International Journal of Computational Intelligence Systems, Vol.3, No. 5 (October, 2010), 542-552

Behavior-aware Trustworthiness Study of Networked Software

Xianwen Fang *
Department of Computer Science and Technology, Tongji University, Shanghai 201804, China

Changjun Jiang ,Xiaoqin Fan
Key Lab of Embedded System and Service Computing, Ministry of Education,
Tongji University, Shanghai 201804, China
E-mail{:cj-jiang,xqfan}@tongji.edu.cn
www.tongji.edu.cn

Received: 26-12-2009
Accepted: 02-06-2010

Abstract

The essence characteristics of software trustworthiness are software execution effect and behavior can be anticipated, which is an important index of software quality. Under the open and dynamic environments, some uncertainty factors cause the behavior of software to be uncontrolled, uncertainty and unpredictable. Behavior-aware networked software trustworthiness research methods are proposed in the paper. Firstly, we propose the analysis methods of the consistency between the inferred specification models with component specification model. Then, for analyzing the component interaction behavior, the behavior relativity analyzing method is presented based on Petri net. Finally, aimed to the outer factors, we analyze the behavioral congruence between theoretical composite models with dynamic behavior model based on running logs. Theoretical analysis and the example analysis indicated that this method is benefit to analyze the trustworthiness of networked software.

Keywords: trustworthiness, software behavior, networked software, consistency, Petri net.

1. Introduction

With the computer application and network technology development, software has entered into the national economy and social life domains, and plays the very important role in the information society. Software is taken as the information system's core, the Internet application's cornerstone, has already become the modern computer system's soul. But computer system's flaw is led to by software's question to a great extent. One hand, along with the application demand's complication and the dynamic change, software's scale becomes bigger and bigger, the function is getting more and more complex, which causes software development and evolution continually to become more and more complex, and corresponding trustworthy software construction technology is lacking, these causes software product having known or the unknown flaws which threat software system security and reliably running seriously. On the other hand, software's running environment and development environment extend from the traditional close static environment to open, dynamic and changeable network environment, thus application software system formerly whose properties are implement in one action, the structure inlaying and the weak evolution, and having the limited independency, the fixed encapsulation, the interactive monotonous, can not meet the present application requirements. Under the network environment, the

* Corresponding author. Tel.: +86-013162346379; fax: +86-021-69589864. E-mail address: fangxianwen@hotmail.com; xwenfang@tom.com.

behavior of computing entity is uncontrollability, uncertainty and unpredictable and so on, so the quality assurance of software has been affected greatly in execution process, at the same time, the traditional software engineering thought faces stern challenge[1]. Therefore, the development trend of software industry faces the complicated and diversified application demand based on existing and new development software components, produces high quality software system with controllable, manageable and prevention ability through component collaboration.

Under the open environment, the trustworthiness of networked software system mainly comes from the three reasons: Firstly, in the traditional data-driven software construction method, data effect is played attention more, the process behavior is neglected, thus which leads to the execution behavior of large software system uncontrolled, unmanageable and unpredictable. This reason is along with the software scale expanding quickly, the software system need use the composite way to build, these components development usually needs over a hundred people, and even over a thousand people design together, so we must avoid person subjective difference as far as possible. At the same time, components are also content self-containing, structure independent. But "the black-box" aware, data-driven construction method is unable to determine and solve individual difference question in the component collaboration process, simultaneously neglect the individual interaction influence each other. Secondly, view from software structure, the software entity and the collaboration part is the tight coupled. This kind of tight coupling development pattern is very difficult to adapt the dynamic change of application demand in networked environment. Finally, the network application environment's opening and dynamic change cause to have the gap between the seal supposition in software design and the network environment opening reality. Network environment's open characteristic causes the software to be able to process the disturbance and the destruction from the natural factor (for example network delay, blocking and so on) and the human factor (for example malicious attacking, virus attacking and so on), and safeguards the software normal operation, however the existing seal supposition design method is unable to satisfy such application requirements.

Therefore, in order to make software system trustworthy, aimed to the open and dynamic environment, software system development, running and maintenance need to be provided direct and effective support, and to transform from the data-driven software construction way to the behavior-aware construction way, from the tight coupling software architecture to the collaboration entity polymerization loose coupling structure, from inlay and monotonous interaction mode to separate and diversiform collaboration mode, from the close running environment to the open running environment.

The rest of this paper is organized as follows. Section 2 introduces related work. In Section 3 details proposes behavior-aware trustworthiness study methods of networked software. A case of E-Banking simulation system is given out in the Section 4. Finally, Conclusions are in Section 5.

2. Related Works

Software trustworthiness is introduced since the 1970s, in the Ref.2 proposed the credible system concept. The trustworthiness question of information system has been focus point of the academic circle and the industrial world. In recent years, the software trustworthiness research has been paid a big attention all over the world. In the US, DARPA, NSF, NASA, NSA and other DoD organization participated positively the trustworthy software and system research[3]. Europe started the research project of "Open Trusted Computing" in January, 2006, in order to develop the source trustworthy software. Regarding the concept of trustworthiness, researcher has proposed different description from different angle. From system's angle, the definition of trustworthiness based on the ISO/IEC 15408 standards are: A trustworthy component, operation or the process behavior under the operating condition can be predicted, and can resist the destruction well by the application software, virus and physical disturbance. The trusted computing organize [4]proposed the definition of trustworthiness: If an entity always execute conforming to the expected goal, then the entity is trustworthy. From the user experience's angle, Ref.5 thought that the trusted computing was the reliable security computing, and including the user trust degree to software. From the network behavior angle, Ref.6 thought that the trustworthy network system's behavior and result may be anticipated, and can reach

the function that the behavior can be monitored, the behavior result may be appraised, and the abnormal behavior may be controlled. Ref.7 proposed that trustworthy software was the credible status, credible ability and the trusted behavior. Ref.8 proposed behavior model and verification methods of internet software architecture aimed to the software behavior uncertainty and imperfection under the networked environment.

For guaranteeing software trustworthiness, the formal method has obtained many attentions. Like formal verification, model checking method, theorem proof method, software testing and so on [9]. The existing theory and method have some role in trustworthiness for these software components which in the close environment supposition, the small scale and low complexity to a certain extent, but have big limitation for opening environment and dynamic construction large scale application software system[10]. This reason is, firstly, the components are trustworthy, but the obtained large scale software system may not be trustworthy after dynamic construction. Because software component behavior may influence mutually in collaboration process, thus causes to composite software behavior may not be equal to the behavior sum of the software components. Secondly, the state space explosion question has limited the formal method enormously in the large scale software application, thus reduces its usability. Moreover the formal theory, software testing and the fault-tolerant technology by simple splicing are also insufficient as unified foundation to analyze software trustworthiness. Thirdly, the formal verification methods mainly aim the program correctness question in the close environment, but under the networked environment, the function correct software cannot guarantee that it is trustworthy. Fourthly, with the improvement of software complex and scale, and the dynamic evolution of process, the traditional software testing technology is difficult to discover and locate software untrustworthy point, and the difficulty is also getting bigger and bigger. Because under the networked environment, the software components carry on the tasks to implement interconnection, intercommunication, cooperation and alliance with other software entity on cross network through collaboration way. Therefore, untrustworthy behavior of software not only exists in the software components, simultaneously also exists in the software collaboration process, but the present checking technology mainly faces in the internal component. Finally, the existing program proof theories and the model checking methods carry on the program properties verification by execution effect. But according to the definition of trustworthiness, the software behavior is the core of the software trustworthiness.

3. Behavior-aware Trustworthiness Study Methods of Networked Software

3.1. *Analysis Framework*

Under the networked environment, owing to software evolution continually, it is very difficult to guarantee the software quality using traditional software engineering method; meanwhile, for dealing with outside attacking, there is a big limitation using the existing program verification and model checking methods. So it is difficult to guarantee networked software controlled, managed, prevention ability, as well as the result and the behavior may be unpredictable, also very difficult to realize the trustworthiness. The software running behavior can reflect comprehensively the software interaction behavior change situation in the dynamic evolution under the complex environment.

Trustworthiness is important index of software quality, its essence is the software execution effect and the execution behavior can be anticipated. Therefore, this paper viewed from the software behavior, a behavior-aware networked software trustworthiness research framework is proposed, showed in Fig.1. The main research ideas are that determining the consistency between the component specification description and component implementation before the components used, that is, the consistency between the specification description models with the inferred specification[11] behavior model. If they are consistent, it showed that the component function of specification description and the component implement function are the same, which may guarantee the component identify, may also analyze some bug in the component. The component which satisfies the specification consistency is sent to the behavior certification center for identity certification, and the center issues the behavior certificate to the component provider after inspection.

Some components with behavior certificate may combine together to realize complex function requirements, but composite component need be verified in the combination process, in order to determine that the composite component is function correctness, does not have the deadlock or trap structures. At the same time, the composite component need analyze component interaction behavior relativity, the aims are to determine whether one component

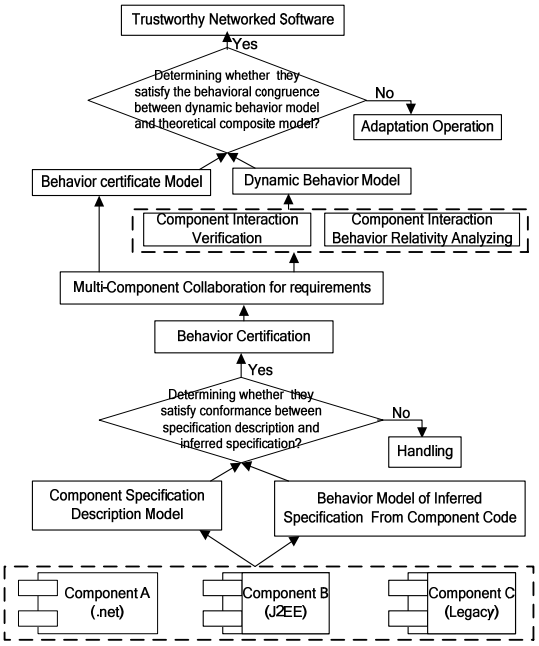

Fig. 1. The trustworthiness analyzing framework of networked software.

behavior be influenced by the others. If the composite component does not satisfy the interaction behavior consistent relativity, then the function of composite component may not be the function sum of the candidate components. So, it can not satisfy the function requirements. Finally, we extract the dynamic behavioral model of composite component according to the component running logs, and compare the dynamic behavior model with theoretical composite model, judge whether the two models satisfy behavioral congruence, if they are behavioral congruence, then the component interaction process is not to be influenced outside factors, such as virus, vicious program, and network environment and so on. The implementation process can be controlled, managed, and prevented, its result may be also anticipated, and building networked software by the methods is also behavior trustworthy.

3.2. *Component Specification Consistency and Behavior Certification*

In networked environment, owing to system evolution constantly and unpredictable external environment, it is necessary to use software behavior to analyze software quality which can be controlled and managed in software lifecycle. So we need keep the software specification description consistent with implementation. Many software tasks require specifications: verifying programs requires specifications of their intended behavior, testing programs requires specifications to determine the input domain and expected outputs, and maintaining programs requires specifications to understand what aspects of the behavior can be modified. Unfortunately, most programs do not come with precise specifications. Worse, those that do often fail to preserve the consistency of specifications and implementations. As the implementation changes, the specification becomes increasingly incorrect[11].

Analyzing the behavior of software systems, in order to aid program comprehension, reduce their maintenance costs, and improve their quality, further to make the software trustworthy, is a complex and challenging task. Having incorrect, incomplete, or outdated documented specifications, as a result of short time-to-market constraints, changing requirements, and poorly managed product evolution, reduces comprehension of the code base, increases maintenance costs, and adds challenges towards verification of their correctness. One approach to address this challenge is to automatically infer specifications of a system from its execution traces by specification mining methods. Specification mining methods have been addressed in Ref.12and Ref.13. Which includes two methods; one is static specification inference[12] on basis of program code, the other is dynamic specification inference[13] using a program's dynamic behavior on sample executions.

Ref.14 proposed a dynamic analysis approach for automatically inferring temporal properties from a program's execution traces and demonstrated that the inferred properties are useful for supporting program evolution on some small examples under controlled conditions, but the methods is only available in small program, and can not deal with dynamic behavior, the networked software scale is comparatively big and dynamic.

Although early work in this area emphasized static analysis of the program text, several researchers have explored the possibility of using a program's dynamic behavior on sample executions to infer a specification recently. Ref.11 identified reasons why scaling dynamic inference techniques had proven difficult, and introduced solutions that enable a dynamic inference technique to scale to large programs and work effectively with the imperfect traces typically available in industrial scenarios, and described approximate inference algorithm, and evaluated heuristics for winnowing the large number of inferred properties to a manageable set of interesting properties. But the methods omitted some inappreciable behavior, which may be vicious behavior or lead to some problem in networked environment. In the paper, we propose a novel analyzing methods about component specification description model consistent with the inferred specification model (showed in the Fig.2), and behavior certification methods.

The digital certificate provides the identity certification for the network computing, but there is still not having the very good mechanism to verify component behavior presently. The behavior certificate method is proposed based on the digital certificate; the aims are to describe the software behavior conveniently, in order to make up the insufficiency that digital certificate can not describe the behavior. The behavior certificate method is a supplement, but it is not a substitute for the digital certificate. About the mechanism of behavior certificate and the behavior certification center construction, the paper does not make the analysis. This paper only uses the behavior certificate methods to carry on the component behavior certification. Through behavior certification, the component meets requirements as follows: component specification description model is consistent with the inferred specification model, and the component is function correct, executable, not existing deadlock or trap structures and so on, which may influence running correctly.

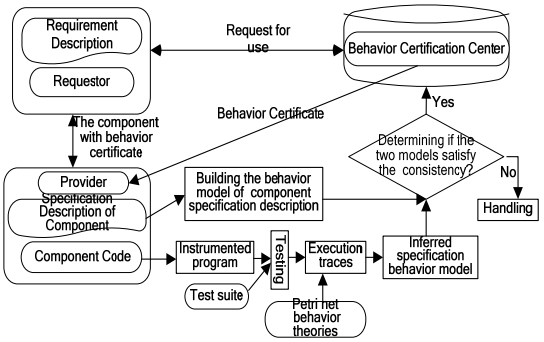

Fig. 2. The methods of component specification consistency and behavior certification.

In the Fig.2, we can see that the methods aimed to verify the consistency between the component specification description and component implementation. In open environment, the new components and legacy components are required to make the specification consistency checking before component used. We build up the behavior model of the component specification description through analyzing the dependent relationships in component specification description, the work is simple, there is some tools[15] can be used. The inferred component specification behavior model is obtained by analyzing component code, which is transformed into instrumented program firstly, and abstract the events and states of interest. Then run the instrumented program through a set of test cases to collect execution traces. We present an algorithm about inferred specification behavior model using the Petri net behavior theories. Finally, we compare the behavior model of component specification description with inferred specification behavior model, and determine whether they satisfy consistency. If they satisfy the consistency, the component specification description is consistent with the component implementation, at the same time, the component is function correctness, no deadlock or trap by analyzing the inferred behavior model, then the behavior certification center passes the component certification, and issue a behavior certificate to the component provider, keep a copy in the behavior certification center. Here, the certificate content is the inferred specification behavior model. In our methods, we adopt the temporal Petri net to analyze the consistency. Temporal Petri net which combine the

advantages of Petri nets and temporal logic in which temporal constraints of a given net are represented by the temporal logic formulas, can describe clearly and compactly causal and temporal relationships between the events of a system[16]. The basic Petri net is convenient to descript the component behavior, and the temporal logic formulas can depict the behavior constraint and component properties.

Definition 1. Let σ is execution trace of event/transition model *M*, if event/transition model *M* satisfies the behavior property bp, noted as M|=bp, then $\forall \sigma' \in \sigma$, $\sigma' \models bp$.

Definition 2.If $M_i \cap M_j \neq \Phi$ $(1 \leq i, j \leq n)$, then model M_i and M_j can be assembled into a new model by the share point, noted as $M_i \propto M_j$.

Algorithm 1. The building algorithm of inferred specification behavior model

Input: component code.

Output: inferred specification behavior model.

(i) Transforming component code into instrumented program, and abstracting the events and states of interest.

(ii) Running the instrumented program through a set of test cases, in order to collect component execution traces.

(iii) Suppose the execution traces are $\sigma_1, \sigma_2, ..., \sigma_n$ based on different test cases, σ is compose of the events and states of interest. Aimed to every execution traces σ , building the corresponding event/transition models $M_1, M_2, ..., M_n$.

(iv) According to component domain knowledge, determining that the component need satisfy behavior properties, and using the temporal logic formulas to represent them, assuming the behavior properties are $bp_1, bp_2, ..., bp_k$.

(v) Selecting the event/transition model in order, according the definition2, assuming $M_i, M_j \models bp$, M_i and M_j composite model is M_{new} , if M_{new} |=bp, then M_{new} is put into the model table, and rid the M_i and M_j from the model table. Else, we use heuristic inference methods[14] to alter the composite model, until all model be accomplished.

(vi) Transforming the event/transition model which satisfies the properties into temporal Petri net model, that is, transforming the event into the place of Petri net, transforming the transition into the transition of Petri net, and transforming the edge into the directed arc of Petri net.

3.3. *Component Interaction Behavior Relativity Analyzing based on the Behavior Theories of Petri net*

Under the networked environment, some complex and large scale function requirements are implemented by many components interaction each other, but there may be have some problems in the process of interaction, such as deadlock, trap, and conflict and so on, so the behavior verification of interaction component is very necessary. Deadlock and trap checking have been studied in many literature presently, in Refs.17-19 mentioned the verification method based on Petri net. Owing to the verification methods based on Petri net has been mature; the problem of behavior verification should not be studied in the paper.

After the interaction behavior of components is verified, the composite interaction component satisfies the structure properties, no deadlock or trap in the composite component. At the same time, the candidate components satisfy the function requirements. But, these according with the function needs may not to reach the expected function requirements. Main reason is the behavior relativity not taking into account during components interacting, which may influence the function implementation. Component behavior relativity mainly refers to one model behavior may be influenced by others when component interacting, leading to some model behavior function occur to change, even some interactions are insignificance. The interaction behavior relativity has four kinds: consistent behavior relativity, interactive behavior relativity, controlled behavior relativity, exclusive behavior relativity. Consistent behavior relativity is one kind of good interactive behavior relationship, which indicates two interaction models accomplishing the function requirements and the behavior of themselves are not influenced. Interactive behavior relativity refers to mutually overlapping phenomenon happened in behavior after interacting with each other, the both model behavior are influenced, and can not keep the behavior invariance. Controlled behavior relativity refers to one model behavior controlled by the other, function and behavior of controlled model can not be original behavior after interacting. Exclusive behavior relativity refers to that the both interaction models are not compatible, behaviors are mutually-exclusive, can not carry out

composition. Under the networked environment, we should guarantee the interaction components behavior is consistent, and unable to appear interactive behavior relativity and controlled behavior relativity, let alone the exclusive behavior relativity. With regard to non-consistent behavior relativity, some adaptation measures must be adopted, in order to make the interaction components behavior consistent. We propose an analyzing algorithm of component interaction behavior relativity (showed in Algorithm 2) based on the behavior theories of Petri net[20,21].

Algorithm 2: The decision algorithm of component interaction behavior relativity

Input: two Petri net models

Output: the interaction behavior relativity type of two components

Let $PN_i = (P_i, T_i; F_i, M_{0i})(i = 1,2)$ are two Petri nets, $PN = PN_1 O_T PN_2$, and $\Delta = T_1 \cap T_2$, $X_{ij_i}\left(j_i = 1,2,...,q_i^{'}; i = 1,2\right)$ are all the minimum T-invariant of PN_i, $X_{ij_i}^{\Delta} = \Gamma_{T_i \to \Delta}\left(X_{ij_i}\right)$, $j_i = 1,2,...,q_i, q_i \le q_i^{'}; i = 1,2$ are the non-zero projection vector of the minimum T-invariant of PN_i, q_1 and q_2 are the number of PN_1 and PN_2 respectively.

(i) According to the definition of Petri net incidence matrix, computing the incidence matrix of PN1 and PN2.

(ii) Computing the minimum T-invariant $X_{ij_i}\left(j_i = 1,2,...,q_i^{'}; i = 1,2\right)$ of PN1 and PN2.

(iii) Computing the projection vector $X_{ij_i}^{\Delta} = \Gamma_{T_i \to \Delta}\left(X_{ij_i}\right)$, $j_i = 1,2,...,q_i, q_i \le q_i^{'}; i = 1,2$ of the minimum T-invariant of PN1 and PN2 on the share transition Δ.

(iv) Determining if the projection vectors can be linear expression each other, that is, if the projection of minimum T-invariant of PN_1(PN_2) can be expressed by linear combination of the projection of minimum T-invariant of PN_2（PN_1）?

\\ The problem can be transformed into to determine if the equation $X_{ij_i}^{\Delta} = \sum_{j_{3-i}=1}^{q_{3-i}} k_{3-ij_{3-i}} X_{3-ij_{3-i}}^{\Delta}$, $j_i \in \{1,2,...,q_i\}$, $j_{3-i} = 1,2,...,q_{3-i}, 0 \le k_{3-ij_{3-i}} \le 1, i = 1 \vee 2$ has non-zero solutions (that is, solving the equation about $k_{3-ij_{3-i}}$).

\\ If the equation has non-zero solutions, the $k_{3-ij_{3-i}}$ is not all zero, which means that $X_{ij_i}^{\Delta}$ can be non-negative linear expressed by other some vectors, then $b\left(X_{ij_i}^{\Delta}\right)$=1, otherwise, there only has 0 solution, that is $k_{3-ij_{3-i}}$=0, which means that $X_{ij_i}^{\Delta}$ can not be non-negative linear expressed by other some vectors, then $b\left(X_{ij_i}^{\Delta}\right)$=0.

The interaction behavior relativity type is determined as followed:

(a) If $\exists X_{ij_i}^{\Delta}, j_i \in \{1,2\cdots, q_i\}$ makes $\exists X_{ij_i}^{\Delta}$ can not be liner represented by vector group $\exists X_{3-ij_{3-i}}^{\Delta}$ $j_{3-i} \in \{1,2,\cdots, q_{3-i}\}$, $i = 1 \vee 2$, then PN_1 and PN_2 are not consistent behavior relativity.

(b) If $\exists X_{1j_i}^{\Delta}, j_i \in \{1,2\cdots, q_1\}$ makes $\exists X_{1j_1}^{\Delta}$ can not be liner represented by vector group $\exists X_{2j_2}^{\Delta}$, $j_2 \in \{1,2,\cdots, q_2\}$, $\Gamma_{T_2 \to \Delta}(L(PN_2)) \subseteq \Gamma_{T_1 \to \Delta}(L(PN_1)$, then PN1 and PN2 are controlled behavior relativity.

(c) If $\exists X_{ij_i}^{\Delta}, j_i \in \{1,2\cdots, q_i\}$ makes $\exists X_{ij_i}^{\Delta}$ can not be liner represented by vector group $\exists X_{3-ij_{3-i}}^{\Delta}$, $\exists X_{3-ij_{3-i}}^{\Delta}, j_{3-i} \in \{1,2,\cdots, q_{3-i}\}$, $i = 1 \wedge 2$, and $\sigma_\Delta \in \Gamma_{T_1 \to \Delta}(L(PN_1)) \cap \Gamma_{T_2 \to \Delta}(L(PN_2))$: $\sigma_\Delta = \sigma_\Delta^{'} \sigma_\Delta^{"*}$, then PN_1 and PN_2 are interactive behavior relativity.

(d) If $\exists X_{ij_i}^{\Delta}, j_i \in \{1,2\cdots, q_i\}$ makes $\exists X_{ij_i}^{\Delta}$ can not be liner represented by vector group $\exists X_{3-ij_{3-i}}^{\Delta}$, $j_{3-i} \in \{1,2,\cdots, q_{3-i}\}$ $i = 1 \wedge 2$, then PN_1 and PN_2 are exclusive behavior relativity.

(v) Repeating the process (ii)-(iv), and then returning the interaction behavior relativity.

3.4. *The Consistency Analyzing between Theoretic Model with Dynamic Behavior Model*

Viewed from component interaction, many components are assembled to realize the function integration, implement new demand, and meet the dynamic change. But the combination operation may bring some problems, mainly including the structure question, the behavior interaction influence, and dynamic running behavior abnormal and so on. The component interaction verification is to process the structure properties question, the aims are that the composite component may carry out, does not have the deadlock or trap. The component interaction relativity analyzing is to solve the component behavior mutually influenced after interacting with each other, in order to avoid the function and behavior of component to suffer injury,

cannot achieve the function and behavior requirements of composite component. But under the networked environment, the component is affected constantly by outer, such as the malicious program, the virus, other extraneous factor and so on, which may lead to the abnormal behavior. It is very difficult to solve through the behavior verification and the behavior relativity analyzing methods. The dynamic behavior analyzing of component is very necessary, to analyze affected situation by other factors in the process of component interaction. The paper uses the running logs to mine the calling relationships of every component, extract the components running traces, and build dynamic behavior model. Then we compare theoretical composite model with dynamic behavior model, if they satisfy the behavioral congruence, then showed that the execution process has not been influenced, the behavior is normal, so the composite component can meet the requirements. Otherwise, it cannot achieve the requirements.

The dynamic behavior model construction has been studied in some literature. Ref.7 proposed the process mining model, generated the running traces through mining the running log, and then analyzed whether the running traces can be accepted by the theoretical model. If it can be accepted, then indicated that its behavior belongs to the theoretical model, otherwise, there exists non-consistent behavior. This method deficiency is that the running traces may be correspond to theoretic model, but the same traces may also be accepted by other model which is different from the theoretic model. This kind of question mainly is easy to appear in the model which has the choice structure; this method deficiency is also mentioned in Ref.22 and Ref.23. The bidirectional simulation methods, its principle is that generating process model according to the running traces firstly, then analyzing the congruence of theoretical model and the process model through bidirectional comparison. But this method deficiency is the result has high error rate. The reason is similar to the front method, the construction process model is not correct, sometimes is not comprehensive. In Ref.22 proposed that an analyzing method based on the behavior inspection, which mainly examines the key part congruence of two models, not overall system behavioral congruence. The methods can avoid causing the non-consistency of whole system because of unimportant part, but the deficiency cannot examine system behavior comprehensively, and it is also difficult to search key behavior.

To the behavioral congruence between theoretical model and the dynamic construction process model, it is insufficient only to take the key behavior into account. For comparing all behavior, the dynamic behavior model must be constructed comprehensively, and the dynamic model need satisfy determinacy in the construction process, like the question in Ref.4 mentioned. In order to determine the choice point in branch structure, we must analyze all running traces, and adopt certain strategies to locate the choice point. In order to make the dynamic behavior model construction as far as possible comprehensively, and correctly, the paper uses L* algorithm thought[24] to realize the dynamic process model construction. The L* algorithm thought is to build definite finite state machine to accept the language strings based on the known alphabet. Its merit is to learn the knowledge in the process of building model, aimed to the contradiction situation; counter-example is constructed, then amending the model to eliminate the counter-example, at last, a comprehensive and reasonable as far as possible dynamic behavior model is achieved. When constructing the dynamic model, the two judgments is need, one is whether the string U belongs to the regular language sets, the other is whether definite finite-state machine *C* constructed can accept these the language strings, that is, $L(C) = U$? If the key of each question is no, then we need produce counter-example *CE*.

Algorithm 3: Dynamic behavior model construction algorithm based on L* algorithm thought

Input: the component running traces

Output: the dynamic behavior model

(i) Let Petri net $N = \Phi$, select the trace string from extracting running traces, and take the character of traces as the transition of *N*. At the same time, add the place of *N* corresponding to the transition of *N*.

(ii) If the running traces are not end, then continue to select the traces. New traces will run in the behavior model constructed, if the model can accept the new selecting traces, then not handling the operation, else, goto (iii).

(iii) If $T \subset \vartheta(\sigma)$, here, $\vartheta(\sigma)$ is the string of trace σ ,T is the transition sets of *N*, then we add the transition $\vartheta(\sigma) - T$ to *N*, and renew the Petri net *N*.

(iv) if $T = \vartheta(\sigma)$ and *N* does not accept the $\vartheta(\sigma)$,

then we using the L* algorithm thought to generate a counter-example *CE*, $CE \in L(N) - U$, and using learning methods to eliminate the counter-example, the obtained model can accept the selecting string. Next, goto (ii), continue to select other traces, until the traces are end.

(v) Output the building Petri net model, which is the dynamic behavior model.

A comprehensive and reasonable as far as possible dynamic behavior model can be achieved based on the algorithm above mentioned, which laid the foundation for the behavioral congruence analyzing between dynamic behavior model and theoretical model. When comparing the behavior congruence of two Petri net models, we use the Petri net language theory, and analyze the equivalence of prefix languages[20] of two models.

4. A Case: E-Banking Simulation System

With the development of electronic and Internet technology, the financial system's business model and management has brought new changes. The form of monetary is changed from the physical currency to the electronic money. Service mode is changed from "people-to- person" on the bank counter to the "people-to-machine" dialogue model. Capital flows is changed from the entity certificate to the electronic certificate. These indicate that the banking industry begin to break through the traditional "reinforced concrete" style of marketing channels, and take part in electronic services, circulation and payment. The concept of banking changes from entities banks to E-Banking. "E-Banking" refers to a virtual bank which provides financial self-service for customers by the network and electronic terminals.

The current solution is workflow method oriented to SOA, and the trustworthiness research is the core in the process of integrating business processes. For example, the upgrading of one module may lead to unpredictable results for other interrelated models. At the same time, for there are many public business interfaces between banking system and interrelated corporation business system, and they lack the corresponding methods, so it leaves chance for malicious attack. Therefore, it is urgent to study effective computing techniques, which can not only make the software components distributed in every banking department constitute the new software rapidly and dynamically with the change of business demands, but also guarantee the consistency in behavior between the components and the composite software.

Some experts and scholars have analyzed and researched the technical aspects of these issues. Existing theories and methods are not specifically designed for the trustworthiness of E-Banking transaction system, but to solve certain aspects of trustworthiness, such as security, reliability, accuracy, etc, or to solve the certification of the object transactions. Scarce research is for the trustworthiness of E-banking transaction system itself. However, the research of the trustworthiness of E-Banking transaction system mechanisms is not just the integration of a variety of trustworthiness properties, because different trustworthiness properties may have conflicts with each other. Therefore, the above theory and methods are limited to protect the trustworthiness of E-Banking transactions system.

Based on the methods proposed in the paper, on the one hand, the identify certification of component can be solved by comparing the consistency between the specification description model with the inferred specification model. On the other hand, the deadlock or trap produced in the process of building business process may be diagnosed, and malicious components can be prevented from choreographing the composite software through dynamic behavior consistency analyzing.

The simulation analysis is in E-Banking simulation system platform, which can simulate some function of E-Banking system. In order to find problem, we select some representative components purposely, some components are not consistency between the specification descriptions with implementation, and some are non-consistent interaction behavior relativity.

We compare our methods with Methods 1 which is studied in Ref.22, in Fig.3, with the component increase, the success ratio of component function (success ratio=the function of success implementation/ component number) declines quickly, and the result indicates the success ratio of our methods is better than Methods 1. The reason is that component specification consistency and dynamic behavior congruence are taken into account in our methods.

In Fig.4, we analyze the relationship between the component numbers with the trust value. The trust value is degree of the effect conforming to user anticipation.

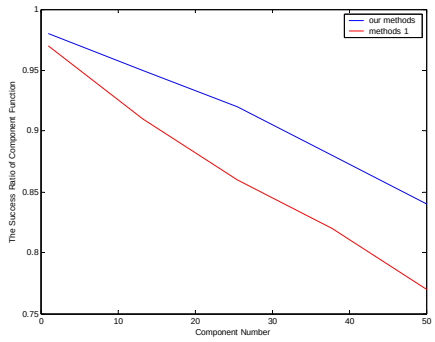

Fig.3. the relationship between the component numbers with the success ratio of component function.

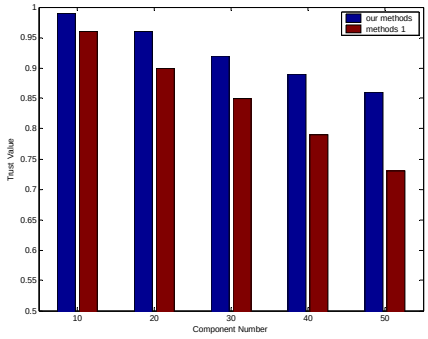

Fig.4.the relationship between the component numbers with the trust value.

The trust value of two methods all decrease with the component number increase, and the trust value of our methods is better than the methods 1, and our methods have little change when the component number reaching certain count.

5. Conclusions

The paper presents the behavior-aware trustworthiness analyzing methods of networked software. Firstly, in order to determine whether the component specification description is consistent with the component implementation, we present the algorithm of building inferred specification behavior model, and analyze the consistency between the inferred specification behavior models with component specification model. For component that satisfying specification consistency, the behavior certification center issues the behavior certificate to the component provider after the component checked. Next, the complex function demand needs many components with behavior certificate interaction to accomplish, for analyzing the component interaction behavior, the behavior relativity analyzing method is proposed based on the behavior theories of Petri net, aiming to analyze whether component behavior is influenced after component interaction. Finally, we extract the composite component running traces, and build the component dynamic behavior model, and compare the dynamic behavior model with the theoretic composite model. If they are behavioral congruence, it manifests that the component execution process is not influenced by outer environment.

Based on the theoretical analysis and experimental results, the innovation and advantage of the paper are: 1) Behavior-aware trustworthiness analyzing framework of networked software is presented, which can realize the software execution process to be manageable, controllable and prevention ability through analyzing software execution behavior. 2) The behavior model building methods of inferred specification is proposed, we can analyze the consistency between the specification descriptions of component with component implementation through comparing the component specification description model with the inferred specification behavior model. 3) For analyzing one component be influenced by other component in the process of component interaction, a determining algorithm of component interaction behavior relativity is given out. 4) Aimed to networked environment, we analyze the behavioral congruence between dynamic behavior models with theoretic composite model, and propose the building methods of dynamic behavior model.

In the future, we plan to study the trustworthy evaluation of networked software, and study the adaptation methods of non-consistent behavior relativity.

Acknowledgements

We would like to thank the support of the National Natural Science Foundation of China under Grant No.90818023 and No.90718012, the National High-Tech Research and Development Plan of China under Grant No.2009AA01Z401, the Program for Changjiang Scholars and Innovative Research Team in University, the Major State Basic Research Development Program of China under Grant No. 2010CB328100, the Anhui provincial Natural Science Foundation of China under Grant No.KJ2010B310.

References

1. H. Wang, Y. Tang and G. Ying, The trustworthy mechanism of internet software, *Chinese Science (E series),* **36**(10) (2006) 1156-1169.
2. J. P. Anderson, Computer security technology planning study. *ESD-TR-73-51*, Vol. I, AD-758 206, ESD/AFSC, Hanscom AFB, Bedford MA, October 1972.
3. High Confidence Software and Systems Coordinating Group. *High Confidence Software and Systems Research Needs.* USA. January 10, 2001.
4. Trusted Computing Group, TCG TNC Architecture for Interoperability Version 1.3. Apr. 28, 2008.
5. G. Bill, Trustworthy computing. *Microsoft Corporation*. July 18, 2002.
6. C. Lin. The study of trusted network, *Chinese Journal of computer*, **28** (5)(2005)751-758.
7. K. Liu, Z. G. Shan and J. Wang, The general study on the major project of trustworthy software, *Chinese Science Foundation,* **22**(3) (2008)145-151.
8. L. Zhou, G. Huang and H. Mei, Negotiation-enabled modeling and verification of architectural behavior of Internetware, *Journal of Software*, **19**(5) (2008)1099-1112.
9. G. J. Holzmann, The spin model checker. *IEEE Trans on Software Engineering*, **23**(5) (1997)279-295.
10. J. Lv and X. Ma, The study and improvement of Internetware, Chinese Science (E series), **36**(10) (2006) 1037-1080.
11. J. Yang, D. Evans and D. Bhardwaj, Perracotta: Mining temporal API rules from imperfect traces. Proc. 28th Int'l Conf. Software Eng., 2006, pp. 282-291.
12. R. Alur, P. Cerny and P. Madhusudan, Synthesis of interface specifications for Java classes. (POPL, 2005).
13. H. Doreswamy and M. N. Vanajaskhi, Similarity measuring approach for engineering materials selection, *International Journal of Computational Intelligence Systems*, **3**(1) (2010)115-122.
14. S. Sharon, Y. Eran and J. F. Stephen, Static specification mining using Automata-based abstractions, *IEEE Transactions on Software Engineering*, **34**(5)(2008) 651-666.
15. J. F. Stephen and Y. Eran, Effective type state verification in the presence of aliasing. *ACM Transactions on Software Engineering and Methodology*, **17**(2) (2008)9:1-34.
16. P. Georg and T. Roger, Intelligent concepts for the management of information in workflow systems, *International Journal of Computational Intelligence Systems*, **2**(4) (2009) 332-342.
17. L. Piroddi, R. Cordone, and I. Fumagalli, Selective siphon control for deadlock prevention in Petri nets, *IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans*,**38**(6)(2008)1337-1348.
18. A. Aybar and A. Iftar, Deadlock avoidance controller design for timed Petri nets using stretching, *IEEE Systems Journal*, **2** (2) (2008)178-188.
19. Y. Huang and M. Jeng, Deadlock prevention policy based on Petri nets and siphons, *International Journal of Production Research,* **39** (2) (2001)283-305.
20. C. Jiang, Behavior theory and its application of Petri net, (*Higher education press*, 2003).
21. T. Murata, Petri nets: Properties, analysis and applications. *Proc. IEEE*, **77**(4) (1989) 541-580.
22. M. P. Wil, D. A. Van and C. Ouyang, Conformance Checking of Service Behavior, *ACM Transactions on Internet Technology*, **8**(3) (2008), 13:1-30.
23. L. Davide, Automatic generation of software behavioral models, in proc.30th International Conference on Software Engineering (IEEE Press, 2008), pp.501-510.
24. S. P. Corina and G. Dimitra, Learning to divide and conquer: applying the L* algorithm to automate assume-guarantee reasoning, *Form Methods System Design*, **32**(2) (2008) 175–205.