Software Behavior Modeling Based on Invariant Constraints

Cheng Peng
School of Information Science and Engineering, Central South University, ChangSha 410083, China；
College of Computer and Communication, Hunan University of Technology, ZhuZhou 412008, China
E-mail: doc_pen@126.com，mjfok@qq.com

Lu-ming Yang [1], Jun-feng Man [2]
[1]School of Information Science and Engineering, Central South University, ChangSha 410083, China；
[2]College of Computer and Communication, Hunan University of Technology, ZhuZhou 412008, China

***Abstract*—Modeling the networked software interactive behavior is the basis of understanding its internal mechanism and the running rules .The software interactive behavior log files are firstly collected by monitoring, and then the invariant constraints are mined from it, finally, a dynamic model learned from the finite state machine is presented. In this model, not only thought over the situation that the parallel partly ordered event sequence generated by the networked software interaction, but the interplay between data values and components interactions are also considered, and the event are satisfied the invariant constraints. In order to ensure certainty and completeness of the model, the method of merging equivalent states in the divided sub-diagram is proposed. The corresponding algorithms are also designed. Meanwhile, the effectiveness and feasibility of the proposed method are validated through experiments.**

***Index Terms*—networked software, interactive behavior, finite state machine, invariant constrain**

I. INTRODUCTION

With the development and progress of the computer network and software technology, the calculation mode updates constantly, and the distributed computing, grid computing, cloud computing and transparent computing technology[1] appear one after another, the software system deployed above them also has a corresponding change, from the single version software to the internetware to the networked software[2], moreover, the operating environment of the software system also transforms from the closed, controllable environment to the open, cross-platform, dynamic, resources sharing, changeful, collaborative, difficult-controlled environment. Under this kind of multiple complicated conditions, how to realize the real-time tasks effectively, and provide stable and sustainable online services, and how to guarantee the security and reliability of the data is an urgent problem to be solved. The key problem not only lies in network protocol, improvement and innovation of the storage and computing mechanism, but also lies in the analysis and research of the constructivism and evolutionary [3] of the software system itself which operating in all kinds of frameworks, not only lies in the development phase of the software, but also in the running phase, the grasp of the vital signs and interactive behavior characteristics of the ontology elements of the software, which will help us improve the credibility[4]of the networked software from another angle, and improve the quality of the service that the user gets in the network environment.

The construction of software behavior model is the basis of software understanding and software analysis. At present, the software behavior modeling method is mainly divided into the following two kinds, one is the static analysis for the program source code, the other is the dynamic analysis for the log documents collected when they are running, and the former is called static modeling, while the latter is called dynamic modeling [5-7].These modeling methods provided us with very important idea and reference for our research. The static modeling method tries to analyze all possible paths formed during the source code invoking between procedures. In fact, it is limited by the copyright and the software decompilation, so the behavior model is difficult to have integrity and applicability, while the dynamic modeling method monitors and collects the log documents produced in the software interaction process, and then extracts the invoking relationship of the procedures, finally generates the behavior model. Although the dynamic modeling method is simple, practical, to be used extensively, there still exists some restrictions: ⅰ .The monitoring objects are limited to the single system software or the application software [8-10], but much less attention has paid on the large-scale networked software system. ⅱ . The monitoring and collecting format of the log document [11] has no standard, but disorganized content, much more redundancy, which brings difficulties for analysis. ⅲ The description of event is too simple, lack of considering the parameter delivery when the software

doi:10.4304/jsw.8.10.2455-2461

performs, resulting in control flow and data flow can not be taken into account together in the model, the attack detection for data semantic will be failed.iv. the model state setting is unreasonable, and the program pointer(PC) as state[12],although which can depict the invoking relationship when the procedure operates, it only contains address information, and can not judge the similar structure in the model, and also can not handle the merger of the state in the trace behavior. From the system resources invoking to resolve the object and used as state [13], is only suitable for the log document analysis of the operating system. v. the sequences of event that the model receives must be the orderly sequences[14-15]arranged by the timestamp, and the problem that the disorder or partially ordered event sequences produced during the parallel execution under the distributed environment[16-18] lost the reception ability has been discussed in our former work [19].vi.In addition, the model has a very strong dependence on the integrity of the log document, and the incompleteness of the log document will lead to the mining invariant[20-22] lost accuracy, while the invariant plays a key role in terms of refinement and abstraction for the model.

In view of this, the paper proposes a dynamic modeling method called IBDM (Interactive Behavioral Dynamic Modeling) which is based on the networked software interactive behavior, IBDM implants the monitoring probe into the networked software, sets the monitoring information format in advance, monitors the dynamic interaction behaviors inside the software in the long-term, and the information collected is stored as the formatted log documents. First of all, it uses the automatic generation of the regular expressions to divide the log documents into some trace sub-diagram, considering the semantic and universality for the state in the model, we adopts the fine-grained event to describe state, then, mines six kinds of invariant from the trace diagram, and it is divided according to the trace sub-diagram whether it meets the invariant constraint conditions or not, finally, IBDM mergers the equivalence structure between the trace sub-diagram which satisfies the invariant constraint, refining and abstracting the initial model, forming the final model. The theoretical analysis and experimental result has proved that IBDM modeling method has the accuracy and validity, which has laid the foundation for the follow-up behavior analysis.

II. Related Works

The work related to the log documents mining mainly focuses on detecting the relevance of the log document, abnormal and the performance test. The purpose of this work is not to find a precise model from the log document that any system produces. For example, the tool of SALSA [23] and Mocha [24] extracted and visualized the behavior of the node in the Hadoop' log document and conducted the performance test, its main task is to do Map Reduce description.J.Yang [25] mined and visualized the temporary attribute of the event trace, and adopted them to study the evolution of the program, but also not use these temporary attribute to infer the system model. With regard to the model inference, K-tail algorithm [26] is widely used, which iteratively merges the state of the sub-diagram with the same maximum depth of K, then generates a simplified finite state machine model. In general, using the K-tail algorithm can infer model for small or simple system without developers' supervising, once the complexity of the system increases, the accuracy of the model will greatly decreased, but its merger mechanism can be improved and perfected, will enlighten us much. In addition, there are many methods for inferring the system model usually by using the statute conditions that the developer written. Whittle and Schumann [27] infer a components state chart from the running scenery and attributes of the system, Damas [28] uses the interactive scene that the developer provides to induce the Labeled Transition System(LTS), later, by reducing the developer's participation to expand this method. However, these methods may produce excessive redundancy models, and need a lot of artificial input.

III.IBDM Model

A Model Definition

Before the definition of the IBDM model, we will firstly give the relevant definition.

Definition 1(event) event is the set of function sequence. When monitoring the online electronic shopping system, we preset the format of monitoring software interactive event as $e=(m,p,v,t,c)$, among them, $m\in M$, $p\in D_R$, $v\in D_V$, $t\in T$, $c\in C$,the given M is the finite set of the function, while R is the finite set of parameters, V is the finite set of variable, and T is the set of time stamp, also C is the set of all kinds classes in the system, then D_R is the domain of parameters and D_V is the domain of variable.

Definition2 (transaction) transaction $t_i=(e_{i1},e_{i2},\cdots,e_{in})$ are the sequence of real-time incidents when complete some kinds of operation that contains, it is the subset of the system trace. One time execution of the online electronic shopping may implement multiple functions, and it depends on customer's requirement, for example, the user can finish the meal ordering and book shopping on the internet in a shopping and so on, while they correspond with different transaction and each type of transaction contains the events which are not completely the same.

Definition 3(trace) trace is the finite set of the event sequence assigned by a fixed time called the timestamp, $bt=(m_1,p_1,v_1)\cdots(m_n,p_n,v_n)$, n is the number of the event instance, while m is the event instance belongs to different transaction.

Definition 4(Invariant) log contains rich information for our analysis, on one hand, the log records the concurrent execution of multiple transaction, the event produced by the concurrent execution has timestamp overlaps, resulting in incomplete order in the internal

trace, namely it has partly order between the events, using $\prec$ symbol to represent. On the other hand, the log collected from software system has certain mode; mining these certain modes can help us construct dynamic model. If a_i, b_j are the event type, $\hat{a}_i, \hat{b}_j$ are the corresponding event instance, we can mine six kinds of invariants that relate to the event type a_i and b_j from the traces. They are:

$a_i \to b_j$: The event whose type is a produced by transaction *i* is always followed by the event whose type is b produced by transaction *j*. it can be expressed as: $\forall \hat{a}_i, \exists \hat{b}_j, \hat{a}_i \prec \hat{b}_j$.

$a_i \not\to b_j$: The event whose type is a produced by transaction *i* is always never followed by the event whose type is b produced by transaction *j*. it can be expressed as: $\forall \hat{a}_i, \nexists \hat{b}_j, \hat{a}_i \prec \hat{b}_j$.

$a_i \leftarrow b_j$: The event whose type is a produced by transaction *i* always precede the event whose type is b produced by transaction *j*. it can be expressed as: $\forall \hat{b}_j, \exists \hat{a}_i, \hat{a}_i \prec \hat{b}_j$.

$a_i \| b_j$: The event whose type is a produced by transaction *i* always concur at the same time with the event whose type is b produced by transaction *j*. it can be expressed as: $\forall a_i, b_j, \left(\hat{a}_i \prec \hat{b}_j \wedge \hat{b}_j \prec \hat{a}_i\right)$.

$a_i \nparallel b_j$: The event whose type is a produced by transaction i always never concur at the same time with the event whose type is b produced by transaction j. it can be expressed as: $\forall a_i, b_j, \left(\hat{a}_i \not\prec \hat{b}_j \wedge \hat{b}_j \not\prec \hat{a}_i\right)$.

$a_i \leftrightarrow b_j$: The event whose type is a produced by transaction i always circulates with the event whose type is b produced by transaction j. it can be expressed as: $\forall a_i, b_j, \left(\hat{a}_i \prec \hat{b}_j \vee \hat{b}_j \prec \hat{a}_i\right)$.

Definition 5(state) the state is the function sequence invoked in the implementation process of the program, which is the event set: $S \supseteq \{START, END\}$.

Definition6 (equivalent state) given two interactive traces defined as follows, the first and the second $bt_1 = (m_1, p_{m_1}, v_1) \cdots (m_x, p_{m_x}, v_x)$, $bt_2 = (f_1, p_{f_1}, w_1) \cdots (f_t, p_{f_t}, w_t)$, $bt_1 \Leftrightarrow bt_2$, if and only if $x = t$,and $\forall i = 1,...,x$, $m_i = f_i$, $p_{m_i} = p_{f_i}$, $v_i = w_i$,so the corresponding state of the event in the trace is also equivalent.

Definition7 (transfer) transfer $t = (s, m, P, s')$, $s, s' \in S$, which respectively represents the source and objective state, $m \in M$,is the function in the process of transfer, P is the transfer judgment.

Definition 8(judgment) it is said the transfer whether established or not, $P : D_R \times D_V \to \{True, False\}$, which illustrates the function whether accepts the input parameters and the value of the variable or not, for example, the union of the function login and the transition conditions $P = length(user) > 0 \wedge length(pwd) > 0$ presents that the parameters of the function login whether meets the judgment conditions when the state transits, from which to decide whether produces transfer or not.

Definition 9(path) given the trace length n , $bt = (START, e_1, ..., e_n, END), n \in \mathrm{N}$,To make the model accept the trace bt, the necessary and sufficient conditions: $\exists \prod = (START, m_1, P_1, s_1)(s_1, m_2, P_2, s_2) \cdots (s_{n-1}, m_n, P_n, END)$ for $\forall i = 1, \cdots, n$, $P_i(p_{m_i}, v_i) = true$, $\prod$ is the complete path of model M, which presents the sequence of the trace that the model accepts, and is the basis for the following behavior analysis(abnormal behavior diagnosis, etc). Definition 10(IBDM model) the IBDM model is the four-tuple (S, T_P, s_c, s_z), among them,

1. $S = \{m \mid m \in M\}$ m is the invoking function when the software executes, which describes the state set of the executive software.
2. $T_P : D_M \xrightarrow{P} S$, $D_M \subseteq M \times R \times V$, presents the condition judgment transfer.
3. $s_c = \{START\}$,it is the initial state set.
4. $s_z \supseteq \{END\}$, it is the final state set.

B Constructing Algorithm

Algorithm 1:IBDM constructing algorithm

Input the log document L, regular expression RegExps;

Extracting the trace diagram traceGraph from the log;

According to the definition 4, mining the invariants from traceGraph;

Getting the sub-diagram set of {G}from the dividing traceGraph; setting the initial steps Step as the dividing times that needed; setting the initial value of the dividing steps value as: n=0;

If the sub-diagram does not meet the invariants, we should continue to divide this sub-diagram into two parts, π_1 , π_2 , π_1 contains the event satisfying the invariants, while π_2 does not;

n=n+1;

If n=Step, or the event in the sub-diagram meets the invariants stops, otherwise, execution circulates;

According to the definition 6, combining the equivalent state between sub-diagram, then generating the judgment of P associated with an event, and forming refinement and abstract of the final model;

Algorithm 2 Mining Invariant

1. Initialing the event type a_i , b_j , circulation times N, setting the initial value i=0;

2. According to the definition 4, calculating the invariants:

If the value Follow [a_i][b_j] of the event $\hat{b}_j$ followed the event $\hat{a}_i$,and this value equals the value Occ[a_i] of the times of the event $\hat{a}_i$ appears, So, generates the invariant $a_i \to b_j$;

If the value Follow[a_i][b_j] of the event $\hat{b}_j$ followed the event $\hat{a}_i$ equals 0, So, generates the invariant $a_i \nrightarrow b_j$;

If the value Prec[a_i][b_j]of the event $\hat{a}_i$ appears before he event $\hat{b}_j$,and it equals the value Occ[a_i] of the times of the event $\hat{b}_j$ appears, So, generates the invariant $a_i \leftarrow b_j$;

If the value Prec[ai][bj] of the event $\hat{a}_i$ appears before the event $\hat{b}_j$ equals the value Follow[a_i][b_j]of the event $\hat{b}_j$ followed the event $\hat{a}_i$,So, generates the invariant $a_i \leftrightarrow b_j$;

If the co-occurrence times of the event $\hat{a}_i$ and $\hat{b}_j$ CoOcc[a_i][b_j]=0, and Follow[a_i][b_j]=0, Prec[a_i][b_j]=0, So,generatesthe invariant $a_i \nparallel b_j$;

If the pair event $<\hat{a}_i, \hat{b}_j>$ appears one after another, and the sum of Followpair[ai][bj] and Precpair[a_i][b_j], equals the co-occurrence cumulative total times CoOcc[a_i][b_j]of the event $\hat{a}_i$ and $\hat{b}_j$ in a trace, So, generates the invariants $a_i \parallel b_j$;

3. $i=i+1$;
4. $i=N$, Completing the calculation, Otherwise repeat;
5. Output the invariants;

IV. Examples and Analysis

A Traces Sub-diagram

The interactive log is collected by the monitoring components when the electronic shopping system running, and stored as text documents form. Because we preset the monitoring format, the events partly and orderly arrange by the timestamp (different components execute paralleling, some events have time overlap), for this special format of log, we use regular expression to analyze them. We induce the same event type as a class, and each vertex represents an event, while the edge represents the relationship between each event, and the attributes on the edge represents the parameters transferred. The set of these kind diagrams is called trace diagram. To the log document in this paper, we use the regular expression :(?< timestamp>). + "(? <TYPE>.+)", which divides the log documents into three traces, and each trace corresponds to an interactive transaction. As shown in figure 1.Among them, the trace corresponding to the transaction 1 is:<0,login>, <1,search-goods>, <2,order-items >, <3,valid-order>, <4,get--card>, <5,check-out>, the integer is the ordered timestamp exported from the log.

The experiment result shows that, for different size of log, the trace number extracted from them is almost at the same. As the system is the same electronic shopping system monitored at a certain period of time, and the interactive operation is a relatively fixed mode. The log collected at different time length, only the repetition operations differs, but not produce new interactive operation, which saves the storage space when we construct the interactive behavior model, and simplify the model structure is becoming possible.

Figure1. the trace sub-diagram extracted from the log

B Invariant

The six kinds of invariant mined from the log is the completely logical structure coverage for the log, and they reveal the multiple interactive relationships between the event instances at the same transaction, and the parallel operation between the event instances at the different transaction. By scanning the log, extracting the atomic operation as a candidate event instance, and then adopting the co-occurrence counting method mentioned before to find the relationship between them, forming the correlative relationship set that contains specific temporal logic and semantic logic. These constraint conditions constituted by multiple relationships between the event instances have made up for the widespread defects that the system bugs which is difficult to be detected from the log. We mined six kinds of invariant from the log that contains 280 records; table 1 has given the part result of invariants. It is easy to find that, in our actual shopping operation, counter-examples "$invalid-order \rightarrow get-card$" of invariants is not allowed. That is to say in the case of lacking stock, still order the goods and make payment, which reflects the existence of the counter-example path in the model. This kind of counter-example invariants is captured from the electronic shopping system, and it is obviously not consistent with the normal situation, but really happened in the process of the transaction, according to these invariants to construct model, then traversing the model and determining whether there is a counter-example path or not, which is the main basis of detecting system bug.

TABLE 1

PARTLY RESULTS OF THE SIX KINDS OF INVARIANTS

$a \rightarrow b$	$a \not\rightarrow b$	$a \leftarrow b$
$get-card \rightarrow check-out$	$check-out \not\rightarrow get-card$	$\log in \leftarrow check-out$
$get-card \rightarrow \log in$	$check-out \not\rightarrow valid-order$	$\log in \leftarrow get-card$
$invalid-order \rightarrow get-card$	$get-card \not\rightarrow valid-order$	$\log in \leftarrow invalid-order$
$valid-order \rightarrow get-card$	$get-card \not\rightarrow order-items$	$\log in \leftarrow valid-order$
$a \leftrightarrow b$	$a \parallel b$	$a \not\parallel b$
$search-goods \leftrightarrow order-items$	$search-goods \parallel search-goods$	$get-card \not\parallel get-card$
$\log in \leftrightarrow check-out$		
$search-goods \leftrightarrow invalid-order$		
$valid-order \leftrightarrow get-card$		

C Refinement and Abstraction

Abstraction and refinement is the double operation of the IBDM modeling. From the beginning of the initial model, the first step is to execute the model refinement, which is an iterative process. IBDM refines every trace sub-diagram until the model meets all mining invariants, then we merge those sub-diagrams with abstraction. The abstraction process is restrained, which is bound to ensure that no invariant violation in the process of refinement. When can not execute abstraction, we output the model. As long as the model does not satisfy the invariants constraint, IBDM will execute the division continuously. The checker based on the model of the FSM (finite state machine) is adopted to verify whether the model satisfies the invariants constraint or not, that is to say, we convert each invariant to the miniature FSM that can accept traces, and these traces satisfy with the invariant constraints. We should update these FSM model when traverse the model, if the model does not satisfy the invariant constraint, the model checker outputs a counter-example path. Refinement implements by identifying the counter-example set, and decide the candidate division set. IBDM parallel searches the counter-example in the traces model to identify division. In the traces, what is needed is the prefix part of the counter-example path, and IBDM finds out the longest prefix. The last division for this prefix in the model is the refinement of the candidate division, we separate the candidate division of the event instance according to the other input-edge divided is whether the direct precursor of the candidate division or not. It is allowed the existence of the counter-example evolutionary path. Through the search of the final model counter-example path, we can detect the system bug; the corresponding description is made in paragraphs of the invariant part.

Refinement may produce many results, when this situation happened, the model contains the division could be merged, and do not violate invariants. IBDM uses the K-tail equivalence class method to abstract; which begins with the most fine grit model, merging each K-tail equivalence partitioning until there is not any pair K-tail equivalence partitioning. For example, no two division is the root of the sub-diagram which has the same depth of K, and each step of the algorithm uncertainly choices and merges a pair of K-tail equivalence partitioning, producing a relatively accurate model. Then, from the start of this model, combining with the constraint conditions to merge all divisions, the merged results can satisfy the invariants. Finally the model produced is local optimal, which is to say, if merging any two division among them would violate the invariants. In ubuntu environment of Linux operating system, execute the IBDM algorithm and output the final model is shown in figure 2.

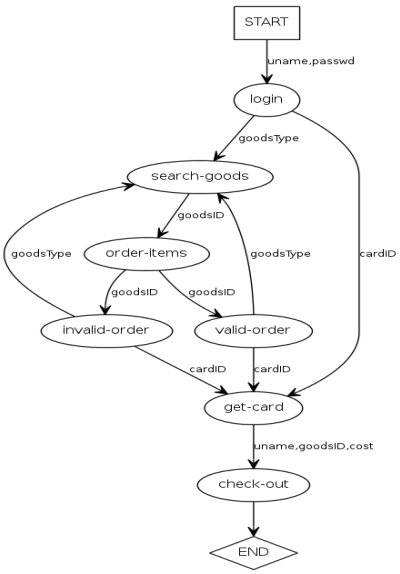

Figure2. The final model

D Accuracy and Efficiency Evaluation

Definition 11(model accuracy) is an evaluation for the model acceptance ability, and describes the event sequence accepted accounting for the proportion of the total number of trace in log f.

By the hypothesis testing and distribution fitting, determined the event appearance in the log obey Poisson distribution $P(\lambda)$, and λ is the average number of events occurred in unit time. We traverse the event existed both in log and in model, splitting them into some operation sequences in chronological order, the ratio between the number of completely path Π in the model and these operation sequences is the accuracy f, which can be

expressed as:

$$f = \frac{\varphi\left(\sum_{m\in M} P(\lambda)\times|L| - I_n\right)}{\psi(N)} \quad (1)$$

Among them, $|L|$ is the length of log documents, I_n is the number of the event not existing in model, $m \in M$ is the event in the log, $\psi(N)$ is the completely path numbers in the model, N is the model scale.

As is shown in figure 3, for the on-line electronic shopping system log, the comparative curses are the popular modeling algorithm K-tail and IBDM modeling algorithm proposed in this paper. All of the data is the average operation result by 100 times. Taking $\lambda = 4$,the experiment shows that, the IBDM modeling algorithm uses the invariant to constraint the model, with the increment of the length of the log documents, the successful matching of probability increases between the completely path and the event sequence in the model, explaining the acceptance ability of the model is also growing, instead, with the expansion of the model scale, without the constraint of the invariants, the model generated by the K-tail algorithm contains more and more redundant state, corresponding to the repetition of the path. The acceptance ability of the model is interfered by the repeated matching, and its accuracy has also been affected. The ratio of IBDM modeling algorithm accuracy is around 70%, and has a relation of the log document integrity. The algorithm still have a strong rely on the log documents, in the same case, a complete log documents that contains all of the unknown event will achieve a better accuracy.

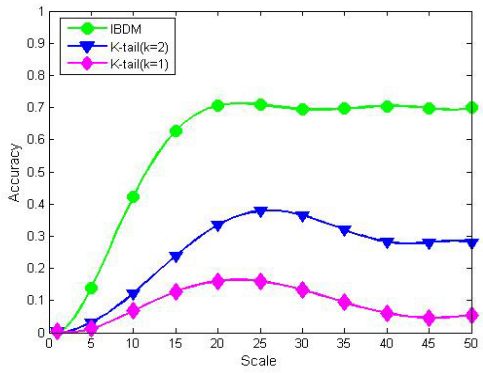

Figure3. Accuracy compared

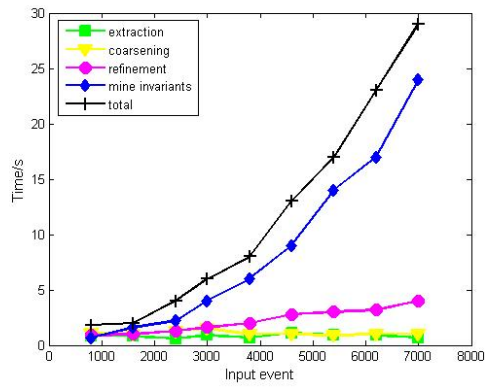

Figure 4.Algorithm efficiency

As is depicted in figure 4, the IBDM modeling can be divided into four processes including the invariant mining, the sub-diagram extraction, refinement and abstraction. The data used in the experiment is the log collected from the online electronic shopping system within one week. From the figure 4, the invariant mining has obviously spent more time comparing with other processes, so the well designed mining algorithms will directly affect the efficiency of the whole modeling. Second, when the model is set up, no other assumptions condition is required, so this method can be used to the behavior modeling of most networked software.

V. SUMMARY AND FUTURE WORK

Based on the constraints of the invariants, the software interactive behavior dynamic modeling provides software behavior analysis with new ideas and method. For the interaction log documents produced by the on-line electronic shopping system, we proposed that abstracting the log with the gigantic scale and complicate content to the intuitive and concise model, which not only meets the invariant constraints extracted from the log, but also vividly describes the program behavior rules. Through which we can have a deep understanding and cognition for the system. Analyzing the counter-example path of the model, can reveal the potential threats existing in the system, and validate the known bug. We can adopt the model for the malicious software spreading dynamics analysis by extracting the malicious software running trace from the model to control and prevent threats, so the reliability and safety of the system will be improved.

How to detect the program behavior anomaly and how to pin down to maintain the whole system stability when the bug is found, how to analyze certain special synchronous behavior will be our future work.

ACKNOWLEDGMENT

This work is supported by the National Natural Science Foundation of China under grant No. 61171192 and 61170102, the Natural Science Foundation of Hunan province in China under grant No. 11JJ4050 and 11JJ3070, the Education Department Foundation of Hunan Province under the grant No. 11B039, 11W002 and 11C0400.

REFERENCES

[1] Zhang Yaoxue. Transparence computing: Concept, architecture and example [J]. ACTA ELECTRONICA SINICA, 2004, 32(12A): 169-173. (in Chinese)

[2] MA Yu-Tao, HE Ke-Qing, LI Bing, LIU Jing. Empirical Study on the Characteristics of Complex Networks in Networked Software[J].Journal of Software, 2011, 22(3): 381-407. (in Chinese)

[3] YANG Fu-qing, MEI Hong, LU Jian, JIN Zhi. Some Discussion on the Development of Software Technology[J]. ACTA ELECTRONICA SINICA, 2002, 30(12A):1901-1906. (in Chinese)

[4] CHEN Huo wang ,WANG Ji ,Dong Wei. High Confidence Software Engineering Technologies [J]. ACTA

ELECTRONICA SINICA 2003, 31(12A):1933-1938.(in Chinese)
[5] Leonardo Mariani, Mauro Pezzè, Oliviero Riganelli, Mauro Santoro. SEIM: Static Extraction of Interaction Models [A]. International Workshop on Software Engineering[C]. Cape Town, South Africa: IEEE Computer Society, 2010. 22-28.
[6] Christopher Ackermann, Mikael Lindvall, Rance Cleaveland. Towards Behavioral Reflexion Models [A]. Reliability Society [C]. Mysuru, India: IEEE Computer Society,2009. 175-184.
[7] Jonathan E. Cook, Alexander L. Wolf. Discovering Models of Software Processes from Event-Based Data [J].ACM Transactions on Software Engineering and Methodology, 1998, 7(3): 215-249.
[8] Kai-Yuan Cai, Bei-Bei Yin. Software execution processes as an evolving complex network [J]. Information Sciences, 2009,179 (12): 1903-1928.
[9] Tao Li, Wei Peng, Charles Perng, Sheng Ma, and Haixun Wang. An Integrated Data-Driven Framework for Computing System Management[J]. IEEE Transactions on Systems, 2010, 40(1): 90-99.
[10] Chun ying Zhao, Jun Kong, and Kang Zhang. Program Behavior Discovery and Verification: A Graph Grammar Approach [J]. IEEE Transactions on Software Engineering, 2010, 36(3): 431-447.
[11] Anton ChuvAkin, GunnAr Peterson. How to DoApplication Logging Right[J]. IEEE Computer and Reliability Societies, 2010, 8(4):82-85.
[12] ZhenLi,JunFeng Tian and Liu Yang. An Improved Software Behavior Model in System Call Level and Trustworthiness Evaluation.[J]. Information Technology Journal, 2011, 10(11):2208-2213.
[13] FU Jian-Ming, TAO Fen, WANG Dan, ZHANG Huan-Guo. Software Behavior Model Based on System Objects[J]. Journal of Software, 2011, 22(11): 2716−2728. (in Chinese)
[14] Curtis E.Hrischuk, Murray Woodside. Logical Clock Requirements for Reverse Engineering Scenarios from a Distributed System [J]. IEEE Transaction on Software Engineering, 2002, 28(4): 321-338.
[15] Selvaraj Srinivasan, R. Rajaram. A Decentralized Deadlock Detection and Resolution Algorithm for Generalized Model in Distributed Systems [J]. Distributed and Parallel Databases, 2011, 29(4):261-276.
[16] Jonathan E. Cook, Zhidian Du, Chongbing Liu, Alexander L. Wolf. Discovering models of behavior for concurrent workflows[J]. Computers in Industry, 2004,53(3): 97 – 319.
[17] Jonathan E. Cook, Zhidian Du. Discovering thread interactions in a concurrent system [J]. The Journal of Systems and Software, 2005,77(3): 285 – 297.
[18] Jonathan E. Cook, Cha He, and Changjun Ma. Measuring Behavioral Correspondence to a Timed Concurrent Model[A]. IEEE Computer Society's Technical Council on Software Engineering[C]. Florence, Italy: IEEE Computer Society,2001.332-341.
[19] PENG Cheng, YANG Lu-ming, MAN Jun-feng. Research on Tokenizing Behavior Footprints of Incomplete Transaction[J]. Journal of Chinese Computer Systems, 2011, 32(8): 1593-1598. (in Chinese)
[20] Zhen Li , Junfeng Tian. A Software Behavior Automaton Model Based on System Call and Context[J].Journal of Computers,2011,6(5):889-896.
[21] Zhen Li , Junfeng Tian. An Approach of Trustworthiness Evaluation of Software Behavior Based on Multidimensional Fuzzy Attributes [J]. Journal of Computers,2012, 7(10):2572-2577.
[22] Yongfeng Yin, Bin Liu. Research on Formal Verification Technique for Aircraft Safety-Critical Software [J]. Journal of Computers, 2010,5(8): 1152-1159.
[23] J.Tan, X.Pan, S.Kavulya, R.G, and P.Narasimhan. SALSA: Analyzing Logs as State Machines[A]. USENIX Workshop [C].San Diego, CA: USENIX, 2008.6-6.
[24] J.Tan, X.Pan, S.Kavulya, R.G, and P.Narasimhan. Mochi: Visual Log-Analysis Based Tools for Debugging Hadoop [A]. IEEE International Conference on Distributed Computing Systems Workshops [C].Geneva, Italy: IEEE Computer Society, 2010. 795-806.
[25] J.Yang and D.Evans. Dynamically Inferring Temporal Properties[A]. ACM SIGART/SIGSOFT Workshop [C]. Washington, DC: Association for Computing Machinary, 2004. 23-28.
[26] A.W.Biermann and J.A.Feldman. On the Synthesis of Finite-State Machines from Samples of Their Behavior [J]. IEEE Trans. Comput., 1972, 21(6):592 – 597.
[27] J. Whittle and J. Schumann. Generating State chart Designs from Scenarios [A]. New York, ACM [C]. Limerick, Ireland: IEEE-CS Computer Society, 2000.314-323.
[28] C. Damas et al. Generating Annotated Behavior Models from End-User Scenarios [J]. IEEE Transaction on Software Engineering, 2005, 31(12): 1056-1073.

Cheng Peng He is pursuing the Ph.D. degree in School of Information Science and Engineering, Central South University, Chang Sha, China. His current research interests include trusted software, computer network and software engineering.

Luming Yang He is a Professor and Ph.D. advisor in School of Information Science and Engineering, Central South University, Chang Sha, China. His research interest includes database system, computer network and software architecture.

Junfeng Man corresponding author,. He received the Ph.D. degree in School of Information Science and Engineering, Central South University in 2010, ChangSha, China. He is professor in College of Computer and Communication of Hunan University of Technology. His research interests include trust software, pervasive computing.