide T(a). Historical information or evidence di represents the fundamental property of the software system in network T(a). Similar to the trusting problem of human, we are concerned with the core of trustworthiness, based on which J evaluates T(a). Evidently, T(a) is correlated with and affected by the running environment ET and Internet and cloud computing environment CT which includes various elements helpful to run the software system. ET and CT will finally influence the evidences provided by T(a). In addition, the model is related to time t, whose attributes emerge only in runtime. The software trustworthiness T(a) is a function of time t, shown as Ta(t). (The following concepts and definition that will be used later are described prior to the definition of software trustworthiness)

Based on the IEEE Std 610-12-1990, we get:
**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.

Please be cautious that failure is different from disfunction. If there is error tolerant design in development, the software can have failure but no disfunction. Otherwise failure becomes disfunction.

To cover more scenarios, we expands Definition 1:

**Definition 4 -- 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 software fault in the present paper refers to definition 4, based on which we defines vulnerability:

**Definition 5 -- vulnerability:** If the fault in software doesn't lead to disfunction in a normal situation (i.e., has no effect on the function of software), but may be utilized by attackers to execute extra baleful code, or leak out information. Then this fault is called vulnerability. And this scenario forms an invasion.

**Definition 6 -- Trustworthiness:** abbreviated as GT(Ta(t)), the trustworthiness of software T(a) refers to the property of being able to gain the trust of Ji with three types of evidence: identity evidence, basic standard or norms (i.e., the basic rule of the trade) evidence and ability evidence in the demanded environment E(t,r,n) and ET(CT) during the demanded time period t even with other factors (e.g., inappropriate operation) interfering or man-made attack.

Notes: Identity evidence involves the software T(a) itself and code sources, shared resources of codes, documents to use, configurations of hardware and software, illustrations of running environments, helpful documents, and so on. To be basic standard evidence, the design and requirement match with the basic standard of the trade or the principle, tactic of system safety so that it won't cause any system safety issue, result in other damage or problem (i.e., leaving private information). Ability evidence includes the reliability on broad sense (defined as Generalized Reliability, G-Reliability) and anti-attack ability of software (defined as Generalized Security, G-Security).

**Definition 7 -- Cost-Loss ω:** due to non-trust factors in software, users of software must pay an extra cost (i.e., money, maintenance workload, effort, lines of code, etc.), which can be grouped into three types: a) software-related extra cost to achieve the specified requirement without satisfying the basic standards; b) extra cost related to software wrong running caused by trusted problems in order to restore the software running; ; and c) extra cost related to the user's business loss resulted from failure operation.

**Definition 7.1 -- Identity Trustworthiness:** indicated as I(Ta(t)). It means, at time t, the Cost-Loss due to the existence of non-trust problems in the software identity evidence. If the value of I(Ta(t)) is higher, the lower trustworthiness is.

**Definition 7.2 -- Basic standard Trustworthiness:** represented as S(Ta(t)). It means, at time t, the Cost-Loss caused by the existence of non-trust problems in the software basic standard evidence.

**Definition 7.3 -- Ability Trustworthiness:** indicated as A(Ta(t)). It means, at time t, the Cost-Loss caused by the existence of non-trust problems in the software ability (including G-Reliability, indicated as AR(Ta(t)), and G-Security, indicated as AS(Ta(t))). evidence.

Then, measurements of different parts of trustworthiness are all incorporated into Cost-Loss ω. So GT(Ta(t)) is easily deduced as: GT(Ta(t)) = I(Ta(t)) + S(Ta(t)) + A(Ta(t)). With reference to theoretical derivations and applications, we advance the following solutions to measure the three trusted properties: a) propose the ability trustworthiness measurement method based on Compound Poisson Process and G-O model to obtain the value of A(Ta(t)); b) establish the rule sets to obtain the value of S(Ta(t)) based on the multiple short-board measure method; c) get the value of I(Ta(t)) depending on the code homology detection tool.

3. Proposed Metrics Solution
This section elaborates the trustworthiness value GT(Ta(t)) based on Cost-Loss.
3.1. Ability Trustworthiness Metrics Solution
The traditional reliability and security measure methods based on vulnerabilities and failures have the following two problems: a) not considering the consequences of the severity of vulnerabilities and failures which can result in the loss of cost; b) lack of a unified process to incorporate different aspects of ability trustworthiness. It worth noting that traditional measure methods often work as a function which takes the occurrence time of failures or vulnerabilities as a variable to predict the total number of failures or vulnerabilities and their future occurrences. Even so, it is not adequate to determine software reliability or security by simply relying on future occurrences and the amount of failures or vulnerabilities. Consequently, is necessary to it examine not only the emergence rate of failures or vulnerabilities but also the Cost-Loss due to the severity of failures and vulnerabilities.

For this purpose, we: (1) use the Cost-Loss as a indicator for the severity of the failures and vulnerabilities so as to predict the results of G-reliability and G-security. (2) choose a NHPP model to predict different indicators, because the distribution of failures and vulnerabilities accord with the stochastic process and their numbers accord with the Poisson process. (3) compute the Cost-Loss value through Compound Poisson Process and then obtain the AR(Ta(t)) and AS(Ta(t)).

However, we must observe that G-Reliability is based on failure data and G-Security is based on vulnerability data. The failure data has great randomness without human control, which accords with the assumption of the compound Poisson process. But vulnerability data is usually published regularly by the related security authorities or companies, although its appearance is also random, which causes a certain time difference. To solve this problem, we transform the time axis according to the time interval, and take the variable t of all G-Security parameters to zoom in or out.

More details and methods will be discussed in our coming papers.

3.2. The Basic Standard Trustworthiness Metrics Solution
As shown in Definitions 6 and 6.2, basic standard trustworthiness has industrial correlation characters. This section discusses a S(Ta(t)) metric method of multi-short board effect based on different industry rules specified by different industrial professional experts.

1) First, we develop the basic standard rule sets for different industries. To evaluate a software trustworthiness, several different experts are invited to score each rule on a scale of 1 to 10, so that we could establish a matrix. Through that, we compute the final score for each rule.

2) To get the final value of S(Ta(t)), we introduce a basic attributes model of software trustworthiness. Based on this basic model, with reference to realities and multi-short boards effects, we present a comprehensive and advanced S(Ta(t)) metric model. Ultimately, through mathematical proof and data simulation, we conclude that this model is correct and effective.

It should be noted that the value of S(Ta(t)) is also determined by Cost-Loss ω which incorporates all parts of trustworthiness.

3.3. Identity Trustworthiness Metrics Solution
In this section, professional code homology detection tools are used to identify similar codes and their percentages from the history software for the testing code. It is assumed that the Cost-Loss caused by historical software during its lifecycle is proportional to the Cost-Loss of a similar code part. Then the testing software's Cost-Loss caused by the identity of code is predicted based on historical data.

However, more research in this area is called for.

4. Conclusion
The present paper describes a measurable software trust framework based on a complete axiom system to facilitate trustworthiness evaluation with the same assessment indicator Cost-Loss. Then, it illustrates different models and methods on the three parts of software trustworthiness respectively. To conclude, this contributes to the current literature in the following ways:

- It presents a measurable and unified software trust framework based on strict and complete sociology axiom system and clearly elaborates the concept of trustworthiness.
- It proposes a generalized indicator -- Cost-Loss which integrates all parts of trustworthiness metric value, based on which different softwares can be compared.
• It advances a comprehensive metric solution for three parts of software trustworthiness. However, we still have a lot of work to do in the software trustworthiness metrics.

5. Acknowledgement
This work is a part of a larger project which 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.

6. References
[1] Department of Defense, “Trusted computer system evaluation criteria [Orange Book],” Computer Security Evaluation Center, USA: http://csrc.nist.gov/publications/history/dod85.pdf, 1983.
[2] National Science, Technology Council (NSTC), “Research challenges in high confidence systems,” Proceedings of the Committee on Computing, Information and Communications Workshop. USA: http://www.hpcc.gov/pubs/hcs2Aug97/intro.html, August 6-7, 1997.
[3] S. Becker, W. Hasselbring, A. Paul, et al., “Trustworthy software systems: a discussion of basic concepts and terminology,” ACM SIGSOFT Software Engineering Notes, vol. 31, pp. 1-18, November 2006.
[4] D. H. McKnight, M. Carter, J. B. Thatcher and P. F. Clay, “Trust in a specific technology: an investigation of its components and measures,” ACM Trans. Manag. Inform. Syst. 2, 2, Article 12 (June 2011), 25 pages.
[5] N. Nagappan, B. Murphy, V. Basili, “The influence of organizational structure on software quality: an empirical case study,” ICSE '08: Proceedings of the 30th international conference on Software engineering, pp. 521-530, 2008.
[6] Z. Jelinski, P.B. Moranda, “Software reliability research,” in: W. Greiberger(ed), Statistical Computer Performance Evaluation, Academic Press, pp. 465-484, 1972.
[7] H. M. Wang, Y. B. Tang, G. Yin and L. Li, “Trustworthiness of Internet-based software,” Science in China Series F: Information Sciences, vol. 49, pp. 1156-1169, 2006.
[8] 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.