Research on Online Monitoring and Analyzing of Interactive Behavior of Distributed Software

Junfeng Man
School of Information Science and Engineering, Central South University, Changsha, China
College of Computer and Communication, Hunan University of Technology, Zhuzhou, China
Email: mjfok@tom.com

Luming Yang[1], Lanjun Wan[2], Xiangbing Wen[2]
[1]School of Information Science and Engineering, Central South University, Changsha, China
[2]College of Computer and Communication, Hunan University of Technology, Zhuzhou, China

***Abstract*—The paper pays close attention to scenario and relationships between behavior and behavioral effects of distributed software at running time, presents a novel online monitoring and analyzing method for software behavior. Dynamic AOP monitoring technology is adopted to monitor interactive events related with business logic which are produced by the third party entities; Scenario-sensitive method is used to model complicated Interactive Behaviors (IBs) among these entities. By fusing real-time self-experience and pervious experience based on knowledge, the creditability of interactive entities is computed automatically. Multi-Entity Bayesian Network (MEBN) tool is adopted to construct reusable domain "knowledge fragment". If current scenario is similar to pervious one, then pervious one is reused; if there is no similar scenario, evidences gained from monitoring and pervious experience are fused to construct behavior model for this scenario. The combination of large and small knowledge reuse improves analysis efficiency of IBs. Above method is used to "Trusted Purchasing Network" that we develop, deceitful or fraudulent behaviors in trade process are online monitored and analyzed.**

***Index Terms*—distributed software, interactive behavior, behavior analyzing, scenario, multi-entity Bayesian network**

I. INTRODUCTION

The emergence of Internet makes running environment of software from static and close to dynamic and open. In order to adapt to this trend, software system architecture gradually changes from centralization to distribution [1]. In recent years, distributed software plays more and more important role in national economy, many of them (e.g., service systems in telecommunication, finance and medical treatment, traffic control system, e-commerce system) are melting into our daily work and life. However, distributed software are always out of order or failure, which brings negative impact on our daily work and life, e.g., the failure of e-commerce system will lead to economic loss, the failure of medical treatment even threats to life. With the scales of distributed software become more and more enormous and function more and more complicated, people pay special attention to software creditability (i.e., availability, reliability and security).

Software trust means that software system always runs according to the way that we set up [2]. The nature of software is to substitute for people to carry out certain behaviors. Software trust is mostly embodied in its behavior trust, which demands that running time behaviors and results of software system are always consonant with people's expectation [3]. Behavior trust isn't no factual basis, which needs to monitor IBs of software entities, and collect data related to trust. On this basis, the system can achieve online diagnosis, prediction and trust-evaluation, which helps to implement dynamic regulation for software behavior, and ultimately improve behavior creditability of software. It is obvious that monitoring and analyzing of IBs are in basic position in ensuring software trust.

In open and dynamic network environment, distributed software is loosely aggregated with several heterogeneous entities. Entity elements may enter and leave dynamically, these elements may interconnect, intercommunicate, collaborate and unite each other in terms with variously cooperative work way. Although software monitoring technology has undergone forty years' development, for the monitoring and analyzing of IBs of distributed software, it is still confronted with many challenges.

(1) What contents does system monitor? How is monitoring effectively implemented? Because the scale of distributed software is enormous and IBs are complicated, system can not and need not monitor all running time behavior. In my opinion, making clear monitoring contents and target and controlling monitoring granularity within reasonable range are indispensable. Monitoring mechanism can be integrated into target system with flexible, loose and transparent way; Monitoring target can be flexibly customized in running process of target system, and monitoring function can be opened or closed dynamically; To a certain degree, autonomous monitoring can be implemented; Monitoring

doi:10.4304/jsw.5.4.361-368

scale can be online extended with the extension of target system. In open and dynamic distributed software environment, designing and implementing an effective monitoring tool that satisfies above demands is a very important technological problem that we will emphatically solve.

(2) How is monitoring information fused with historical data to provide evidence for the analysis of IBs. In open and dynamic distributed software environment, the data monitored from multi-source may be represented with many forms. It is unpractical that information fusion is implemented by simple syntax matching. Advancing information processing from data to knowledge level, the realization is growing that sharing knowledge among distinct information systems requires first arriving at a common understanding of their respective semantics, and then formalizing that semantics in computable representations. Thus, computer can analyze and reason about IBs, proactively predict subsequently possible trend.

(3) In open and dynamic distributed software environment, how to solve the problem of uncertain knowledge between complicated interactive entities and their relationships. Uncertainty is ubiquitous to knowledge fusion. Almost any source of primary data carries some degree of uncertainty. Bayesian probability is a principled formalism for representing uncertainty and drawing inferences in the presence of uncertainty. Specifically, in a standard Bayesian Network (BN), all the hypotheses and relationships are fixed in advance, and only the evidence can vary from problem to problem. In open and dynamic distributed software environment, loosely-coupled interactive entities may be strange for each other, or the entities which ever have interacted may have new interaction in new scenario, numbers of interacting entities cannot be known in advance. Standard BN cannot flexibly represent complexity and uncertainty of interactive entity behavior. It is another technological problem to be solved that we find an effective method which effectively represents the uncertainty of complicated interactive entities and relationships, and provides support for online analysis and trend prediction of IBs by multi-source fusion of knowledge.

Because loosely-coupled entities in distributed software have their own profits, behavior strategies and rules, their running time behaviors have inherent laws, the collaboration of interactive entities makes them show some statistical characteristic in the mass at running time. Distributed software should be “monitored” and “grasped” in open and dynamic environment, the scenario and relationships between behavior and behavioral effects at running time are investigated, and Multi-Entity Bayesian Network (MEBN) tool is adopted to analyze running time behavior states and traces, behavior analyzing and predicting model is constructed, the intentions of interactive entities are inferred, and subsequently possible trend is proactively predicted.

The rest of this paper is organized as follows: Section 2 is related research and corresponding analysis; Section 3 introduces monitoring mechanism of IBs of distributed software; Section 4 illustrates online analyzing technology of IBs; In Section 5, above method and technology are used to “Trusted Purchasing Network” that we develop, deceitful or fraudulent behaviors in trade process are online monitored and analyzed, experiment results and corresponding analysis testify our theory.

II. Related Research and Analysis

A. Software Monitoring Technology

Software behavior monitoring refers to monitoring software running behavior, collecting behavior information and providing basic data for diagnosis, prediction and trust-evaluation. For widespread applications of distributed Software, its behavior monitoring attracts more and more attention. Many scholars are engaged in related researches and many mature monitoring technologies are constantly emerging.

In the aspect of monitoring mechanism, there are wrapper, interceptor, AOP method, reflection method, instrumenter and monitoring API. In the aspect of monitoring technology based on component wrapper, papers [4] research on a kind of running monitoring mechanism for distributed components. Component wrapper encapsulates monitored code, which helps to monitor performance, status and interactive events, and collects components’ interactive information. Paper [5] presents a kind of Behavior and Capture Technique (BCT), which uses component wrapper to capture running behavior of program automatically. The advantages of using component wrapper are no need to modify source code, and fit for third party components. Its obvious disadvantages for developers are to program lots of monitoring code manually, it is not fit for distributed software with large scales and complicated functions.

In the aspect of monitoring technology based on AOP, paper [6] applies AOP to running trace monitoring of software, which can inject into monitoring function of running behavior when system is running, and provide quantization evidences for system failure diagnosis. The monitoring logic and business logic are separate and loosely-coupled, which is convenient to construct a monitoring system whose scale can continuously increase to satisfy new demands.

The reflection middleware provides supports for monitoring system. DynamicTAO [7] is reflection CORBA software based on ORB reflection mechanism, which can monitor interactive information of distributed middleware based on CORBA.

Obvious disadvantages of above researches about software monitoring are as follows: 1) Above researches lack effective supports for IBs monitoring among entities. Some of them support the monitoring for IBs of components, but their monitoring granularity is too large to provide detailed IBs information. 2) Most of these researches only pay attention to monitoring themselves, and lack enough monitoring for business logic of software system. Furthermore, they mostly pay attention to unilateral trust of software themselves, such as the monitoring for availability, reliability and security. For

IBs monitoring of distributed software, we should pay more attention to the monitoring related to business logic except the monitoring of software themselves. 3) Most researches do not provide a set of favorable monitoring mechanism, and their monitoring logic and business logic are closely-coupled. These methods need to modify source codes and increase programming burden, and do not support to open or close monitoring mechanism when target system is running, are not suitable for IBs monitoring of distributed software with large scale and dynamic change. 4) Most of existing researches do not consider the characteristic of dynamic change of distributed network environments, lack effective collection and storage mechanism of monitoring information, which lead to heavy monitoring load and low monitoring efficiency.

B. Software Behavior Analyzing Technology

Paper [8] defines the concept of software behavior trust that IBs and results of entity elements can be predicted and controlled at running time, namely, behavior states can be monitored, behavior process can be analyzed, behavior results can be evaluated and predicted and exceptional behaviors can be controlled. Software behavior analyzing and predicting are very important parts of software creditability analysis. In this aspect, many researchers have had beneficial exploration.

Some people predicted software subsequent behaviors by referring historical behaviors. Based on past behavior patterns, Nielsen et al. computed maximum expectation of future software behaviors [9]. Mello et al. used neural network to analyze and predict the behaviors of application [10]. Bouguila et al. used statistical method of BNs to analyze and predict application accessing contents [11]. Above predicting methods have definite restriction, do not adapt to open, dynamic and complicated distributed software environment.

Some researchers analyzed and predicted behavior trust of entities with the models that were defined in advance. For example, Tian et al. used BNs to predict user behavior trust, their method might predict qualitative rating of behavior trust under multi-attribute [12]. Peng et al. presented a kind of distributed trust mechanism, based on bargaining history and iteration method, which could compute global buying and selling reputation of every node [13]. In open and dynamic distributed software environment, loosely-coupled interactive entities may be strange for each other, or the entities which ever have interacted may have new interaction in new scenario, numbers of interacting entities cannot be known in advance, the evidences gained from different scenarios are likely to be different. Above models are already fixed before behavior analysis and prediction, which are not suitable for open, dynamic and complicated distributed software environments.

Most of above models or methods adapt to conventional distributed software system. Though a few researches discuss the problems of software trust in current ones, they do not present perfect solution to running time behavior monitoring and analyzing. New software environments are faced with new problems, and a novel method should be presented to solve them. In my opinion, by fusing monitoring information with historical data, MEBN tool is used to construct behavior analyzing and predicting model for specific scenario, Multi-Entity Decision Graphs (MEDGs) are used to effectively analyze IBs, which provides solid foundation for whole management and flexible adjustment of distributed software.

III. Interactive Behavior Online Monitoring

For having right understanding for IBs, the definition of software behavior is presented. Software behavior is the sequence composed of Interactive Events (IEs), an IE is for subject to employ a service to an object, which is represented with formula: Event = { $e = S: f$ (O) | S: Subjects, f: Functions, O: Objects}. Here, e represents an event, S represents a subject, f represents a service, O represents an object. An event is composed of three elements: subject, object and employed service. For two events, if one of these three elements is different, two events are different.

In distributed components software, when a component as a subject accomplishes a certain functions, it needs to interact with other components by interface, which is defined as IEs of component. IEs of component can be understood as follows: behavior subject and object are all components of system. That the subject employs a service to an object means that a component provides service or send request to another component. For the interaction between two components, it reflects in their inner state transition from inner behavior view, and reflects in a series of call between them from external behavior view.

The IBs of components are divided into functional and non-functional IBs: 1) Functional IB is interactive activity between components to accomplish a certain function, which is related with business logic; 2) Non-functional IB is interactive activity between components to accomplish a certain non-functional property, which is not related with business logic, and involves log record, transaction processing and performance optimization.

A. Monitoring Target of IBs

According to different IBs monitoring types of components, the system can monitor different types of data. Basic information monitoring of IBs is to acquire data included in IEs and thread and performance information related with IEs execution. For the monitoring of validity, security, reliability, availability and timeliness, related data can be acquired from corresponding measure formulas. The system may selectively monitor IBs we are interested in according to different application scenarios.

B. Monitoring Requirement Management Framework

Monitoring requirement is to guide and confirm monitoring agent how to monitor business rules of IBs according to user intention and running environment change, which is the basis of system monitoring. The object of monitoring requirement management to ensure business rules to be conformed to, and supports dynamic

configuration, autonomous regulation and automatic deployment of business logic. Monitoring requirement management is a very important part of monitoring center module, its basic framework is showed in figure 1.

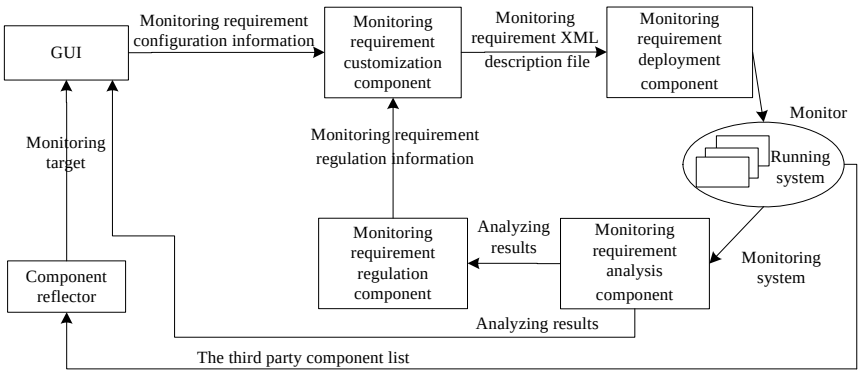

Figure 1. Monitoring requirement management framework

In figure 1, monitoring requirement management framework includes component reflector, monitoring requirement customization component, monitoring requirement deployment component, monitoring requirement regulation component and monitoring requirement analysis component. The management process of monitoring requirement is illustrated as follows: 1) According to the third party component list of target system, component reflector is used to extract inner structure information of the third party component and uncover monitoring target, which is convenient to configure and regulate monitoring requirement. 2) Manager may dynamically configure monitoring requirement, customize monitoring target and acquire monitoring configuration information by GUI in the running process of target system. 3) Monitoring requirement customization component uses XML file to store monitoring requirement configuration information and generate monitoring requirement description file. 4) Monitoring requirement deployment component parses monitoring requirement description file (XML), and automatically generates monitor with the help of code template. Monitor is dynamically deployed to corresponding node of target system, it can monitor IBs of components, collect the information of IEs, and store them into repository. Monitoring requirement analysis component takes charge of analyzing and processing these information. 5) Monitoring requirement regulation component autonomously regulates monitoring requirement according to analyzing results, generates regulation information of monitoring requirement, and transmits monitoring requirement to customization component. 6) Manager may dynamically reconfigure monitoring requirement by GUI according to analyzing results.

C. Monitoring information storage

Monitoring information storage mechanism solves the problem of which format and where IBs information of components collected is stored. The IBs information captured by monitors is often discrete and fragmenting, they should be organized according to standard and consistent IBs trace format before they are stored.

In order to accurately understand IBs trace, software behavior trace is defined: the events that the same subject produces within an observation interval are ordered according to occurrence time and recorded as event sequence. Software behavior trace is represented with formula: EventTrace = (∂ = S: $e_1 e_2 ... e_n$ | S: Subjects, e_1, e_2, … , e_n: Event), here, ∂ represents a behavior trace, S represents a subject, $e_1 e_2 ... e_n$ represents a string composed of events $e_1, e_2, ..., e_n$. The sequence of the string represents the sequence that events occur. Events are recorded as string according to occurrence time, which is software behavior trace. Conforming to this definition, component IBs trace is defined: interactive actions that the same component occurs within an observation interval are ordered according to occurrence time and recorded as event sequence. The format of component IBs trace is showed in figure 2.

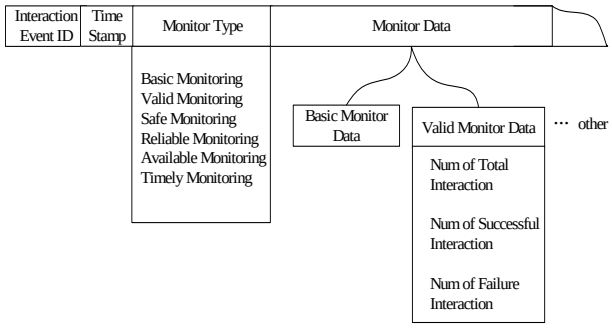

Figure 2. The format of component IBs trace

From figure 2, we know that component IBs trace is indexed with IEs ID and ordered with Time Stamp, different IEs monitoring types have different formats of monitoring data, e.g., monitoring data of IEs basic information includes event name, event type<request event or offering event>, sender<component and interface name that produces event>, receiver< component and interface name that receives event >, event input parameter<parameter name, parameter type, parameter value>, return results<return type, return value>, start time, end time. Corresponding formalization representation is {event 1, required event,

Subject<component 1, interface 1>, Object<component 2, interface 2>, Event Parameter<param 1, int, 100>, Return Result<int, 200>, 01/01/2009 00:00:00, 01/01/2009 00:00:01, other}.

In monitoring agent, monitors gain primary IEs information from monitoring event message sequence, and then send these information to monitoring event processor, which organizes these information into continuous and whole IBs trace, and stores them into local monitoring information repository.

IV. Interactive Behavior Online Analysis

In software creditability computing, the creditability of interactive entities is computed automatically by fusing real-time self-experience and pervious experience based on knowledge, which can objectively and instantaneously judge whether current interactive entities is trusted. We present scenario-sensitive method to model complicated entities and their interactive relationships. If current scenario is similar to pervious one, then pervious one is reused; if there is no similar scenario, evidences gained from monitoring and pervious experience are fused and "knowledge fragments" are reused to construct behavior model for this scenario. This new behavior model is stored into repository and convenient for reuse in similar scenario.

A. Multi-Entity Bayesian Network

The W3C responded to this limitation with the recently created Uncertainty Reasoning for the World Wide Web Incubator group (URW3-XG) [14]. The group's mission was to better define the challenge of representing and reasoning about uncertain information within the World Wide Web and its related technologies. The use of probabilistic reasoning enables information systems to derive benefit from uncertain, incomplete information, instead of being restricted to complete knowledge alone. This seems to be a promising prospect for the SW. One of the most promising approaches to deal with uncertainty in the SW is BNs, a graphical, flexible means of parsimoniously expressing joint probability distributions over many interrelated hypotheses. However, BNs have some limitations on representational power that restricts their use for the SW. Amongst these limitations are the fact that the number of variables has to be known in advance and the technique's lack of support for recursion. In order to address these shortcomings within the context of the SW, Costa proposed a Bayesian framework to probabilistic ontologies that provides a basis for representation and reasoning under uncertainty with the expressiveness required by SW applications [15]. This framework is based on the probabilistic ontology language PR-OWL, which uses MEBN [16] as its underlying logic. MEBN is a formalism that brings together the expressiveness of First-Order Logic (FOL) with BN's ability to perform plausible reasoning.

MEBN represents the world as comprised of entities that have attributes and are related to other entities. Knowledge about the attributes of entities and their relationships with each other is represented as a collection of MEBN fragments (MFrags) organized into MEBN Theories (MTheories). MFrag consists of both a set of Conditional Probabilistic Tables (CPTs) and FOL logical constraints that establish their validating conditions. The number of random variables (RV) is not fixed in a MEBN model. Instead, RVs are instantiated dynamically. An MTheory is a set of MFrags that satisfy certain FOL consistence conditions that guaranty the existence of a unique Joint Probabilistic Distribution (JPD) under its RVs. When all RVs are instantiated, all consistence conditions are satisfied, and all CPTs are generated, the MEBN yields a Scenario Specific Bayesian Network (SSBN). An SSBN is a normal BN. SSBN is stored into repository, which may be reused in similar scenario. Thus we implement two-stage knowledge reuse, MFrags are reused in constructing SSBN, SSBN is reused in analyzing similar scenario. This is a very important feature of the logic for modeling complex and intricate scenario.

"Trusted Purchasing Network" online business system is composed of 14 Mfrags (figure 3). Each of these eleven MFrags represents the probability information about a group of their respective RVs. Collectively, the group implicitly expresses a JPD over truth-values of sets of FOL sentences. That is, probability distributions are specified locally over small groups of hypotheses and composed into globally consistent probability distributions over sets of hypotheses. MEBN theories extend ordinary BNs to provide an inner structure for RVs. RVs in MEBN theories take arguments that refer to entities in the domain of application. This is because an MFrag is just a template, in other words, it does not represent individuals RVs, but a class of RVs. The values of its states appear only when the MFrag is instantiated.

MTheories includes three kinds of nodes: context node, input node and resident node. The context nodes are Boolean variables that represent conditions that have to be satisfied so that the probabilistic distribution of the resident nodes applies. Their possible values are: True (the condition is satisfied), False (the condition is not satisfied), and Absurd (a condition expression does not make sense). Input nodes are variables that influence the probabilistic distribution of its child resident nodes, but their distributions are defined within their own MFrags. In other words, in a complete MTheory, every input node must be a resident node in another MFrag, where its probabilistic distribution will be defined. Resident nodes have the local probabilistic distributions defined in that MFrag, including the probabilistic dependence on its parent values (that can be input or resident nodes). A node can have a list of arguments in parenthesis, which are replaced by unique identifiers of domain entities when the net is instantiated.

Another advantage of MEBN is to support recursion when constructing BNs. Its obvious difference with dynamic BNs is to execute recursion for part nodes, which decreases complexity of constructing BNs and improves inference efficiency. Figure 4 is the Mfrag AddtoCart operation, whose the second parameter is

orderable, i.e., AddtoCarts(OR123, T1) is computed on condition that AddtoCarts(OR123, T0) is known.

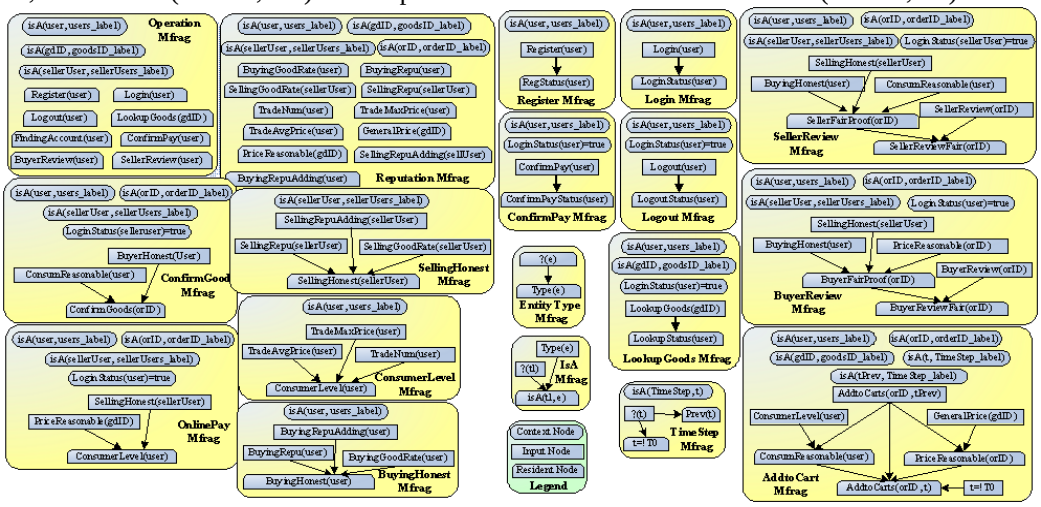

Figure 3. The MTheory of "Trusted Purchasing Network" online business system

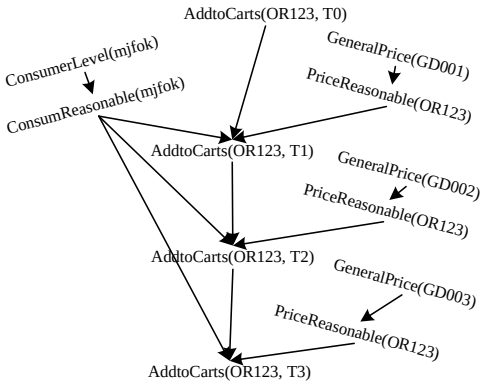

Figure 4. AddtoCarts SSBN with recursion

B. Decision-making and inference

MEDGs extend MEBN logic to support decision making under uncertainty. MEDGs are related to MEBNs in the same way influence diagrams are related to BNs. A MEDG can be applied to any problem that involves optimal choice from a set of alternatives subject to given constraints. When a decision MFrag (i.e. one that has decision and utility nodes) is added to a generative MTheory, the result is a MEDG.

The MTheory depicted in Figure 3 is a generative MTheory, which provides prior knowledge that can be updated upon receipt of evidence represented as finding MFrags. In a BN model, assessing the impact of new evidence involves conditioning on the values of evidence nodes and applying a belief propagation algorithm. When the algorithm finishes, beliefs of all nodes, including the node(s) of interest, reflect the impact of all evidence entered thus far. This process of entering evidence, updating beliefs, and inspecting the posterior beliefs of one or more nodes of interest is called belief propagation. Usually, the belief propagation process is carrying on answering probabilistic queries. Whereas BNs are static models that must be changed whenever the situation changes (e.g. number of buyers, time recursion, etc.), an MTheory implicitly represents an infinity of possible scenarios. Figure 5 illustrates the scenario that two users(users1 and user2) earn reputation for user3 by deceitful trade, thick arrows represent the process of decision-making.

MEBN inference begins when a query is posed to assess the degree of belief in a target RV given a set of evidence RVs. It is started with a generative MTheory, add a set of finding MFrags representing problem-specific information, and specify the target nodes for the query. The first step in MEBN inference is to construct the SSBN, which can be seen as an ordinary BN constructed by creating and combining instances of the MFrags in the generative MTheory. Next, a standard BN inference algorithm is applied. Finally, the answer to the query is obtained by inspecting the posterior probabilities of the target nodes.

V. DECEITFUL OR FRAUDULENT BEHAVIORS ANALYZING

A. Online monitoring tool

We have developed visual monitoring tool for monitoring the third part components. Figure 6 is the GUI of monitoring requirement configuration. Component reflector is used to extract inner structure information of components, and these information is showed in GUI of monitoring requirement configuration, user may configure monitored targets. Running time IEs information of monitored targets are saved automatically, which provide basic data for online analysis of IBs.

B. Deceitful analyzing of AddtoCarts behavior

AddtoCarts IB is composed of several AddtoCarts events. For 'PriceReasonable' of every added goods, 'ConsumReasonable' of this user, the creditability of AddtoCarts IB is computed. Formula (1) is used to compute 'PriceReasonable', formula (2) is used to compute the creditability of AddtoCarts IB after an AddtoCarts event. The process of computing posterior probability of the creditability of IBs with SSBN is the process of fusing real-time self-experience and pervious experience based on knowledge.

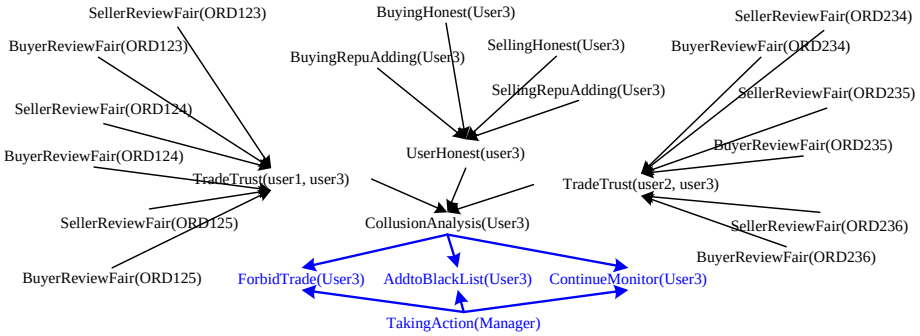

Figure 5. The MEDGs of earning reputation by deceitful trade

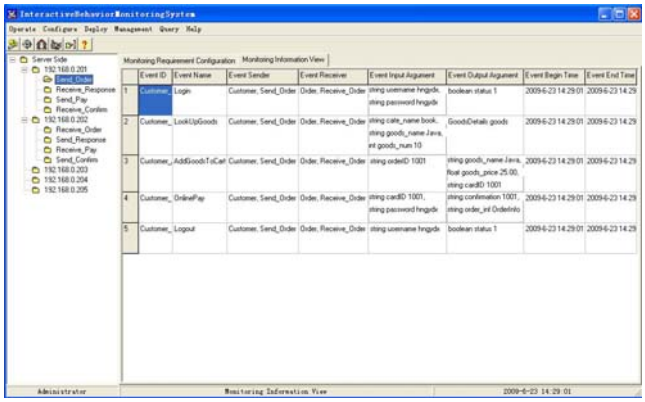

Figure 6. The GUI of Monitoring requirement Configuration

$$\text{PriceReasonable} = 1 - \text{Min}\left(\frac{|\text{CurrentPrice-GeneralPrice}|}{\text{GeneralPrice}}, 1\right) \quad (1)$$

$$\text{AddtoCarts}(t) = \text{AddtoCarts}(t-1)*W1 + \text{PriceReasonable}*W2 + \text{ConsumReasonable}*W3 \quad (2)$$

$$W1+W2+W3=1$$

Figure 7 is the analysis of practical AddtoCarts IBs. In the process of AddtoCarts, the trust vaule of AddtoCarts IBs constantly fluctuates. It this value is no less than threshold, AddtoCarts operation may continue, or else, the system will send early alarm to user, which may warn sellers' reputation or price reasonableness.

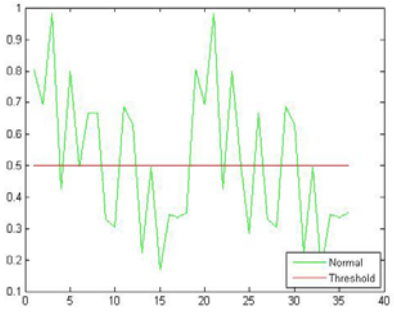

Figure 7. The analysis of practical AddtoCarts IBs

We analyze four types of creditability of AddtoCarts IBs. User1 has good pervious reputation, he has an honest trade this time; User2 has bad pervious reputation, he has a deceitful trade this time; User3 has good pervious reputation, he has a deceitful trade this time; User4 has bad pervious reputation, he has an honest trade this time. Trusted threshold is set to 0.5. From figure 8, we can see that this online analyzing method of IBs can accurately and instantaneously identify deceitful or fraudulent behaviors.

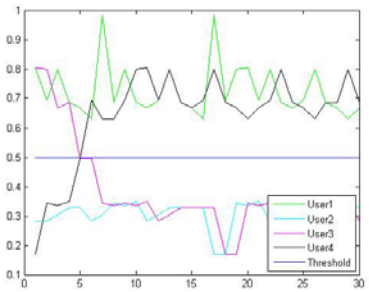

Figure 8. Creditability analysis of four types of AddtoCarts IBs

C. The analysis of earning reputation by deceitful trade

In figure 5, the value of 'TradeTrust' is computed with formula (2). Here, PF is Penalty factor, if user has continuous dishonest trade, he will be punished, his TradeTrust value will decrease very quickly. 'UserHonest' is computed with formula (4), 'CollusionAnalysis' is computed with formula (5).

$$TradeTrust(n) = W4*TradeTrust(n-1) + W5*(Sell\,\text{Re}viewFair + Buyer\,\text{Re}view)/2*(1-PF)^{n-1} \quad (3)$$
$$(Sell\,\text{Re}viewFair \le 0.5 \text{ and } Buyer\,\text{Re}view \le 0.5)$$
$$W4+W5=1$$

$$UserHonest = W6*BuyingHonest + W7*SellingHonest + W8*(1-BuyingRepuAdding) + W9*(1-SellingRepuAdding) \quad (4)$$
$$W6+W7+W8+W9=1$$

$$CollusionAnalysis = W10*TradeTrust(user1,user3) + W11*TradeTrust(user2,user3) + W12*UserHonest \quad (5)$$
$$W10+W11+W12=1$$

$$TakingAction = \begin{cases} ForbidTrade & \text{CollusionAnalysis} \le 0.3 \\ AddtoBlackList & \text{CollusionAnalysis}>0.3 \text{ and CollusionAnalysis}<0.7 \\ ContinueMonitor & \text{CollusionAnalysis} \ge 0.7 \end{cases} \quad (6)$$

Figure 9 shows the collusion analysis for four kinds of trade process. At first, honest values of Type1 and Type3 are greater than threshold (0.5), there is no collusion to earn reputation, honest values of Type2 and Type4 are less than threshold, there are possibly deceitful behaviors of collusion. The system will send early alarm. With the

increase of trade number, posterior probability of collusion analysis is also constantly changing, if their values are less than threshold, the system will send early alarm. The system makes decision according to collusion analysis results. There three kinds of strategies: ContinueMonitor, AddtoBlackList and ForbidTrade. This method can execute real-time analysis for collusion in the process of trade, analyzing results guide system to take appropriate measure, which guarantees trade to be secure and reliable by the greatest extent.

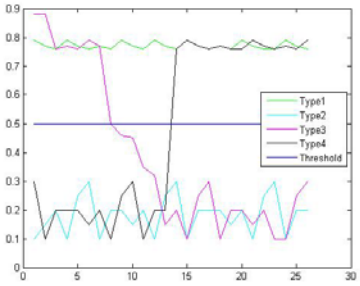

Figure 9. Creditability analysis of earning reputation by deceitful trade

VI. CONCLUSION

Open and dynamic distributed software system loosely aggregates several heterogeneous entities. Entity elements may enter and leave dynamically, their IBs are complicated and changeful. How to monitor and analyze IBs of distributed software is a very important scientific problem that has academic meaning and application value. We develop a set of online monitoring software for the third part components. Using MEBN tool, IBs analyzing efficiency is improved by reusing large and small granularity knowledge.

In subsequent research, we will improve the efficiency of SSBN construction and query; continue consummating formalization representation of behavior; pay more attention to solve the problem of inconsistent knowledge in the process of behavior analysis and inference; constantly enrich rules and repository to implement unsupervised monitoring and analyzing for IBs.

ACKNOWLEDGMENT

The financial supports from the national natural science fund of China under the grant No. 60773110, post-doctoral science fund of China under the grant No. 20080440216, natural science fund of Hunan province under the grant No. 09JJ6087 is gratefully acknowledged.

REFERENCES

[1] F.Q. Yang, H. Mei and J. Lv. “Some discussion on the development of software technology”. Chinese of Journal Electronics, 2002, vol. 30, No 12, pp. 1901-1906.

[2] H. W. Chen, J. Wang and W. Dong. “Trusted software model and security evaluation in telecom field with open network”. Chinese of Journal Electronics, 2003, Vol. 31, No. 12, pp. 1933-1938.

[3] H. M. Wang, Y. B. Tang and G. Yin. “Trust mechanism of Internet software”. Science in China Ser. E Information Sciences, 2006, vol. 36, No. 10, pp. 1156-1169.

[4] P. Herrmann and H. Krumm. “Trust-adapted enforcement of security policies in distributed component-structured applications”. Proceedings of the 6th IEEE Symposium on Computers and Communications, Hammamet, Tunesia, 2001, pp 2-8.

[5] L. Mariani and M. Pezze. “A technique for verifying component-based software”. Electronic Notes in Theoretical Computer Science, 2005, vol. 116, No. 19, pp. 17-30.

[6] P. Avgustinov, J. Tibble and E. Bodden. ”Aspect for trace monitoring - formal approaches to testing systems and runtime verification”, Seattle, WA, USA, 2006, pp. 20-39.

[7] F. Kon, M. Román and P. Liu. “Monitoring, security, and dynamic configuration with the dynamicTAO reflective ORB”. Proceedings of the IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing (Middleware'2000), New York, 2000, pp. 121-143.

[8] C. Lin, L. Q. Tian and Y. Z. Wang, “Research on user behavior trust in trustworthy network”, Journal of Computer Research and Development, vol. 12, No.12, Dec. 2008, pp. 2033-2043.

[9] M. Nielsen and K. Krukow, “A Bayesian model for event-based trust”, Electronic Notes in Theoretical Computer Science (ENTCS), vol. 172, No.4, 2007, pp. 499-521.

[10] R. Mello, L. Senger, and L. Yang, “Automatic text classification using an artificial neural network”, High Performance Computational Science and Engineering, vol. 17, No.9, 2005, pp. 1–21.

[11] N. Bouguila, J. H. Wang and A. B. Hamza. “A Bayesian approach for software quality prediction. 2008 4th International IEEE Conference "Intelligent Systems", 2008, pp. 49-54.

[12] L. Q. Tian and C. Lin. “A kind of game-theoretic control mechanism of user behavior trust based on prediction in trustworthy network”. Chinese Journal of Computer. 2007, vol. 30, No. 11, pp. 1930-1938.

[13] D. S. Peng, C. Lin and W. D. Liu. “A distributed trust mechanism directly evaluating reputation of nodes”. Chinese Journal of Software. 2008, vol. 19, No. 4, pp. 946-955.

[14] K.J. Laskey, K.B. Laskey and P.C.G. Costa. “Uncertainty reasoning for the world wide web Incubator group charter (W3C Incubator Activity) (2007)”, www.w3.org/2005/Incubator/urw3/charter

[15] P.C.G. Costa. “Bayesian semantics for the semantic web”. PhD thesis, Department of Systems Engineering and Operational Research, George Mason University, 2005, pp. 60-100

[16] K.B. Laskey. “MEBN: a language for first-order Bayesian knowledge bases”. Artificial Intelligence, 2007, vol. 172, No. 2, pp. 172 – 225

Junfeng Man Born in Suihua, China, on January, 8, 1976. He received the master degree in Computer Software and Theory from Yanshan University in 2003. He is presently working on his PhD in College of Information Science and Technology of Central South University. He is associate professor in College of Computer and Communication of Hunan University of Technology. His research interests include trust software, pervasive computing. Contact: mjfok@tom.com.