A Method for Software Credibility Assessment Based on Runtime Behavior

Zhiming Wang, Changyun Li, Shenglong Hu, Huang Tan
School of Computer and Communication, Hunan University of Technology, Zhuzhou 412008, China

ABSTRACT
The software is essentially replacing human beings to carry out some certain behavior and software credibility is mainly represented by its credible behavior, so it will be more direct and reliable to assess software credibility according to the information of software runtime behavior. The concept of behavior pattern is abstracted from the software observable behavior and the pattern’s description language is designed. A behavior pattern composed of a trace pattern and a constraint reflects the essential character of software runtime behavior. A bayes model to assess credibility is presented and the model is composed of nodes of behavior pattern matching levels and nodes of credibility attributes. Since the model implicates the condition relation of behavior patterns and credibility attributes, software credibility can be inferred from the uncertainty software runtime behavior. Our metric has been evaluated by an electronic transaction system and its validity is proved.

INTRODUCTION
How to effectively make use of the information of software runtime to assess the software credibility has attracted more and more attention in software engineering. The reasons are the following: (1) Software credibility means the software runtime behavior and its results are always in line with expectations of us, and the software system can still provide persistent service, though it has been interfered. (2) Because of the increasing complexity and the openness of software, it enable software credibility not only depends on good design and implementation of the software, but also needs to be adjusted and controlled in a real running process. Therefore, it will be more direct and reliable to assess software credibility according to the information of software runtime rather than reputation, recommendation, subjective experience and information of software design. The information of software runtime includes software runtime behavior, possession of resource and some other things (such as performance, security, etc). Since the software is essentially replacing human beings to carry out some certain behavior, and software credibility is mainly represented by its credible behavior, the information of software behavior is the core information in its runtime, which is also an important proof of assessing software credibility. The software runtime behavior can be divided into observable behavior and unobservable behavior. The interaction between the software system boundary and external environment is obviously observable. Modern software systems are generally assembled by a number of components. The interaction among components may be available by using some certain monitoring technologies and tools, even if the components are the third party, therefore, it is also observable. For the calculation of the internal components, it is difficult, or even unable to monitor and collect, and it is considered as unobservable. The proof of assessing software credibility focuses on observable behavior. According to the observable behavior, we can infer the credibility of software. For example, when the software runs with errors or acts unreliably, its observable behavior often shows certain characteristic. In Web Services applications, when the destination service does not exist, the calling towards it will result in an exception interaction event. The order for digital electronic resources, campus network users can download articles legally, but if they use tools to download a large number of articles, and their behavior may be regarded as untrustworthy. The former observable behavior can be characterized as an exception interaction event and the latter behavior can be characterized as a sequence of special events which consist of a series of interactive events. We define the constraint with which the special events, the sequence of special events and the events sequence may comply, as behavior pattern. Since the behavior pattern reflects the features of software observable behavior, to some extent, it also reflects the software credibility. How to describe the behavior pattern, how to match the behavior pattern for software runtime observable behavior, and how to effectively use the behavior pattern to assess the software credibility, all these become critical for software credibility assessment based on runtime behavior.

In this paper, our main contributions are the following. In section 1, we introduce the software credibility evaluation framework based on software runtime behavior. In section 2, we introduce the description of behavior patterns and the matching algorithms. In section 3, we discuss a metrics for software credibility by using behavior patterns. In section 4, we analyze the metrics for software credibility in a specific application. In the end, we summarize our findings and the conclusions of our work.

The Whole Process of Software Credibility Evaluation Based on Runtime Behavior
The basic idea of software credibility evaluation based on runtime behavior is assessing software credibility according to the behavior patterns showed in software running process, the whole model showed in Figure 1. By monitoring spots in application software, behavior monitoring, and services monitoring, so as to collect and organize the software observable behavior in its running process. Referring to behavior pattern library, we can carry out pattern matching for the observable behavior. According to the matching result and credibility attribute evaluation model, we can assess the software security, reliability and other credibility attributes.

Fig.1 The process of software credibility evaluation based
The task of monitoring service is to collect software observable behavior in its running process. The technology that we can insert or program the monitoring spots in application software has become more mature, just as package, capture, AOP, reflection etc. These technologies can be effective in monitoring and collecting interactive events between system and external, as well as those events among system internal components. Another task of monitoring service is how to organize these interactive events. Depending on different application purposes, the organizing method can take tasks, event source components or event target components as clues, ultimately to produce a sequence of events. Behavior patterns should be designed and described in pattern language by application field experts in advance, and stored in behavior patterns library. Taking the interactive events which are collected and organized by monitoring services as the source, behavior patterns in pattern library as the reference, so as to do pattern match, and calculate the match value and see event sequence match with which behavior patterns. Credibility attribute evaluation model is bayesian network model. In the model, some nodes represent pattern matching rate and the other nodes represent credibility attributes. Through the pattern matching rate and bayesian network probability calculation, we can find each rank probability of credibility attribute, and then take the maximum probability rank as evaluation value of credibility attribute.

Description and Matching of Behavior Pattern

Behavior Pattern Language BPL

Behavior pattern is the events sequence which has some recurring features in software observable behavior. And it’s the external indicators of software credibility in runtime. Therefore, Description of behavior pattern focuses on the following 3 aspects:

(1) abstract. Behavior pattern is an abstract of software observable behavior and a summary of a kind of interactive events sequences with common characteristics.

(2) segment. Behavior pattern reflects a segment characteristic of observable behavior rather than the whole characteristics of observable behavior in the running process of software.

(3) constraint. Behavior pattern is not only the characteristics of events sequence but also a constraint of some behavior, just as constraint of event parameters, sequence relations between events etc.

Based on the above-mentioned, we designed a behavior pattern language BPL. BPL’s BNF syntax is as below:

```
<Behavior Pattern Set>::=
"Pattern Set"<IDENTIFIER> { Event Definition }

( "," Behavior Pattern Definition ) "End"
//appointed pattern marked by alphabetic string begin with capital English letter;

Event Definition::=
"Event"<IDENTIFIER>
= "Update"<Variable>
| "StartM" <Method Definition > |
| "EndM" <Method Definition > |

//Appointed event marked by small English letter;
Method Definition::= Method"<IDENTIFIER>"("Parameter List"
"|
| [], From" Component ] [", To" Component "]"
| [ Other relevant information, Like thread of method execution ]

Behavior Pattern Definition ::= "Pattern" <IDENTIFIER>
```

Statement ::= <Trace Pattern> [Constraint]
Trace Pattern ::= "Trace Pattern" < It is described by regular expressions whose event logo is letter>
Constraint ::= "Constraint" (<Simple constraint> | <constraint> && <constraint> | <constraint> || <constraint>)
Simple constraint::= <Boolean Expression>

A behavior pattern set includes several event definitions and behavior pattern definitions. There are two types of events: variable update(Update), method invocation(StartM,EndM).

The update followed by observable variable to the monitoring service, such as process variable in the workflow system. When the variable changes, the update is triggered. Method invocation which reflects interaction of components has two cases: start a method invocation (StartM), and method return (EndM). The method prototype of Method Definition should be identical with the application components prototype, namely the same method name and the same parameters list. In addition, to Method Definition, there is optional terms both for method invocation (From Component) and method invocation (To Component), as well as other relevant information options, this enables the definition of the event for the component approach with the ability of different levels of abstraction. For example, when applications need to pay attention to many method calls of the same name but from different source components, we can fill From Component option in the definition of method. When there is no need to concern about the different components, the From Component option can be ignored. Similarly, concerning about the implementation of the same name components method whether in the same thread, we can fill the thread information of method in the “other relevant information” option.

Behavior pattern is mainly made up of Trace Pattern. The trace refers to the fragment of sequence events. Trace Pattern is an abstract and generalization of trace. In order to figure the commonness of similar traces and statistical features, trace pattern is described by regular expressions whose event logo is letter.

For example, trace pattern "(a * b)" means a number of random events begin with event a, and end with event b, such as “ab”, “acdeab” and “afcb”. Trace pattern “(ab)[2,6]” means events a and b alternate 2-6 times, such as “abab”, “abababab” and so on.

BNF grammar for trace patterns:
Trace Pattern ::= “Trace Pattern” < The event logo for the letter regular expressions >

Table 1 is the Regular expression syntax for trace patterns.

Besides trace pattern, the behavior pattern may also contain constraint. Constraint depends on trace pattern, expressed by logical expressions. It describes the constraint relation between events in trace by the expression of trace pattern, and the event parameters should be complied with. For example, "valueof e.arg (1)> 6" means that the first parameter of e events must be greater than 6. In order to better express constraint pattern, we pre-defined a number of functions, shown in Table 1.
Table 1. The Regular expression syntax for trace patterns

<table>
<thead>
<tr>
<th>character</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>Matching with the preceding subexpression no or multiplying. For example, so * can match the &quot;z&quot; and &quot;fooood&quot;. * is equivalent to [0].</td>
</tr>
<tr>
<td>+</td>
<td>Matching with the preceding subexpression one or multiplying. For example, 'zo +' can match the &quot;zo&quot; and &quot;fooood&quot;, but can not match the &quot;z&quot;, + is equivalent to [1].</td>
</tr>
<tr>
<td>?</td>
<td>Matching with the preceding subexpression no or one. For example, &quot;do (es)?&quot; can match &quot;do&quot; or the &quot;do &quot; in the &quot;does&quot;. ? is equivalent to [0,1]. When this character followed by any other qualifier (*, +, ?, [N], [n], (\in), (m)), the matching pattern is non-greedy. Non-greedy mode is that it will match with the search string as little as possible, while the default greedy pattern will match with the search string as much as possible. For example, for the string &quot;ooood&quot;, 'o +?' will match a single 'o', while 'o +' will match all these 'o'.</td>
</tr>
<tr>
<td>[n]</td>
<td>n is a non-negative integer. Matching to the n times. For example, 'o [2]' cannot match the 'o' in &quot;Bob&quot; but to the two &quot;o&quot; in &quot;food&quot;.</td>
</tr>
<tr>
<td>[n,]</td>
<td>n is a non-negative integer. Meaning that it will match n times at least. For example, the 'o' 'o [2]' can not match the &quot;Bob&quot;, but it can match all the &quot;o&quot; in &quot;fooood&quot;. 'o [1]' is equivalent to 'o +'. 'o [0]' is equivalent to 'o *'.</td>
</tr>
<tr>
<td>[n,m]</td>
<td>m and n are non-negative integer, n =&lt; m means that it will match n times at least and m times at most. For example, 'o [1,3]' will match to the first three 'o' in &quot;fooood&quot;. 'o [0,1]' is equivalent to 'o?'. Please note that there is no spaces between the comma and the numbers.</td>
</tr>
<tr>
<td>[s]</td>
<td>s is a character. For example, 'z' matches &quot;z&quot; or &quot;food&quot;, 's' matches &quot;ood&quot;, 'z' matches &quot;ood&quot;, 'or' matches &quot;.food&quot;.</td>
</tr>
<tr>
<td>[xyz]</td>
<td>The set of characters. Matching with any contained character. For example, '[abc]' &quot;plain&quot; can match the 'a'.</td>
</tr>
<tr>
<td>[^xyz]</td>
<td>Negative character set. Matching with any characters which is not contained. For example, '[^abc]' will match the 'plain' in the 'p'.</td>
</tr>
</tbody>
</table>

Table 2 Pattern constraint function

<table>
<thead>
<tr>
<th>Function name</th>
<th>Meaning of Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event((t,e):c</td>
<td>Taking out the No. i event in trace t</td>
</tr>
<tr>
<td>Eventindex(t,e):i</td>
<td>Taking out the position where the No. i event appeared</td>
</tr>
<tr>
<td>Events(t,e):i</td>
<td>Concluding times of e event, e by * means the total amount of event</td>
</tr>
<tr>
<td>Valueof(e,arg(i)):v</td>
<td>Value of the No.i in e event</td>
</tr>
<tr>
<td>Typeof(e,arg(i)):s</td>
<td>Type of the No. i parameter in e event</td>
</tr>
<tr>
<td>Timeof(e,t)</td>
<td>Time of the e event happened.</td>
</tr>
</tbody>
</table>

t - trace, i - integer, e - events, v - type variable parameter, s -string

Behavior Pattern Calculation

Monitoring service collects software interaction events, and organizes them as event trace. Behavior pattern match is referred by some certain behavior pattern, to search all the nonrecurring son traces in event trace which comply with the pattern. To describe the degree of event trace behavior pattern matching, we introduce into match degree. To a certain event trace, match degree means the rate which events in the sun trace to total events in pattern. Setting the event trace as T; pattern as P, the sub-trace in which E matches P, as T1, T2, ...Tk, and the number of events counted by function Events(t) in table 1, M-T’s match degree in P, formula as

\[ M = \frac{\sum_{i=1}^{k} \text{Events}(T_i, \cdot^*)}{\text{Events}(T)} \]  

One purpose of behavior pattern match is calculating the match degree. The overall process of algorithm is as following: The first step is to find and save all the sub-tracks in line with the behavior patterns and nonrecurring; second step is to test whether or not each sub-trace bound to meet the behavioral pattern constraint, and remove the subtracks which are not compatible with the constraint; the third step is to calculate the matching degree in accordance with the formula (1). Due to events such as calling method, mapping for the letters of the alphabet, trace events can be expressed by alphabet letter strings formally, and trace patterns are in regular expressions, making the first step transform to find substring in line with the regular expressions. This is a classic problem with the existence of sophisticated algorithms and tools. As the second step of the algorithm, due to constraints that are in logical expressions, detecting whether or not each sub-track to meet the constraints is essentially a first-order logic formula satisfiability problem, and the problem is classic. Described by pseudo-code, the algorithm is as following:

1. Begin
2. \( 0 \Rightarrow i \)
3. \( 0 \Rightarrow sub\_sum \)
4. e Set variable e to the alphabetic string corresponding to trace pattern
5. get event trace and constraints by setting variable t and c respectively
6. getEventTraces() => t
7. getConstraints() => c
8. According to the event mapping library of the behavior pattern, mapping from the event trace to the corresponding alphabetic string
9. MapStr() => ts
10. Seeking those nonrecurring son trace match for the trace pattern, storing them into temporary group Result
11. Match(ts,e) => result
12. While (i < result.length){
13. Check this son trace whether it is compatible with constraint of behavior pattern, if so, conclude the total amount of the event in this son trace.
14. If Check(c, result [i]) then
15. \( sub\_sum + \text{Events}(\text{result}[i++]) \Rightarrow sub\_sum \)
16. end if
17. }
18. Return to pattern match
19. return (sub\_sum/Events(t))
20. End

The above algorithm assumes the length of sequence as n, the length of track mode as m. Then the frequency of GetAllEvent function as n,
A METHOD FOR SOFTWARE CREDIBILITY ASSESSMENT BASED ON RUNTIME BEHAVIOR
Zhijing Wang, Changyun Li, Shenglong Hu, Huang Tan

frequency of getENames function as n, frequency of Split function as m, time frequency of Match () function as n * m, therefore, the algorithm time complexity is (n + n + m + m * n = m * n +2 n + m = O (m * n))

MODEL OF CREDIBILITY ASSESSMENT AND CALCULATION OF CREDIBILITY

Bayesian Model of Credibility Assessment

Assessment model establish the mapping relation between evidence of credibility and software credibility. The observable behavior of software reflects the degree of software credibility, and its characteristics can be reflected through the behavior patterns which the observable behavior complied, therefore, the assessment of credibility can be measured by the behavior pattern which the software observable behavior matched. However, sometimes, it’s uncertain for the relationship between behavior pattern and software credibility. For example, in the digital subscription of electronic resources, users download a large number of articles in a short period of time, they may pursue their own interest, but it may indeed be in order to their study and research needs. So the assessment model should have the ability of expression and reasoning for uncertain knowledge.

Bayesian network, also known as credibility network, is one of the most effective theory in the field of uncertain knowledge representation and reasoning (Friedman, Linial, 2007). A Bayesian network is a DAG (Directed Acyclic Graph, DAG). The node stand for random variables, the directional edge between nodes represents the relationship among nodes. The conditional probability represents strength of relationship, and information of no-father nodes is expressed by a priori probability. Node variable can be the abstract of any question, such as: the value of test, observed phenomena, consultation, etc. Bayesian Networks applied to expressing and analysing those uncertainty and probability events, and it can be used for decision-making which the conditionally dependent on a variety of factors, to make reasoning from those incomplete, inaccurate or uncertain knowledge or information . Therefore, we use Bayesian networks to build the assessment model of credibility.

Bayesian model of credibility assessment is shown in Figure 2. In Bayesian model of credibility assessment, there are two types of the node variable: pattern-matching level and the software attributes of credibility. Pattern-matching level is to divide software observable behavior into a number of levels N, by whatever match degree they comply with behavior pattern, and number the levels from high to low with the integer variables i, i ≤ N. They represent the matching degree ranging from low to high order of [0,1 / N], [1 / N, 2 / N] ... [(N-1) / N, 1]. What range the match degree fall in, what matching level it is. In addition to pattern-matching level, the other node type is the software attributes of credibility. Due to the overall credibility of software with characteristics of ambiguity and general, it is not conducive to be measured and quantified. In practice applications, according to general requirements and functions, overall credibility is divide into a number of attributes, then assess each credibility attribute by using credible evidence so as to avoid generality and ambiguity in assessment of the overall credibility. In order to effectively assess the attributes of credibility, it needs to be divided into a number of trust levels for different requirement. In the credible assessment model of Bayesian, to build a Bayesian network for each software trusted property, in which the parent nodes represent the software reliability attributes, child nodes for the pattern-matching level.

Fig.2. Bayesian model of credibility assessment

Construction of Bayesian Model

Construction of Bayesian network is a much complicated task that requires the participation of knowledge engineers and experts in the field. First of all, the experts design the behavior pattern based on knowledge of application field, and determine the model collection of assessing evidence for each credibility attribute in qualitative analysis of the possibility of conditional correlation between behavior pattern and assessing credibility attributes. Then for each credibility attribute, taking credibility attribute as the root node, relevant pattern as child node, we establish credibility attribute structure of Bayesian network model.

Next, all the parameters of Bayesian network model must be determined, that is, a prior probability of attributes, pattern-matching level of prior probability and its conditional probability. We determine the parameters through the training of sufficient sample data. When application run under different typical scenario, for each running, the experts in the field determine the level of credibility attributes. At the same time, from the collection of software observable behavior by monitoring service, according to 2.2, the match degree associated with the credibility attributes of mode and even the match-class can be calculated. After each running, the count of running plus 1, which level the value of the node fall on, the count of corresponding levels adds 1, and the others remain unchanged. In order to calculate credibility attribute levels under multi-mode, we need to save the times of more than two different nodes level appearing at the same time. Application programme is presumed to run N times totally, Ci said that attributes i of level C, Pj that match level j of pattern P, | Ci | that the times of attributes i of level C appearing, | Paj | that the times of match level j of pattern Pa appearing, | Ci ∩ Paj ∩ Pbk | that the times of attributes i of level C, match level j of pattern Pa and match level k of pattern Pb appearing at the same time. The use of p (Ci), p (Paj), p (Pbk) respectively represent the priori probability of attributes i of level C, match level j of pattern Pa and match level k of pattern Pb. The priori probability of attribute C is:

p(Ci) = | Ci | / N \sum_{i=1}^{N} p(Ci) = 1 \quad (2)

Similarly, the priori probability of pattern matching level is:

p(Paj) = | Paj | / N \sum_{j=1}^{N} p(Paj) = 1 \quad (3)

p(Pbk) = | Pbk | / N \sum_{k=1}^{N} p(Pbk) = 1 \quad (4)

In addition to calculating the prior probability, it is also necessary to calculate the conditional probability of nodes which is in each level of pattern match. Taking p (Paj / Ci) conditional probability for example, it means the probability of credibility attribute the root node C which is in the Ci-level and pattern Pa in the Paj level:
A METHOD FOR SOFTWARE CREDIBILITY ASSESSMENT BASED ON RUNTIME BEHAVIOR
Zhijing Wang, Changyun Li, Shenglong Hu, Huang Tan

Calculating of the Credibility Attribute Level

With the priori probability and the conditional probability of all leaf nodes, we can assess the probability of credibility attribute level under the conditions of certain behavior pattern matching level. First, checking the assessment of credibility attribute under the conditions of single pattern-matching level, use of the Bayesian formula to calculate

\[ p(Ci | Pa) = \frac{p(Pa | Ci) p(Ci)}{p(Pa)} = \frac{\frac{p(Ci)}{|Ci| N} p(Ci) p(Pa | Ci)}{\frac{p(Pa)}{|Pa| N}} \]

(5)

Using Bayesian formula, we can also assess the probability of credibility attribute level under the conditions of multi-level pattern matching. For example, we assume that all pattern matching level is divided into four classes, namely, the exact match (match-class 1), the comparative match (match-class 2), the basic match (match-class 3), and the non-match (match-class 4); the credibility attribute trust level is also divided into four classes, namely, the exact trust(trust level 1), the comparative trust (trust level 2), the basic trust (trust level 3) and non-trust (trust level 4). Then use (C3/Pa2, Pb3), we can calculate the probability of credibility attribute C in "basic trust" (trust level 3) under the conditions of pattern Pa as "comparative match" (match-class 2), pattern Pb as "basic matching" (match-class 3). See formula (6):

\[ p(Ci | Pa, Pb) = \frac{p(Pa | Ci) p(Pb | Ci) p(Ci)}{p(Pa, Pb)} = \frac{\frac{p(Ci)}{|Ci| N} p(Ci) p(Pa | Ci) p(Pb | Ci)}{\frac{p(Pa, Pb)}{|Pa, Pb| N}} \]

(6)

In the same pattern-matching level, through the probability calculation of Bayesian network, we can calculate the probability of the credibility attribute in each level, and set the level of the greatest probability as its assessment level.

EXAMPLES AND ANALYSIS OF ELECTRONIC TRANSACTIONS

We verify the validity of the method proposed in this paper on an electronic trading system. Trust-purchase network is a C2C electronic trading platform, anyone can register online, purchase goods, and open a shop. As parties to the transaction general are unfamiliar individuals and there is no reliable machine to verify their identity information, to identify each other's credibility in accordance with the transaction behavior has become a demand from trading side. Buyers need to identify the credibility index of the seller's sales in order to decide whether to order and remit; sellers need to distinguish the credibility index of consume to decide whether to transact, ship goods and make discount. Through the observation and analysis of a large number of transaction processes, five kinds of patterns behavior are set as following:

Mode 1 (Pa): buyer search for goods (search) - buyer order (order) – buyer advance (imprest) - buyer confirm receiving (accept), trade mode is expressed as: (srp) (1, ) * a;

Mode 2 (Pb): buyer order (order) - seller to confirm, modify the price (modification) – buyer advance(imprest) - seller deliver (consignment) - buyers receive and pay (payment), trade mode is expressed as: (r * m * p * e * c) (1,);

Mode 3 (Pc): seller view orders (examine) - seller amend the price (modification) – seller check payment information (inquire) – seller deliver (consignment), trade mode is expressed as: em * qc;

Mode 4 (Pd): buyer non-payment orders in many times (more than 2 times), trade mode is expressed as: r (2 ,) * c

Mode 5 (Pe): buyer orders for many times (more than 2 times), advances, the seller does not ship, trade mode is expressed as: (srp) (2 ,) * c;

On the above 5 kinds of mode, Pa, Pb, Pd, Pb, Pc, Pe are related with consumer credibility index (Ca), and Pb, Pc, Pe are related with seller credibility index (Cb). Assessment model structure of Bayesian network is shown in Figure 3.

Assuming pattern-matching level, the credibility index of consume, and the credibility index of sale are all 4 levels. In order to determine the parameters of Bayesian network, under the premise of the known buyer’s credibility index of consume and the seller’s credibility index of sale, we monitored 27 times electronic transactions process in trust-purchase network, and we take the first 20 times as the sample data , latter 7 times as the test data, shown in table 3.

According to the formula (2)--(5), the parameters of Bayesian network a,b in figure 3 can be calculated, for example the probability of Pa when its match level is comparative match(level 2) is:

\[ p(Pa) = \frac{|Pa|}{N} = \frac{6}{20} \]

For Bayesian network a, the credibility index of consume as "comparative trust" (level 2), the conditional probability of Pa when its match level is "basic matching" is:

\[ p(Pa2/Ca2) = \frac{|Pa2 \cap Ca2|}{|Ca2|} = \frac{1}{6} \]

In Table 3, through the verification data, the accuracy of the calculated about consume credibility index reached 86% and the accuracy of the calculated about the sale credibility index reached 71%, which means the method in this paper is effective.
A METHOD FOR SOFTWARE CREDIBILITY ASSESSMENT BASED ON RUNTIME BEHAVIOR
Zhijing Wang, Changyun Li, Shenglong Hu, Huang Tan

Table 3 Sample data in trust-purchase network

<table>
<thead>
<tr>
<th>Number</th>
<th>Event trace in transaction</th>
<th>Pa</th>
<th>Pb</th>
<th>Pc</th>
<th>Pd</th>
<th>Pe</th>
<th>Ca</th>
<th>Cb</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>srspsrpemepcyca</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>srspsrpsrspapsrpar</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>srspsrpsrpsrrpar</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>srspsrpar</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>srspsrpamarrpqcy</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>pprsrprrpar</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>apersprrmrrpsrrpar</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>8</td>
<td>sparspsrrpqas</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td>apsprrpsypsr</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>10</td>
<td>sprppysrpe</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>11</td>
<td>paemsrprpac</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>12</td>
<td>1aemsersmsrpsrrpsrcasysqsrc</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>13</td>
<td>aremsrppagpsyr</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>14</td>
<td>sprrapsppmipisrncy</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>rsermaspsrrcay</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>16</td>
<td>pxxsrrrrsraay</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>17</td>
<td>parrrrsrrssrasa</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>18</td>
<td>parrrssrrarsrrasrcqcy</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>19</td>
<td>asrrrrasrrcramyasas</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>20</td>
<td>rrrrrarrrsraaa</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

Validation data

<table>
<thead>
<tr>
<th>Number</th>
<th>Event trace in transaction</th>
<th>Pa</th>
<th>Pd</th>
<th>Pe</th>
<th>Xa</th>
<th>Xb</th>
<th>Xc</th>
<th>Xd</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>srsprrppsa</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>1</td>
<td>(1)</td>
<td>(4)</td>
</tr>
<tr>
<td>2</td>
<td>krsrrxxsr</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>(1)</td>
<td>(2)</td>
</tr>
<tr>
<td>3</td>
<td>rmsrrpxspq</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>(3)</td>
<td>(1)</td>
</tr>
<tr>
<td>4</td>
<td>rmmsrrsrr</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>srpacarpemssqac</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>(2)</td>
<td>(3)</td>
</tr>
<tr>
<td>6</td>
<td>rmirsprssrmmapcy</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>(3)</td>
<td>(2)</td>
</tr>
<tr>
<td>7</td>
<td>srsprrspccaa</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>1</td>
<td>(1)</td>
<td>(2)</td>
</tr>
</tbody>
</table>

Note: the credibility index of consume and the credibility index of sale in bracket are calculated by Bayesian network

RELATED WORK AND SUMMARY

The credibility of the software has two sides: subjectivity and objectivity (Ding et al., 2011). The subjectivity of credibility focuses on user's subjective feelings for the software, and the objectivity of credibility focuses on objective ability of the software. Software credibility assessment is a process to evaluate the credibility degree of software, according to different concerns, the software credibility assessment can be divided into two categories: subjective assessment and objective assessment.

The evidence of subjective assessment comes from users' subjective feelings of the software, such as user's evaluation, experience, reputation and so on, and they pay much attention to the software whether it meets the evaluation of their expectation. Abdul-Rahman and Hailes introduced trust and reputation into the Trusted Computing Agent (Abdul-Rahman, Hailes, 2000). With this method, the trust comes from the Agent of itself which had direct and interactive experience and knowledge in the past, while the reputation comes from the comments and recommendations by other agents, using the method of weighted average that trust and reputation are combined into the final evaluation. T. Beth etc. (Junfeng, Ruling, 2011) proposed a credibility assessment model based on experience and probability statistics, and worked out the credibility speculation and integrated calculation formula by experience recommendation. Trust model proposed by Huimin W (Huimin et al., 2006, Wen et al., 2004) is also based on the recommendation trust model, and the basic idea is similar to the above-mentioned methods, but it has better performance. Yuan W constructed a subjective credibility model based on a fuzzy set theory, in which the language variables and the fuzzy logic are introduced into credibility description and assessment (Yuan et al., 2006). Contrapositive (verbal) the fuzziness and randomness of the subjective credibility, cloud model used in the credibility assessment in articles (Huimin et al., 2006, Haicheng, Ruchuan, 2008), so this formed a model of credibility assessment - the trust cloud model.

The objective assessment focuses on the calculation of credibility-related quality attributes (reliability, availability and security, etc.), and its evidence comes from related data collection in software development and operation, P. Herrmann (Herrmann, 2003) has developed a credibility assessment system, using the method based on runtime observation to collect valuable information for the measure of component security. Abdul-Rahman and other people (Abdul-Rahman, Hailes, 1998) proposed a distributed credibility measure model which is used for quantified assessment of the software behavior results. Peking University, starting form how to meet the QoS requirements of different users, proposed a description and evaluation framework which is suitable for supporting multi-dimensional credibility attributes (Zhou et al., 2005). Jøsang method (Jøsang, Knapskog, 1998) is the integration of subjective and objective assessment, introducing the concept of evidence space and the opinion space to describe and measure the credibility relation. Evidence represents the subjective attributes of credibility, and concept represents subjective experience of credibility.

In this paper, the method is based on software runtime information for assessment evidence. It is essentially an objectivity assessment method, and moreover possesses the characteristics of subjective assessment method. As the design of behavioral patterns and the process of constructing evaluation model, it needs the experience and knowledge of experts in the field. Compared to the past work, this method has two characteristics:

1. The credibility assessment will be more direct and reliable for the evidence of software runtime behavior information. The software is essentially replacing human beings to carry out some certainly behavior and software credibility is mainly represented by its credible behavior. Credibility assessment and the protection of assessment for the software in the period of its design will ultimately show software credibility in its run-time. The reputation, recommendation values, the subjective experience and the other evidences of subjective assessment, in essence, are indirectly reflected by the results of software running behavior.

2. Bayesian network assessment model which is made up of those nodes of credibility attributes and pattern matching levels, can conduct the software credibility from uncertainty software running behavior. The assessment model includes not only the experience and knowledge of experts in the field, but also the analysis results of objective data, containing the conditional relationship between credibility attributes and behavior patterns. And the behavior pattern embodies the essential characteristics of software running behavior, as track patterns described the common of a similar event sequence, and statistical features, while constraint means behavior abidance.

Further research work: depending on the assessment results of multi-dimensional credibility attributes, we would assess the overall integrated credibility of software; we would study how to automatically or semi-automatically extract the commons from a large number of behaviors in order to form behavior pattern; based on matching and analysis of behavior patterns, we would study models and methods of software behavior prediction.
A METHOD FOR SOFTWARE CREDIBILITY ASSESSMENT BASED ON RUNTIME BEHAVIOR
Zhiting Wang, Changyun Li, Shenglong Hu, Huang Tan

ACKNOWLEDGMENT

This work is supported by the Natural Science Foundation of Hunan Province under Grant No. 11JJ4050, and the Education Department Foundation of Hunan Province under Grant No. 11W002, No. 11B039 and 11C0398.

REFERENCES


P. Herrmann(2003). "Trust-based protection of software component users and designers". Proceedings of the First International Conference of Trust Management (iTrust 2003), LNCS 2692, Crete, Greece, pp. 75 ~ 90.


