Diagnosis of Internetware Systems Using Dynamic Description Logic

Kun Yang, Weiqun Cui, Junheng Teng, Chenzhe Hang

Diagnosis of Internetware Systems Using Dynamic Description Logic

Kun Yang, Weiqun Cui, Junheng Teng, Chenzhe Hang

Chinese National Institute of Metrology, Beijing, China
yangkun@nim.ac.cn

Abstract. This paper proposes a kind of multi-agent based internetware system architecture, and introduces a diagnoser that can perform on-line diagnosis by observing the behaviors of it. This diagnoser brings semantic description to the procedure of states conversation of the system to be diagnosed by using dynamic description logic, which makes it possible to make use of more knowledge to analyze the failure further.

Keywords: diagnosis; internetware; dynamic description logic; discrete-event

1 Introduction

The problem of failure diagnosis has received considerable attentions in the literature since detecting and isolating failures play an important role in building trustworthy software systems. There are two main methods in the research area of failure diagnosis: expert system-based method and model based method. The problem faced by the former is it is difficult to extract heuristic rule from expert knowledge and it is area-dependent that means the diagnostic rules have to change with the changing of the areas. As for the latter, it models the features of the object to be diagnosed and records the deep knowledge of them, and makes the diagnostic object model and the diagnostic process independent of each other.

Model based method doesn't depend on specific modeling language, and qualitative deviation, qualitative differential equations, process algebras, Bayesian networks, discrete event system method and Petri nets[1, 2] are all be used to model and analyze the diagnostic objects. Among them, the research about discrete event system modeling is the most active one. In [3], diagnostic object is modeled as a global finite state automata, and failure is modeled as series of state transferring that can't be observed, the diagnoser predicts current state of the system and judge whether a specific failure happens based on the event sequence observed. Based on the global model in [3], [4] proposes a distributed diagnostic strategy and protocol. [5] proposes a method to construct distributed diagnostic model for large scale discrete event system when global model can't be obtained and apply it to the diagnosis of telecommunications networks. Although the diagnostic objects of these works are physical system that is different from our internetware system, their modeling method for discrete event sys-

tem and the strategy of reducing the diagnostic searching space provide us a good reference and guidance.

As a novel software paradigm, Internetware's failure diagnosis problem is urgent with its environment becoming open, dynamic and difficult to control. As a dynamic system, it has a definite boundary and can be abstracted as a discrete event system to some extent, which makes it possible to observe its behaviors at some granularity and research its failure diagnosis under the discrete event system framework. Current discrete event system modeling methods focus more on the state transition, while more proper semantic to the process of state transition will improve its diagnosis for following reasons:

- Software failure is semantically related with an application. People find that the software failures often happen for the components don't follow the design fully and execute the behaviors forbidden by designers or programmers, which means they violate the semantic of the system.
- Failure self-recovery needs semantic information. Because of the limitation of the observation method, the granularity of diagnosis result of traditional discrete event system is large, and the diagnostic searching space is huge. Lacking related semantic description makes it difficult to determine the exact failure state and recover that failure at the next step.

In order to bring proper semantic to the diagnostic process, this paper proposes a dynamic description logic based failure diagnosis method under the discrete event system framework, where event is modeled as an action, and state is modeled as a set of description logic simple formulas with their variables assigned specific values. Based on the analysis and reasoning to the formulas about a failure state, the failure location and type can be determined and its self-recovery becomes possible.

2 Multi-agent based internetware system

Ref [6] summarizes the software technology trend in the open, dynamic and variable internet environment and proposes the concept of internetware. Compared with classical software system, it has some distinctive features, such as the intelligentization of software entity, the separation of software cooperation and the autonomization of system management and so on. Based on these features, we design an Internetware system whose architecture is illustrated as Fig.1 below.

There are two layers in the architecture above: the control layer and the target layer. The former mainly includes the sensor that perceives the changes of the environment and the effector that adjusts the actions of the system. The latter includes classic software system and its environment. Agent owns features needed by Internetware so it is the best basic component for Internetare system. While it is not a software development technique in mainstream currently, and most Web services doesn't developed by agent technique, so we package these web services by agent to map them to a multi-agent system, so as to combine them to finish a task by the coop-

eration of agents without modifying them. The diagnostic object here is a software system conforming to the architecture introduced.

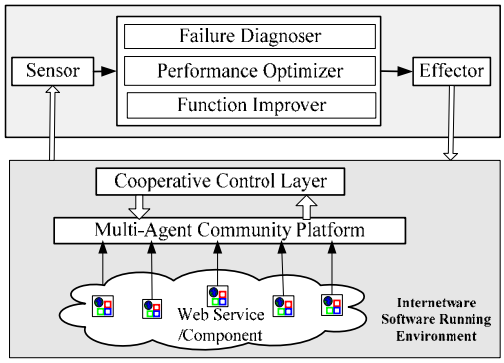

Fig. 1. The architecture of the Internetware System

3 D-ALCO based failure diagnosis

3.1 D-ALCO Brief Introduction

Based on description logic ALCO, [7] proposes dynamic description logic D-ALCO by combining description logic, dynamic logic and theory of action. As a kind of formal tool for knowledge representing, D-ALCO has clear semantic feature and can provide not only decidable reasoning service, but also representing and reasoning for dynamic process and operating mechanism effectively.

The basic symbols in D-ALCO include ①Concept set N_C; ②Role set N_R; ③ Individual set N_I; ④Individual Variable set N_{IV}; ⑤Atom action set N_A; ⑥Concept constructors: { }, ¬, ⊔ and ∀; ⑦Formula constructors:¬, ∨ and < >; ⑧Action constructors: ∪, ;, * and ?; ⑨Some other symbols include≡, ( ) and ','. With these symbols, roles, concepts, formulas and actions can be constructed.

Def 1 for a given TBox $\mathcal{T}$, $\alpha(v_1, \ldots, v_n) \equiv (P_\alpha, E_\alpha) \equiv (P_\alpha(v_1, \ldots, v_n), E_\alpha(v_1, \ldots, v_n))$ is defined as an atom action. Where

1. $\alpha \in N_A$, is the name of the atom action defined;
2. $(v_1, \ldots, v_n)$ is the finite sequence of all individual variables appeared in P and E;
3. P_α is the finite set consisted by simple formulas, which denotes the precondition that must be satisfied before the action performs;
4. E_α is the finite set consisted by simple formulas, which denotes the result after the action's performing;
5. P_α and E_α satisfy the constraint that for any $\varphi \in E_\alpha$, $\varphi^\neg \in P_\alpha$ is satisfied.

Def 2 *D-ALCO* is a triple $M = (\Delta, W, I)$, where Δ is a non-empty set of individual which acts as the discourse domain of the model; W is the set of possible world; I gives an explanation $I(w) = (\Delta, {}^{I(w)})$ for each possible world w in W.

Def 3 for any $M = (\Delta, W, I)$, $\gamma : N_{IV} \to \mathcal{I}^I$ is called one designation based on M.

Any designation γ gives an assignment for each individual variable in N_{IV} and designates them as individuals in $\mathcal{I}^I$. If $v \in N_{IV}$ and $\gamma_1, \ldots, \gamma_n$ is all the designations based on M then the explanation domain for v can be denoted as $\mathcal{D}(v)=\cup_{1\le i\le n}\{\gamma_i(v)\}$. For any $M = (\Delta, W, I)$, an explanation I and a designation γ can determine a mapping $\Gamma_{M,\gamma}$ from simple formula set $\varepsilon(v_1, \ldots, v_n) = \{\psi_1(v_1), \ldots, \psi_n(v_n)\}$ to Boolean value {True, False}, which can be denoted as: $\Gamma_{M,\gamma}(\varepsilon(v_1, \ldots, v_n))=\varepsilon(v_1, \ldots, v_n)^{I,\gamma} = \{\psi_1(v_1), \ldots, \psi_n(v_n)\}^{I,\gamma} = \psi_1(v_1)^{I,\gamma} \wedge \ldots \wedge \psi_n(v_n)^{I,\gamma}$.

3.2 D-ALCO Based System Modeling

Under the framework of discrete event system, the multi-agent based internetware system can be modeled as a finite state automata $G=(X, \Sigma, \delta, X_0)$, where X denotes the finite set of state; Σ denotes the finite set of event; $\delta \subseteq X \times \Sigma \times X$ denotes the finite set of state transition; $X_0 \subseteq X$ denotes the set of initial states. The states and events in G have no semantic information, now we can model them by D-ALCO as follows.

Assume $(v_1, \ldots, v_n)$ is the finite sequence of individual variables involved in G, where $v_i \in N_{IV}$, $1 \le i \le n$. The state space X of G corresponds to the Cartesian produce of the explanation domain of all individual variables, that is $X := \prod_{i=1}^{n} D(v_i)$. Let $\Theta(v_1, \ldots, v_n)$ is the set composed by simple formula set $\varepsilon(v_1, \ldots, v_n)=\{\psi_1(v_1), \ldots, \psi_n(v_n)\}$. For each $\varepsilon \in \Theta(v_1, \ldots, v_n)$, there is only one designation γ for the finite sequence $(v_1, \ldots, v_n)$, which satisfies the function $\Gamma_{M,\gamma}(\varepsilon(v_1, \ldots, v_n):X \to \{\text{True, False}\}$ is True. Let the explanation domain for the finite sequence $(v_1, \ldots, v_n)$ under any state $x_i \in X$ as a designation γ, then with this γ, the simple formula set ε that makes the $\Gamma M, \gamma(\varepsilon(v_1, \ldots, v_n):X \to \{\text{True, False}\}$ is true corresponds to that state x_i, which means there is a one-to-one correspondence between simple formula set Θ and state set X. The simple formula set $\varepsilon \in \Theta(v_1, \ldots, v_n)$ with corresponding designation γ that corresponds to any state $x_i \in X$ can be recorded as $\varepsilon_\gamma(x_i)$, then the initial state x_0 in G can be recorded as $\varepsilon_\gamma(x_0)$.

An atom action α in D-ALCO is used to model an eventσ, and the finite set of atom action N_A is used to model the finite set of eventΣ. Based on the correspondence between simple formula set Θ and state set X, the transitions of the states can be extended to the changing of the formula set. A state transition in G can be described by using an atom action, and the form of atom action is as follows:

$$\alpha(v_1, \ldots, v_n) \equiv (P_\alpha, E_\alpha) \equiv (P_\alpha(v_1, \ldots, v_n), E_\alpha(v_1, \ldots, v_n))$$

where an action $\alpha \in N_A$ denotes an event; P_α and E_α are both finite sets of simple formula that denote the precondition before α performing and the effect after α performing respectively. The change caused by action α to the simple formula set can be described as $\rho_\alpha(\varepsilon)::= (\varepsilon \backslash E_\alpha\neg) \cup E_\alpha$, and that to the individual variables $(v_1, \ldots, v_n)$ can be described as $\Gamma_{M,\gamma}(\rho_\alpha(\varepsilon(v_1, \ldots, v_n))::= \Gamma_{M,\gamma}((\varepsilon(v_1, \ldots, v_n) \setminus E_\alpha\neg) \cup E_\alpha)$. For an event σ denoted by an action $\alpha = (P_\alpha, E_\alpha)$, if $\delta(x_j, \sigma)=x_k$ is satisfied in G, then $\varepsilon_\gamma(x_k) \supseteq (\varepsilon_\gamma(x_j) \backslash E_\alpha\neg) \cup E_\alpha$ and $\Gamma_{M,\gamma}(\varepsilon_\gamma(x_j))::= \Gamma_{M,\gamma}(\varepsilon_\gamma(x_k))=\text{True}$.

The process of modeling above can be illustrated intuitively as Fig.2 below.

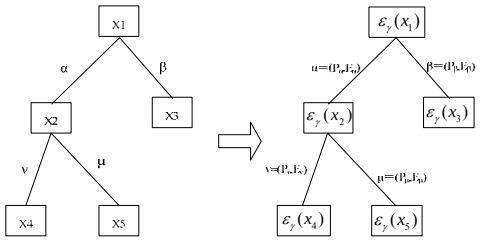

Fig. 2. Illustration of the D-ALCO modeling

3.3 D-ALCO Based Diagnoser Construction

Failure diagnoser is a finite state automata constructed based on G_e=(X×Δ, ∑, δ_e, X_{e0}) that is obtained after extending the *D-ALCO* model of G introdued above. It is used to observe the system online and diagnose the failure based on those observations. Its model is as follows:

$$G_d = (Q_d, \sum_o, \delta_d, q_0)$$

Where $Q_d, \sum_o, \delta_d, q_0$ denote the state space, event set, transfer function and initial state of the finite state automata of failure diagnoser. G_d records all the states that the system can reach starting from the initial state and experiencing the same event observation sequence. For the existence of unobservable event, there may be several state transitions under the same event observation sequence, so $Q_d \subset 2^{\Theta\times\Delta}$, here Θ has the same meaning with the definition before, meaning all the sets of simple formula $\varepsilon_\gamma(x_i)$ that corresponding to the state set *X* in *G*; $\Delta=\{N\} \cup 2^{\Delta_f}$ which means the set of state label, where N means normal state and Δ_f means the failure category label set. $\sum_o$ denotes the observable event set in the system G. $q_0=(\varepsilon_\gamma(x_0),\ \{N\})$, which is the same as the initial state X_{e0} of G_e.

For any state $q_d \in Q_d$, it can be denoted as $q_d=\{(\varepsilon_\gamma(x_1), l_1), \cdots, (\varepsilon_\gamma(x_n), l_n)\}$, where $x_i \in X, l_i \in \Delta$. δ_d can be defined as $\delta_d \subseteq (Q_d \times \sum_o \times Q_d)$. For any $\alpha \in \sum_o$, $(q_d,\ \alpha,\ \widetilde{q_d}) \in \delta_d$ is satisfied, if for any $(\varepsilon_\gamma(x_i), l_i) \in q_d$, there is a route s in G_e satisfying $\delta((\varepsilon_\gamma(x_i), l_i),\ s)=(\varepsilon_\gamma(x_j), l_j)$ and M(s)=α, then $(\varepsilon_\gamma(x_j), l_j) \in \widetilde{q_d}$.

3.4 Distributed Diagnoser and Diagnostic Strategy

Web service system is distributed. Even for a centralized system, it is divided into several subsystems to solve problem cooperatively for the consideration of the communication between agents, so the distributed diagnostic strategy is important. The distributed diagnoser and its diagnostic strategy can be applied for several distributed sites, but here we only involve two sites for the illustrative purpose.

The event set $\sum=\sum_o \cup \sum_{uo}$ and the failure event $\sum_f \subseteq \sum_{uo}$. For the two distributed diagnoser, assume their observable event sets are $\sum_{o1}$ and $\sum_{o2}$ respectively and $\sum_o=\sum_{o1} \cup \sum_{o2}$, then their local unobservable sets are $\sum/\sum_{o2}$ and $\sum/\sum_{o1}$. With the

enlarging of the unobservable set, the possible state set that each diagnoser can reach under the same observable event sequence enlarges too, which makes the state space for diagnostic searching greater and makes it more difficult to locate failure and need higher computing cost. In order to limit the diagnostic searching space by making full use of known information, we use following strategies during the process of distributed diagnosis.

- *Recording the path of state transferring:* for the limitation of observation methods, different event tracks appear the same observation feature. Diagnoser records all the states a system can reach under the same observation, while if it can record the path of state transferring, then it can find impossible paths and delete the corresponding states on those paths, so as to reduce the diagnostic searching space.
- *The cooperation among distributed diagnosers:* although the information is local and limited for each distributed diagnoser, they can complement information and constrain mutually to limit the global diagnostic result to a smaller state space.

Now assume the model of a distributed diagnose is as follows:

$$G_{di}^{e} = (Q_d^e, \sum_{oi}, \delta_{di}^{e}, q_0^e)(\text{here } i \in \{1,2\})$$

Based on the first strategy, we record the path of state transferring in G_{di}^{e}, but only the direct precursor state of any state for the consideration of storage space and computational complexity. Q_d^e is obtained by extending Q_d introduced in previous section: let the state member of each state includes not only the state itself, but also its direct precursor state, that is $Q_d^e \subset 2^{(\Theta\times\Delta)\times(\Theta\times\Delta)}$. Let the direct precursor state of the initial state is itself, then $q_0^e = \left\{\left(\left(\varepsilon_\gamma(x_0), \{N\}\right), \left(\varepsilon_\gamma(x_0), \{N\}\right)\right)\right\}$ and for any $q_d^e \in Q_d^e$, $q_d^e = \{((\varepsilon_\gamma(x_1'), l_1'), (\varepsilon_\gamma(x_1), l_1)), \cdots, ((\varepsilon_\gamma(x_n'), l_n'), (\varepsilon_\gamma(x_n), l_n))\}$ $(l_i \in \Delta)$, where $(\varepsilon_\gamma(x_1'), l_1')$ and $(\varepsilon_\gamma(x_n'), l_n')$ correspond to the direct precursor states of the states corresponding to $(\varepsilon_\gamma(x_1), l_1)$ and $(\varepsilon_\gamma(x_n), l_n)$ in G_e respectively.

For the distributed diagnoser $G_{di}^{e}(i \in \{1,2\})$, its unobservable event set is $\sum/\sum_{oi}$, and for any event $\beta \in \sum/\sum_{oi}$, $M(\beta) = \emptyset$ is satisfied. δ_{di}^{e} is defined as follows:

$$\delta_{di}^{e} \subseteq (Q_d^e \times \sum_{oi} \times Q_d^e)$$

For any $\alpha \in \sum_{oi}$, $(q_d^e, \alpha, \widetilde{q_d^e}) \in \delta_d$ is satisfied. For any $\left(\left(\varepsilon_\gamma(x_i'), l_i'\right), \left(\varepsilon_\gamma(x_i), l_i\right)\right) \in q_d^e$, if there is a event path s started from the state corresponding to $(\varepsilon_\gamma(x_i), l_i)$ that satisfies $\delta_e\left(\left(\varepsilon_\gamma(x_i), l_i\right), s\right), \left(\varepsilon_\gamma(\tilde{x}_l), \tilde{l}_l\right)$ in G_e, and $s \in (\sum/\sum_{oi})^*\alpha$, then $\left(\left(\varepsilon_\gamma(\widetilde{x_l'}), \tilde{l}_l'\right), \left(\varepsilon_\gamma(\tilde{x}_l), \tilde{l}_l\right)\right) \in \widetilde{q_d^e}$. Where $\left(\varepsilon_\gamma(\widetilde{x_l'}), \tilde{l}_l'\right)$ corresponds to the precursor of the state corresponding to $\left(\varepsilon_\gamma(\tilde{x}_l), \tilde{l}_l\right)$ in G_e.

The diagnoser introduced above records all the states that the system can reach driven by specific observable event. For the limitation of local information and observation methods, the state transferring of the system driven by unobservable event is not so definite, and according to the second strategy, it is needed to make that definite by cooperating with another diagnoser, so we consider the state transferring in a diagnoser driven by the event unobservable to itself while observable to another.

Def 4 for any state $q_d^e = \{((\varepsilon_\gamma(x_1'), l_1'), (\varepsilon_\gamma(x_1), l_1)), \cdots, ((\varepsilon(x_n'), l_n'), (\varepsilon_\gamma(x_n), l_n))\}$

$(l_i \in \Delta)$ in a distributed diagnoser $G_{di}^e(i \in \{1,2\})$, $S_i(q_d^e) = \left\{s \in (\sum/\sum_{oi})^* : s \in L_\sigma\left(G_e, (\varepsilon_\gamma(x_k), l_k)\right)\right\}$ is an path set, where $\sigma \in \Sigma_{oj}$, $j \in \{1, 2\}/\{i\}$, $k \in \{1, \cdots, n\}$, and $s \in L_\sigma\left(G_e, (\varepsilon_\gamma(x_k), l_k)\right)$ denotes those event paths that start from the state corresponding to $(\varepsilon_\gamma(x_k), l_k)$ in $G_e$and end with specific event σ, then the state set that q_d^e can reach driven by the unobservable event in $\sum/\sum_{oi}$ can be denoted as follows:

$$UR_i(q_d^e) = \{q_d^e\} \cup \bigcup_{s \in S_i(q_d^e)} \left\{\left((\varepsilon_\gamma(\widetilde{x_k}'), \widetilde{l_k}'), (\varepsilon_\gamma(\widetilde{x_k}), \widetilde{l_k})\right)\right\}$$

Where $k \in \{1, \cdots, n\}$, $(\varepsilon_\gamma(\widetilde{x_k}), \widetilde{l_k})$ is the state that $(\varepsilon_\gamma(x_k), l_k)$ reaches after the path $s \in S_i(q_d^e)$, and $(\varepsilon_\gamma(\widetilde{x_k}'), \widetilde{l_k}')$ is the precursor of $(\varepsilon_\gamma(\widetilde{x_k}'), \widetilde{l_k}')$ on that path.

Each distributed diangnoser determines the current state of the system based on the knowledge it owns and its observation method, and a coordination agent summaries their diagnostic results to obtain the global diagnostic result. Before introduce the strategy of summary, we introduce two operations definitions first.

Def 5 let $q_{d1}^e = \{((\varepsilon_\gamma(x_1'), l_{x_1}'), (\varepsilon_\gamma(x_1), l_{x_1})), \cdots, ((\varepsilon_\gamma(x_n'), l_{x_n}'), (\varepsilon_\gamma(x_n), l_{x_n}))\}$, $q_{d2}^e = \{((\varepsilon_\gamma(y_1'), l_{y_1}'), (\varepsilon_\gamma(y_1), l_{y_1})), \cdots, ((\varepsilon_\gamma(x_n'), l_{y_m}'), (\varepsilon_\gamma(x_n), l_{y_m}))\}$, the operation $\cap_e^i$ $(i \in \{1,2\})$ can be defined as follows:

$q_{d1}^e \cap_e^i q_{d2}^e \triangleq \{((\varepsilon_\gamma(z'), l_z'), (\varepsilon_\gamma(z), l_z)) \in 2^{(\Theta \times \Delta) \times (\Theta \times \Delta)}, (\varepsilon_\gamma(z), l_z) = (\varepsilon_\gamma(x_k), l_{x_k}) = (\varepsilon_\gamma(y_t), l_{y_t}), k \in \{1, \cdots, n\}, t \in \{1, \cdots, m\}$, if $i=1$, $(\varepsilon_\gamma(z'), l_z') = (\varepsilon_\gamma(x_k'), l_{x_k}')$, if $i=2$, $(\varepsilon_\gamma(z'), l_z') = (\varepsilon_\gamma(y_t'), l_{y_t}')\}$.

The operation $\cap_e^i$ is used to summary the diagnostic results submitted by every distributed diagnoser, and it denotes that for the different event paths to a specific state submitted by different distributed diagnosers, the summary result accepts the diagnostic result submitted by the distributed diagnoser that is more convinced about its path. This operation make it possible for all the distributed diagnosers to make full use of limited information and observation methods to constrain each other, so as to reduce the diagnostic space.

Def 6 let $q_1 = \{((\varepsilon_\gamma(x_1'), l_{x_1}'), (\varepsilon_\gamma(x_1), l_{x_1})), \cdots, ((\varepsilon_\gamma(x_n'), l_{x_n}'), (\varepsilon_\gamma(x_n), l_{x_n}))\}$, $q_0 = \{((\varepsilon_\gamma(y_1'), l_{y_1}'), (\varepsilon_\gamma(y_1), l_{y_1})), \cdots, ((\varepsilon_\gamma(x_n'), l_{y_m}'), (\varepsilon_\gamma(x_n), l_{y_m}))\}$, the operation $\cap_c$ can be defined as follows:

$q_1 \cap_c q_0 = \{((\varepsilon_\gamma(z'), l_z'), (\varepsilon_\gamma(z), l_z)) \in 2^{(\Theta \times \Delta) \times (\Theta \times \Delta)}, (\varepsilon_\gamma(z'), l_z') = (\varepsilon_\gamma(y_t), l_{y_t}) = (\varepsilon_\gamma(x_k'), l_{x_k}'), k \in \{1, \cdots, n\}, t \in \{1, \cdots, m\}, (\varepsilon_\gamma(z), l_z) = (\varepsilon_\gamma(x_k), l_{x_k})\}$.

The operation $\cap_c$ is used to update the global diagnostic result in real time based on the summary result.

The global diagnostic result can be obtained based on the information submitted by all distributed diagnosers. Let q_1, q_2 and q denote the states of distributed diagnosers G_{d1}^e, G_{d2}^e and global diagnoser G_d^e respectively, the event path $s = s_1 a u b$, where $a, b \in \sum_o (\sum_o = \sum_{o1} \cup \sum_{o2}), u \in \sum_{uo}^*$, and q_{old} denotes the state of G_d^e after executing

the event path of $s_1 a$, then the state q after G_d^e executing the event path s can be obtained as follows:

1. If $b \in (\sum_{o1} \cap \sum_{o2})$ then
   (1). $q = (q_1 \cap_e^1 q_2) \cap_c q_{old}$ (when $a \in \sum_{o1}$)
   (2). $q = (q_1 \cap_e^2 q_2) \cap_c q_{old}$ (when $a \in \sum_{o2}$)
2. If $b \in (\sum_{o1} / \sum_{o2})$ then
   (1). $q = \left(q_1 \cap_e^1 UR_2(q_2)\right) \cap_c q_{old}$(when $a \in \sum_{o1}$)
   (2). $q = \left(q_1 \cap_e^2 UR_2(q_2)\right) \cap_c q_{old}$(when $a \in \sum_{o2}$)
3. If $b \in (\sum_{o2} / \sum_{o1})$ then
   (1). $q = (UR_1(q_1) \cap_e^1 q_2) \cap_c q_{old}$(when $a \in \sum_{o1}$)
   (2). $q = (UR_1(q_1) q_2) \cap_c q_{old}$(when $a \in \sum_{o2}$)

The distributed diagnosters can limit the failure states of the system by cooperation and bring a *D-ALCO* based description for every failure state, which lays a foundation for the failure recovery in the next step.

4 Knowledge based failure recovery

4.1 Process of Failure Recovery

Based on the semantic description of failure state, we can recovery it by knowledge-based method. Its process is described as illustrated in Fig.3 below.

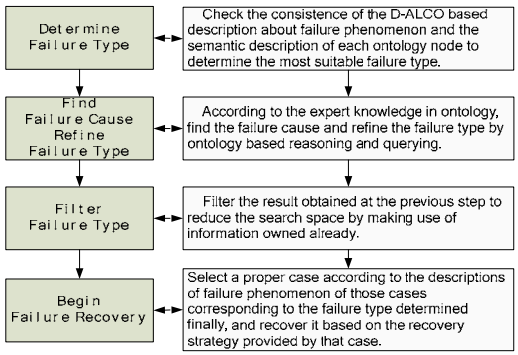

Fig. 3. Semi-automatic failure recovery flowchart

The green blocks diagram on the left denote the recovery steps, and the white block diagram on the right denotes the specific operations corresponding to each step. For the limitation of observation method, the description of failure state is not so exhaustive. People hope to resolve the failure problem in a smaller granularity and not limited to retrying or substitution, so step 2 and step 3 above repeat until there is no more expert knowledge can be used to filter the failure type.

4.2 Construction of Failure Knowledge Ontology

Failure knowledge ontology places an important role in the process of failure recovery. It lays a foundation for reusing expert knowledge automatically, analyzing the failure type accurately and reducing the failure searching space. It also acts as a backbone of knowledge to organize and manage the failure case library, so as to improve the efficiency of case searching and reduce the case searching space.

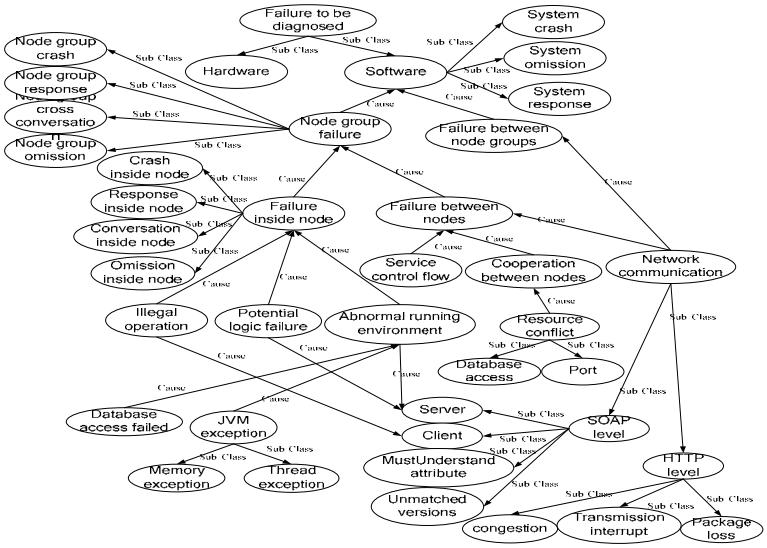

Fig. 4. Failure knowledge ontology

As illustrated in Fig.4 above, the failure knowledge ontology is constructed according the failure granularity and their causal relationship directed by experts. Each failure type node in it is described by a group of simple formula sets, and all failure cases are mapped with proper concepts in it according to their failure types.

4.3 Mapping between Failure Ontology and Case Library

The mapping between the concepts and the cases plays an important role during the process of failure recovery, and it is constructed as following code illustrated:

```
<O2DMappings srcOntology="DiagnosisOntology">
 <O2DMapping>
  <concept name="NodeFailure"/>
   <mappingExpression>
  <queryStatement name="query1"destSource="DiagnosisLibDB">
     <expression>
        Model.list ResourcesWith Property(symrec.FaultString,
           "NodeFailure")
     </expression>
     <objectKey >
```

```
        <keyAttribute name="recordid" type="string"/>
      </objectKey>
     </queryStatement>
    </mappingExpression>
</O2DMapping>
             ...
</O2DMappings>
```

Where "NodeFailure" denotes a concept node in it, and "DiagnosisLibDB" denotes the failure case library. The code denotes constructing all the cases that match the query "query1" with the ontology concept "NodeFailure".

5 Conclusion

This paper proposes a D-ALCO based failure diagnostic method for a multi-agent based software architecture. It has some advantages as follows:

（1） It can be real-time and be transparent to the system to be diagnosed;

（2） It can find the running failure of the system and determine it type and location without manual intervention, so as to support the recovery next.

（3） It is general and independent. It describes the complex interactions between the components of a system by a concise model independent of it, so as to shorten the development cycle of the diagnostic system and improve the diagnostic efficiency by separating the reasoning kernel of the diagnostic system and the model of the system to be diagnosed.

References

1. Fabre, E., et al., Fault detection and diagnosis in distributed systems: an approach by partially stochastic Petri nets. Discrete Event Dynamic Systems: Theory and Applications, 1998. 8: p. 203-231.
2. Genc, S. and S. Lafortune, Distributed Diagnosis of Discrete-Event Systems Using Petri Nets. Lecture Notes in Computer Science (LNCS), 2003. 2679(316-336).
3. Sampath, M., et al., Failure Diagnosis Using Discrete-Event Models. IEEE TRANSACTION ON CONTROL SYSTEMS TECHNOLOGY, 1996. 4(2): p. 105 – 124.
4. DEBOUK, R., S. LAFORTUNE, and D. TENEKETZIS, Coodinated Decentralized Protocols for Failure Diagnosis of Discrete Event Systems. Discrete Event Dynamic Systems: Theory and Applications, 2000. 10: p. 33-86.
5. Pencole, Y. and M.-O. Cordier, A formal framework for the decentralised diagnosis of large scale discrete event systems and its application to telecommunication networks. Artificial Intelligence, 2005. 164: p. 121-170.
6. Yang, F.-q., et al., Some Discussion on the Development of Software Technology ACTA ELECTRONICA SINICA, 2002. 30: p. 1901-1906.
7. Chang, L., et al., A tableau decision algorithm for dynamic description logic. Chinese Journal of Computers, 2008. 31(6): p. 896-909.