U.P.B. Sci. Bull., Series C, Vol. 82, Iss. 3, 2020 ISSN 2286-3540

CREDIBILITY EVALUATION METHOD OF SOFTWARE BASED ON FUZZY MATHEMATICS THEORY

Xuejun YU[1], Xiaofeng MIAO[2*]

With the continuous expansion of software scale, its internal structure is becoming more and more complex, and its application environment is becoming more and more open. These factors make people pay more attention to the credibility of software. Aiming at the problem of low accuracy of measurement methods in software credibility evaluation, this paper proposes a research method for software credibility evaluation based on fuzzy mathematics theory. This method uses fuzzy sets to characterize behavioral declaration and actual behavior, and then respectively give the credibility measure calculation method of single behavior and the credibility evaluation calculation method of software as a whole based on the idea of "words and deeds". This method gives the evaluation results in a quantified form, which improves the accuracy of the evaluation results compared with the previous credibility evaluation methods using qualitative analysis. This method has certain practical significance in the credibility evaluation of software, and provides a new idea for measuring the credibility of software.

Keywords: software credibility; fuzzy mathematics; credibility evaluation

1. Introduction

At present, the information infrastructure that is centered on communication, storage and computing, has penetrated into all levels of political, economic, military, cultural and social life [1]. The widespread use of software has greatly promoted the development of the information society. With the continuous increase in the scale and complexity of software, the problem of software credibility has become increasingly prominent. Once the software fails, it will definitely have an adverse impact on people's work and life, and even cause huge losses [1,2]. Therefore, evaluating the credibility of software is an important research direction to ensure the credibility of software at runtime.

Software credibility evaluation is a hotspot in software credibility research [3,4,18,20], and it is highly concerned by scholars in China and abroad. Yu Xuejun proposed a model for using the behavior declaration to guarantee the credibility of software in the whole life cycle of software [5]. Xiao Ran proposed an implicit indicators model based on K-means clustering for credibility measurement [6]. Liu Yuling proposed a trust model based on checkpoints for behavioral risk assessment [7], by weaving a number of checkpoints in the

[1] Associate Prof., Faculty of Information Technology, Beijing University of Technology, China
[2 *] Master of Software Engineering, Faculty of Information Technology, Beijing University of Technology, China, corresponding author, E-mail: miaoxiaofeng666@163.com

software behavior trajectory. This risk assessment strategy was used to determine the checkpoints with suspected risks. Amoroso and his colleagues divided the software's credibility into six levels based on how well the software process conforms to the software's trusted specifications [8]. Li and his colleagues proposed a credible evaluation model based on distrust factors in the software life cycle [9]. LI Zhen proposed a cloud service credibility evaluation model based on sliding windows [15]. WANG De-Xin proposed a bottom-up credible evaluation model of software process based on evidence [17]. Chen Qianqian and his colleagues proposed an improved algorithm to deal with the high conflict of evidence [19]. ZHANG Fan and his colleagues proposed a software real-time credible measurement theory based on the non-interference model [16].

Most of the above studies have obtained the expected behavior of the software by analyzing the historical behavior of the software [10,11]. Based on the results, it can be shown as a one-sided, static behavior, because it is an inductive method, which varies from deductive methods. Inductive methods cannot guarantee the completeness of the expected results. Therefore, it is impossible to fully determine the credibility of the software. In addition, most of the judgement results are described in a credible level, it is a qualitative analysis, containing a certain fuzziness.

This paper puts forward the idea that "a software behavior is composed of a finite number of ordered basic actions" and introduces the concept of "action space" for the first time. In addition, it redefines terms related to the credibility of research software by combining mathematical knowledge. Moreover, it focuses on the use of fuzzy sets to define behavior declaration and software behavior more precisely. Based on the above theory, this paper proposes a research method for software credibility evaluation based on fuzzy mathematics theory, and gives the concrete solution. First of all, we extract all the basic actions d contained in a software S by analyzing the trusted requirements. Then we construct the action space Ω corresponding to the software S and the membership function corresponding to the basic action d. Finally, a fuzzy set can be used to define the behavior declaration more precisely. In the stage of secure compilation, we add a secure compilation component to detect the completeness of the basic action ω contained in the software S compared to the action space Ω, and construct the membership function corresponding to the basic action ω included in the software S. In the stage of dynamic monitoring, we extract and characterize the actual running behavior of the software in order to verify the completeness of the behavior set of the software S. In the stage of credibility evaluation, we respectively give the credibility measure calculation method of single behavior and the credibility evaluation calculation method of software as a whole based on the idea of “words and deeds”. The feasibility of the method is verified by experiments, which provides a new measurement method for software credibility evaluation.

2. Related Concepts about Software Credibility

We combined mathematical knowledge to redefine the related concepts about software credibility. These concepts are also the theoretical basis for our subsequent credibility evaluation.

2.1 Related definition

Definition 1 Basic action: The simplest actions that make up software behavior, denoted as d .

Definition 2 Action space: All the basic actions of software behaviors make up a collection, denoted as $\Omega=\{d_1,d_2,d_3,...,d_n\}$.

Definition 3 Software behavior: A sequence is made up of certain basic actions, denoted as $B=\{d_{i_1},d_{i_2},d_{i_3},...,d_{i_k}\}$

From this definition, we can get that the software behavior has the following characteristics:

(1) Any software behavior B is a subset of the corresponding action space Ω .

(2) Software behavior B occurs if and only if the subset referred to in (1) is an ordered sequence.

Definition 4 Behavior declaration: The related people of software have an expectation about software behavior, denoted as D .

Definition 5 Behavioral measurement: For a software S , assume that its software behavior set is $\mathrm{B}=\{B_1,B_2,B_3,...,B_n\}$, the behavior declaration set is $\Delta=\{D_1,D_2,D_3,...,D_n\}$, the measurement result is $\mathrm{M}=\{M_1,M_2,M_3,...,M_m\}$; for $\forall B\in \mathrm{B}$, there is always $f(B)\in \mathrm{M}$, then we call $f:\mathrm{B}\overset{\Delta}{\rightarrow}\mathrm{M}$ is a behavioral measurement of software S .

Definition 6 Credibility evaluation: For a software S , assume that its software behavior set is $\mathrm{B}=\{B_1,B_2,B_3,...,B_n\}$, the behavioral measurement is f , then we call $\Phi\left(f(B_1),f(B_2),f(B_3),...,f(B_n)\right)=\varsigma$, (where $\varsigma\in[0,1]$) is a credibility evaluation of the software S .

2.2 Definition of behavior declaration

In the requirement analysis phase of the software, we obtain the trusted requirements of the software, so as to construct the action space Ω of the software, and then extract and characterize the behavior declaration of the software. Whether the software behavior is credible or not, it is a fuzzy concept. In order to give a quantitative evaluation result, we use the fuzzy mathematics research method to study the credibility of software behavior. Therefore, we use $\tilde{D}=\{(d_1,S(d_1)),(d_2,S(d_2)),(d_3,S(d_3)),...,(d_n,S(d_n))\}$, (where $d_i \in \Omega, i=1,2,3...n.$) to portray a software behavior declaration.

Behavior declaration as a criterion for judging the credibility of software behavior, we need to make a reasonable definition of the membership function $S(d)$. $S(d)$ indicates the degree of credibility of a basic action belonging to software S. We can extract all the basic actions contained in the software S by analyzing the trusted requirements. The frequency that appears in the software S with each basic action is taken as $S(d)$. Therefore, we can define $S(d_i)$ for each basic action as follows:

$$S(d_i)=\frac{c_i(d_i)}{\sum_{i=1}^{n} c_i(d_i)} \tag{1}$$

where $d_i \in \Omega, c_i(d_i) \in \mathrm{N}^+, i=1,2,3,...,n$. ($c_i(d_i)$ represents the number of times the basic action d_i appears in the software S).

3. Characterization of the Actual Running Behavior of the Software

The actual running behavior of the software can be expressed by referring to the definition of the behavior declaration. Then we need to know the membership function and sequence of the basic actions in the actual running behavior. In order to solve this problem, we constructed the membership function of the basic action in the secure compilation phase, and monitored the occurrence sequence of the basic action in the dynamic monitoring phase. The specific introduction is as follows:

3.1 Secure compilation

3.1.1 Principle of implementation of secure compilation

In the process of compiling the code by the usual compiler, a secure compilation component is added to perform security detection on the code. In the security compilation phase, we mainly detect whether the code contains all the basic actions contained in the action space Ω. If there are some basic actions missing or more, it will promptly inform the developers who can modify the action space Ω or modify the software code so that the two can match the content exactly. The process of secure compilation is shown in Fig. 1.

The secure compilation component can detect class names, constant names, variable names, method names, and so on. A method can be used as a basic action d. The secure compilation component does not take into consideration the specific logical structure of the basic action, and only considers whether the basic action is contained in the software source code.

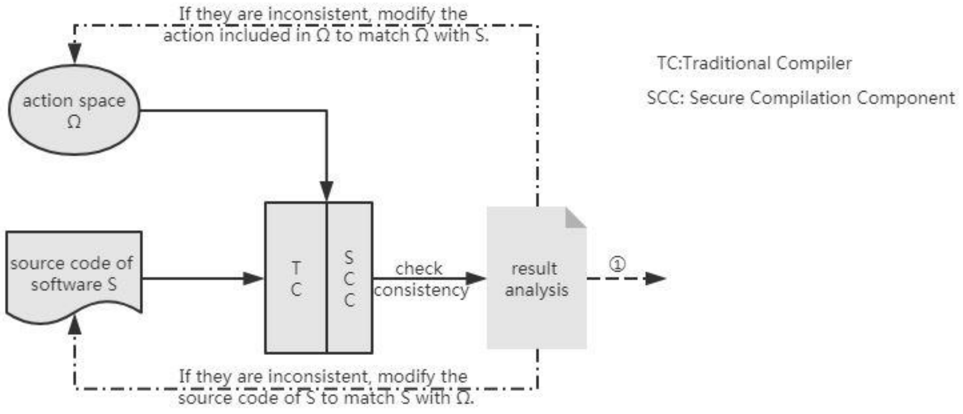

If they are inconsistent, modify the action included in Ω to match Ω with S.
TC:Traditional Compiler
SCC: Secure Compilation Component
action space Ω
source code of software S
T C
S C C
check consistency
result analysis
①
If they are inconsistent, modify the source code of S to match S with Ω.

Fig. 1. The process of secure compilation.

3.1.2 The main functions and purposes of secure compilation

(1) It is used to detect the completeness of the basic action ω contained in the software S compared to the action space Ω.

The completeness of the basic action ω contained in the software S is a prerequisite for ensuring the credibility of the software behavior. If the software S lacks some of the basic actions specified in the action space Ω, it will inevitably lead to some software behavior being untrusted. When the completeness of the basic action ω contained in the software S is guaranteed, the software behavior may be in a credible state.

(2) It is used to construct the membership function of the basic action ω contained in the software S.

Software behavior $B=\{\omega_{i_1},\omega_{i_2},\omega_{i_3},\ldots,\omega_{i_k}\}$ is a sequence consisting of some basic actions in the action space Ω. Since each basic action ω corresponds to the credibility of the behavior B, there is a certain degree of membership. Therefore, in order to better describe a software behavior in the dynamic monitoring phase, we will use the membership degree of the basic action ω in the software S approximately as the membership degree of the basic action ω in the software behavior B.

3.1.3 Method of constructing membership functions

By analyzing the results of the secure compilation, we can classify the test results into the following three categories:

(1) The software S contains the basic action ω, and the basic action ω belongs to Ω;

(2) The software S does not contain the basic action ω, but the basic action ω belongs to Ω;

(3) The software S contains the basic action ω, but the basic action ω does not belong to Ω;

Therefore, we can construct the membership function corresponding to the basic action ω contained in the software S as follows:

$$S'(\omega) = F_{\tilde{S}(\omega)} = \begin{cases} S(d), & \omega \in S, \text{and } \omega \in \Omega, \omega = d \\ H(\omega), & \omega \in S, \text{and } \omega \notin \Omega \\ 0, & \omega \notin S, \text{and } \omega \in \Omega \end{cases} \tag{2}$$

Among them, for the case of $\omega \in S, \text{and}\, \omega \notin \Omega$, we can use the fuzzy statistical method to construct it.

The randomness of probability theory research and the fuzziness in fuzzy theory are both uncertain. Therefore, the idea of probability and statistics can be used to construct the membership function corresponding to $H(\omega)$. This method is called fuzzy statistical method. The idea of probability and statistics is to repeatedly perform random experiments, when the number of trials tends to infinity, the frequency at which an event occurs will tend to a stable value, and this value is called the probability in which this event will occur. The idea of fuzzy statistical method is to repeatedly define the fuzzy set, when the number of definitions tends to infinity, the number of times at which anything belongs to the set will tend to a stable value, and thus, the ratio of this value to the number of definitions is the membership of the thing.

Since $\omega \in S, \text{but}\, \omega \notin \Omega$, in this case we cannot assert whether the basic action ω is harmful to the software S, the inclusion relationship between the basic action ω and the software S can only be described by a certain degree of membership.

3.2 Dynamic monitoring

3.2.1 Principle of implementation of dynamic monitoring

AOP (Aspect Orient Programming), which we generally call aspect-oriented programming, which is a complement of object-oriented, is used to deal with the crosscutting concerns distributed in each module in the system, such as: transaction management, logging, caching, exceptions, and so on. We can choose to embed the facet at compile time, or we can choose to embed the facet at runtime. It does not change the code logic, nor does it affect the software function implementation. Therefore, it is a feasible operation to use AOP technology to implant the monitoring module at runtime. The process of dynamic monitoring is shown in Fig. 2.

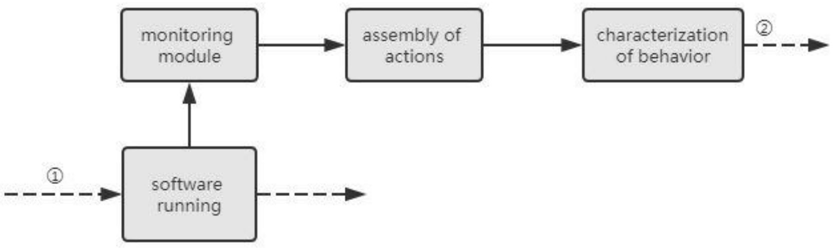


Fig. 2. The process of dynamic monitoring

3.2.2 The main functions and purposes of dynamic monitoring

(1) It is used to extract and characterize the actual behavior of the software.

We characterize the actual behavior of the software by monitoring the execution sequence of the software's basic actions ω during the running phase. Therefore, we can use $\tilde{B}=\{(\omega_1,S'(\omega_1)),(\omega_2,S'(\omega_2)),...,(\omega_n,S'(\omega_n))\}$ to describe the actual running behavior of the software. Since we constructed the membership function of the basic action ω contained in the software S during the secure compilation phase, the $S'(\omega_i)$ in $\tilde{B}$ (where $i=1,2,3,...,n.$) is easy to obtain.

(2) It is used to verify the completeness of the behavior set B in the software S.

We can obtain the actual behavior set $\mathrm{B}=\{\tilde{B}_1,\tilde{B}_2,\tilde{B}_3,...,\tilde{B}_n\}$ by monitoring the running behavior of the software S, since the behavior declaration set is $\Delta=\{\tilde{D}_1,\tilde{D}_2,\tilde{D}_3,...,\tilde{D}_n\}$, the completeness of the behavior set B in the software S can be tested. The test results can be divided into the following five cases:

a. When $\mathrm{B}\cap\Delta=\varnothing$, it indicates that the actual behavior set of software S does not match the behavior declaration set completely.

b. When $\mathrm{B}\cap\Delta\neq\varnothing$, and $\mathrm{B}\not\subset\Delta$, $\Delta\not\subset\mathrm{B}$, it indicates that the actual behavior set of the software S does not exactly match the behavior declaration set.

c. When $\mathrm{B}\subset\Delta$, it indicates that the actual behavior set of software S is not complete.

d. When $\mathrm{B}=\Delta$, it indicates that the actual behavior set of the software S is complete;

e. When $\Delta\subset\mathrm{B}$, it indicates that the actual behavior set of software S is complete, but there are also parts beyond the behavior declaration set.

4. Credibility Evaluation

In the previous two chapters, we present the behavior declaration and the actual running behavior based on the fuzzy set. Next, we will give a calculation method to measure the credibility of a single software behavior, and a calculation method to evaluate the overall credibility of the software. The process of credibility evaluation is shown in Fig. 3.

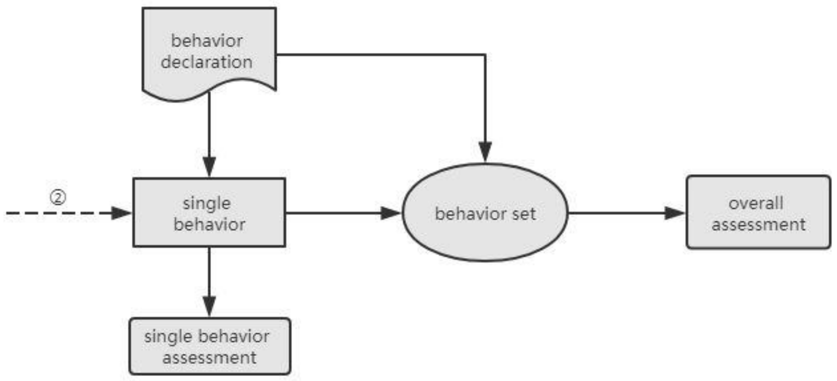


Fig. 3. The process of credibility evaluation

4.1 Method for calculating the credibility of a single behavior

For a software S, assuming that its behavior declaration is $\tilde{D}=\{(d_1,S(d_1)),(d_2,S(d_2)),\ldots,(d_n,S(d_n))\}$, the actual behavior $\tilde{B}=\{(\omega_1,S'(\omega_1)),(\omega_2,S'(\omega_2)),\ldots,(\omega_n,S'(\omega_n))\}$ corresponds to it, then the credibility of the behavior is $M=\sigma_H(\tilde{D},\tilde{B})=1-\frac{1}{n}\sum_{i=1}^{n}\left|S(d_i)-S'(\omega_i)\right|$.

4.2 Method for calculating the overall credibility of software

For a software S, assuming that $\tilde{\mathrm{M}}=\{(\tilde{B}_1,f(\tilde{B}_1)),(\tilde{B}_2,f(\tilde{B}_2)),\ldots,(\tilde{B}_n,f(\tilde{B}_n))\}$ is a measure set of all behavioral credibility on it, then the overall credibility of the software S is

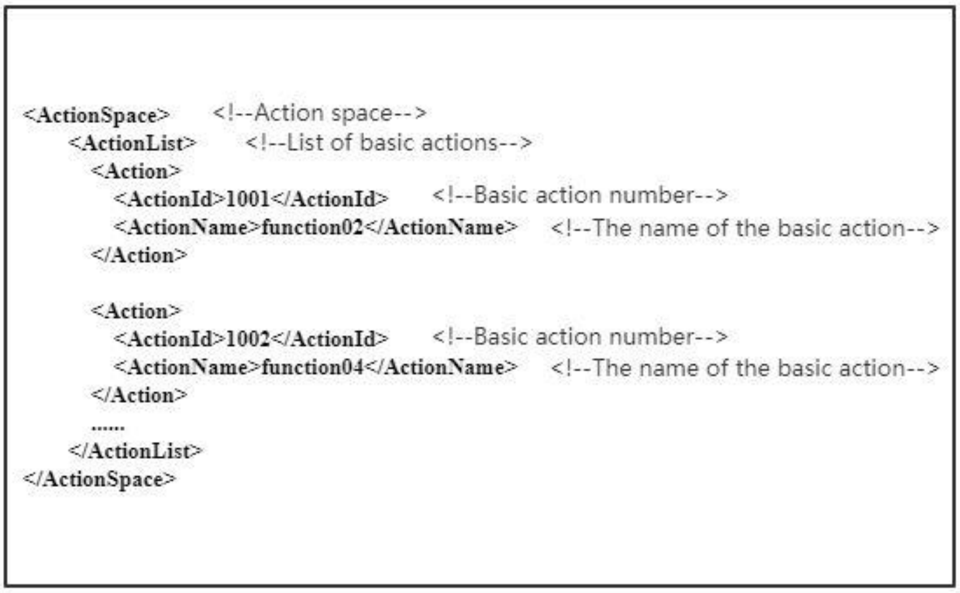
$$\Phi(\tilde{\mathrm{M}})=\left[\sum_{i=1}^{n}\frac{\left[f(\tilde{B}_i)-f'(\tilde{B}_i)\right]^2}{n}\right]^{\frac{1}{2}}$$

, where $f'(\tilde{B}_i)$ is a characteristic method of $\tilde{\mathrm{M}}_1$.

5. Experiments

5.1 Testing the feasibility of secure compilation

(1) Fig. 4 shows the representation of the action space Ω.



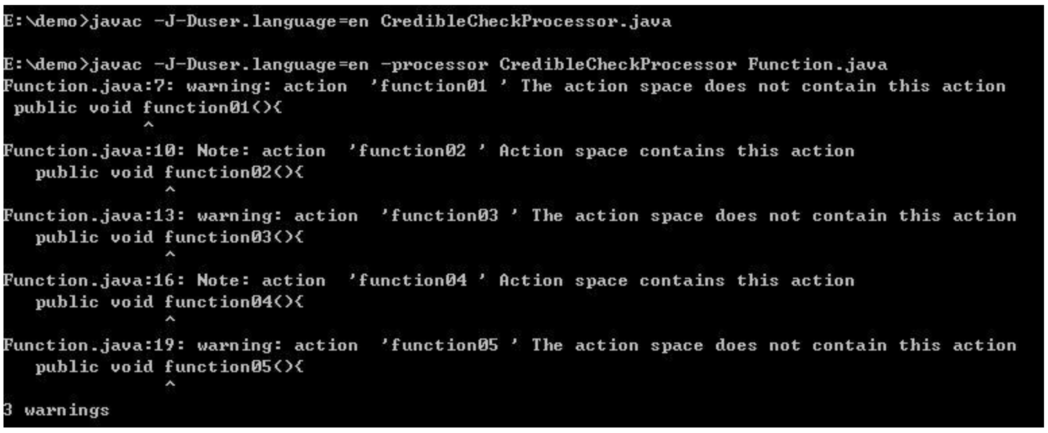
```
<ActionSpace>     <!--Action space-->
   <ActionList>     <!--List of basic actions-->
     <Action>
       <ActionId>1001</ActionId>     <!--Basic action number-->
       <ActionName>function02</ActionName>     <!--The name of the basic action-->
     </Action>

     <Action>
       <ActionId>1002</ActionId>     <!--Basic action number-->
       <ActionName>function04</ActionName>     <!--The name of the basic action-->
     </Action>
     ......
   </ActionList>
</ActionSpace>
```

Fig. 4. The representation of action space.

(2) Fig. 5 shows the results of secure compilation

"CredibleCheckProcessor.java" is a secure compilation plugin. In the compilation result, the prompt "Note" indicates the basic action contained in the action space; the prompt "warning" indicates the basic action that is not included in the action space.



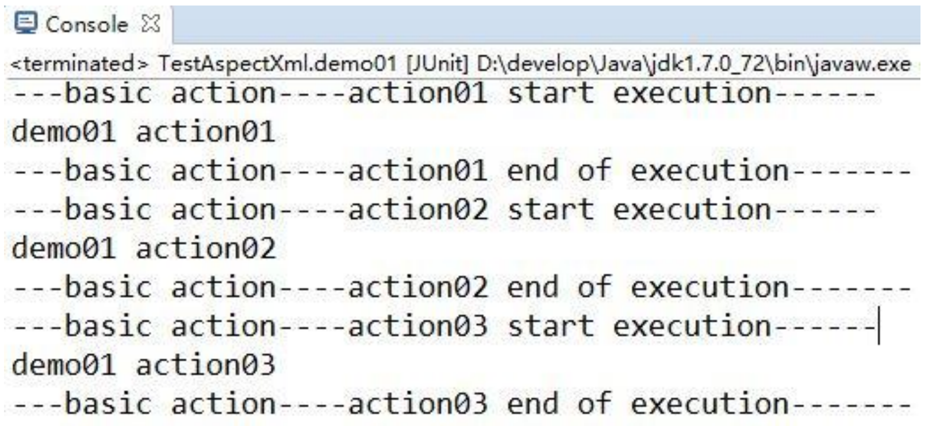
```
E:\demo>javac -J-Duser.language=en CredibleCheckProcessor.java

E:\demo>javac -J-Duser.language=en -processor CredibleCheckProcessor Function.java
Function.java:7: warning: action  'function01 ' The action space does not contain this action
 public void function01(){
            ^
Function.java:10: Note: action  'function02 ' Action space contains this action
   public void function02(){
              ^
Function.java:13: warning: action  'function03 ' The action space does not contain this action
   public void function03(){
              ^
Function.java:16: Note: action  'function04 ' Action space contains this action
   public void function04(){
              ^
Function.java:19: warning: action  'function05 ' The action space does not contain this action
   public void function05(){
              ^
3 warnings
```

Fig. 5. The results of secure compilation.

5.2 Testing the feasibility of dynamic monitoring

Fig. 6 shows the results of the monitoring, which shows the order in which the basic actions are executed.



```
<terminated> TestAspectXml.demo01 [JUnit] D:\develop\Java\jdk1.7.0_72\bin\javaw.exe
---basic action----action01 start execution------
demo01 action01
---basic action----action01 end of execution-------
---basic action----action02 start execution------
demo01 action02
---basic action----action02 end of execution-------
---basic action----action03 start execution------
demo01 action03
---basic action----action03 end of execution-------
```

Fig. 6. The results of the monitoring.

6. Results and Discussion

In the previous research on software credibility, most people conducted statistics and analysis on as much historical software behavior as possible, and used historical behavior as the expected behavior of the software, the evaluation results were finally given in the form of credibility grade. This software credibility evaluation method, which gives the evaluation results at the credibility level, has a reference significance to the software credibility evaluation to a certain extent. However, this evaluation method also has some problems and limitations. First of all, the historical behavior of software is taken as the expected behavior of software, which is an inductive method, which cannot guarantee the completeness of the expected behavior. Secondly, the evaluation results are given in the form of credibility level, there is a certain degree of ambiguity, the degree of software credibility cannot be intuitively displayed, and the credibility of software behaviors at the same level cannot be compared. Finally, there is a certain subjectivity in the evaluation process and the division of the credibility level, which cannot well describe the objective reality of the software.

The software credibility evaluation method based on fuzzy mathematics theory proposed in this paper solves the above problems to a certain extent. The software credibility evaluation method based on fuzzy mathematics theory is a quantitative evaluation, which uses behavior statement as the expected behavior of software, and the acquisition process of behavior statement is a deductive method. The deductive method has strict logic and can improve the accuracy of behavior statement. Secondly, fuzzy sets are used to construct behavior statements and actual behaviors in a quantified form, which improves the accuracy of software behavior characterization. Finally, the construction of membership function in fuzzy set has some objectivity, so it can give the credibility of individual software behavior and the credibility of the whole software relatively objectively. Through the above comparative analysis, Table 1 summarizes the characteristics of the two credibility evaluation methods. It is found that the software credibility evaluation method based on fuzzy mathematics theory improves the accuracy of the evaluation results to a certain extent.

Table 1

Characteristics and comparison of credibility evaluation methods

Name / Characteristic	Credibility Evaluation Method of Software Based on Fuzzy Mathematical Theory	Credibility Evaluation Method of Software Based on Credibility Level
Evaluation Results	quantitative	qualitative
How to Get Expected Behavior	deduction	induction
Objectivity	strong	weak
Comparison of The Same Credibility Level	easy	difficult
Scope of Application	can be promoted	relative limitations

7. Conclusion

It is the core content of software credibility fundamental research to evaluate the credibility of software [12-14]. Different metrics and methods have an impact on the accuracy of the assessment results. In the current research, most of them use the historical behavior of software as the basis for judging. But using historical behavior as the expected behavior is one-sided and static, as it is an inductive method, which differs from the deductive method. Inductive methods cannot guarantee the completeness of expected results, so it is impossible to fully determine the credibility of the software. In addition, most of the judgement results are determined by a credible level, it is a qualitative analysis, and cannot give an intuitive degree of credibility with quantitative results, thus resulting in a certain fuzziness. This paper took the behavior declaration as the expected behavior of the software, based on the idea of "words and deeds" and the research results of fuzzy mathematics theory, and gave a calculation method for evaluating the credibility of software. It produced the evaluation results in quantitative form, which improved the accuracy of the evaluation results to a certain extent.

In the future work, the construction of membership functions corresponding to fuzzy sets will be the focus of our research. Because when using fuzzy sets to characterize software behaviors, the more accurate the membership function corresponding to the basic action is, the more fully characterizing the software behavior, and the higher the accuracy of the measurement result.

Acknowledgement

The authors would like to thank the reviewers for their detailed reviews and constructive comments, which have helped improve the quality of this paper.

R E F E R E N C E S

[1]. *J. Zhou, M. Zhang*, "Survey on Trustworthy Software Evaluation", Application Research of Computers, **vol. 29**, no. 10, 2012, pp. 3610-3613.
[2]. *C. Shen, Zhang H., H. Wang*, "Research and Development of Trusted Computing", Chinese Science: Information Science, **vol. 40**, no. 2, 2010, pp. 139-156.
[3]. *L. Zhuang, M. Cai, C. Li*, "Trusted Dynamic Measurement Based on Software Behavior", Journal of Wuhan University (Natural Science Edition), **vol. 56**, no. 2, 2010, pp. 133-137(Ch).
[4]. *S. Ding, S. Yang*, "Research on Evaluation Index System of Trusted Software", Proc of the 4th International Conference on WiCOM, 2008, pp. 1-4.
[5]. *X. Yu, G. Jiang, P. Wang, H. Song, K. Wang, Y. Liang*, "Research on Application's Credibility Verification Based on ABD", Wuhan University Journal of Natural Sciences, **vol. 21**, no. 1, 2016, pp. 63-68.
[6]. *R. Xiao, X. Yu*, "Credibility Verification Method and Calculation Based on Application Behavior Declaration", Computer Systems Applications, **vol. 27**, no. 11, 2018, pp. 17-26.
[7]. *Y. Liu, R. Du, J. Feng, J. Tian*, "Trust Model of Software Behaviors Based on Check Point Risk Assessment", Journal of Xidian University, **vol. 39**, no. 1, 2012, pp. 179-184+190.
[8]. *E. Amoroso, C. Taylor, J. Watson*, "A Process-oriented Methodology for Assessing and Improving Software Trustworthiness", Proc of the 2nd ACM Conference on Computer and Communications Security, New York: ACM, 1994, pp. 39-50.
[9]. *M. Li, X. Zhou, J. Wang*, "A Perspective of Software Trustworthiness Based on Distrustable Factors", Proc IEEE International Conference on Networking, Sensing and Control, 2009, pp. 873-878.
[10]. *H. Wei, X. Chen, C. Wang*, "User Behavior Analyses Based on Network Data Stream Scenario", 2012 IEEE 14th International Conference on Communication Conference, Piscataway N J; IEEE Press, 2012, pp. 1017-1021.
[11]. *D. Sun, J. Li, Z. Wang*, "A Method of Dynamic Trusted Researching of Software Behavior and Its Trusted elements", Network Security Technology and Application, no. 4, 2013, pp. 14-17.
[12]. *D. Wang, Q. Wang, J. He*, "Software Process Reliability Model Based on Evidence and Evaluation Method", Journal of Software, **vol. 28**, no. 7, 2017, pp. 1713-1731.
[13]. *X. Jia, W. Zhang, A. Liu*, "Research and Application of Hierarchical Software Trustworthiness Evaluation Model", Computer Science, **vol. 44**, no. 4, 2017, pp. 169-172.
[14]. *X. Luo, Z. Tang, Y. Zhao*, "Dynamic Software Reliability Assessment Based on Markov Chain", Application Research of Computers, **vol. 32**, no. 8, 2015, pp. 2401-2405.
[15]. *LI Zhen, YU Xue-Jun*, Optimization of Cloud Service Trusted Evaluation Model Based on Sliding Window, Computer Systems & Applications, **vol. 28**, no.7,2019, pp. 35-43.
[16]. *ZHANG Fan, XU Ming-Di, CHAO Han-Chieh*, Real-time Trust Measurement of Software: Behavior Trust Analysis Approach Based on Noninterference, Journal of Software, **vol. 30**, no.8, 2019, pp. 2268-2286.
[17]. *WANG De-Xin, WANG Qing*, Trustworthiness Evidence Supporting Evaluation of Software Process Trustworthiness, Journal of Software, **vol. 29**, no.11,2018, pp. 3412-3434.
[18]. *YANG Xi, LUO Ping, GUL Jabeen*, The Concept Model of Software Trustworthiness Based on Trust-Theory of Sociology, Acta Electronica Sinica, **vol.47**, no.11, 2019, pp. 2344-2353.
[19]. *Chen Qianqian, Xu Lixing, Gong Bin*, Software Trustworthy Evaluation for High-conflict Evidence Reasoning, Ordnance Industry Automation, **vol.38**, no.2, 2019, pp. 55-59.
[20]. *TIAN Junfeng, GUO Yuhui*, Software behavior trust forecast model based on check point scene information, Journal on Communications, **vol.39**, no.9, 2018, pp. 147-158.